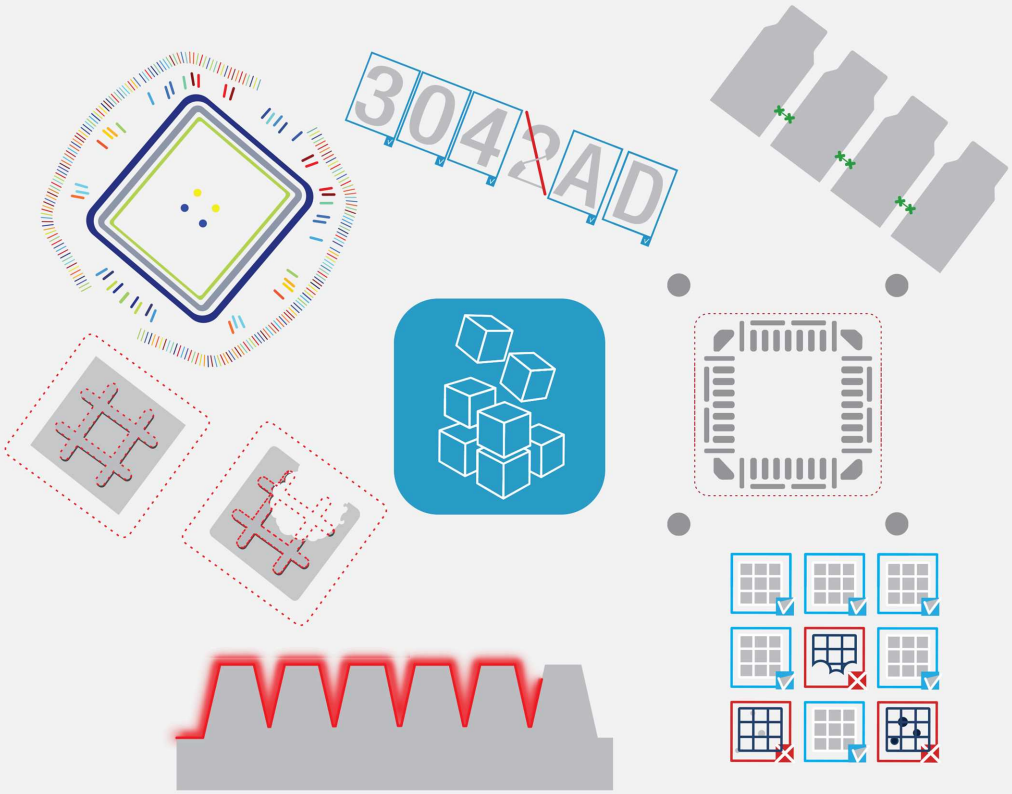
euresys
Empowering Computer Vision

REFERENCE MANUAL

Open eVision



This documentation is provided with **Open eVision 24.02.0** (doc build **1197**).
www.euresys.com

Contents

1. Pixel Accessors 112
1.1. EBW8PixelAccessor Class 112
EBW8PixelAccessor::EBW8PixelAccessor 112
EBW8PixelAccessor::GetPixel 112
EBW8PixelAccessor::SetPixel 113
1.2. EBW16PixelAccessor Class 113
EBW16PixelAccessor::EBW16PixelAccessor 113
EBW16PixelAccessor::GetPixel 114
EBW16PixelAccessor::SetPixel 114
1.3. EBW32PixelAccessor Class 115
EBW32PixelAccessor::EBW32PixelAccessor 115
EBW32PixelAccessor::GetPixel 115
EBW32PixelAccessor::SetPixel 116
1.4. EC15PixelAccessor Class 116
EC15PixelAccessor::EC15PixelAccessor 117
EC15PixelAccessor::GetPixel 117
EC15PixelAccessor::SetPixel 117
1.5. EC16PixelAccessor Class 118
EC16PixelAccessor::EC16PixelAccessor 118
EC16PixelAccessor::GetPixel 118
EC16PixelAccessor::SetPixel 119
1.6. EC24APixelAccessor Class 119
EC24APixelAccessor::EC24APixelAccessor 120
EC24APixelAccessor::GetPixel 120
EC24APixelAccessor::SetPixel 120
1.7. EC24PixelAccessor Class 121
EC24PixelAccessor::EC24PixelAccessor 121
EC24PixelAccessor::GetPixel 121
EC24PixelAccessor::SetPixel 122
2. Common 123
2.1. Easy Class 123
2.2. Image and ROI Classes 124
2.3. Region Classes 124
3. Libraries 125
3.1. Easy3D Library 126
3.2. Easy3DLaserLine Library 127
3.3. Easy3DObject Library 127
3.4. Easy3DMatch Library 128
3.5. EasyImage Library 128
3.6. EasyColor Library 129
3.7. EasyObject Library 130
3.8. EasyMatch Library 131
3.9. EasyFind Library 131
3.10. EasyGauge Library 132
3.11. EasyOCR Library 132
3.12. EasyOCR2 Library 133
3.13. EasyBarCode Library 133
3.14. EasyBarCode2 Library 134
3.15. EasyMatrixCode Library 134
3.16. EasyMatrixCode2 Library 134
3.17. EasyQRCode Library 135
3.18. EasyClassify Library 135
3.19. EasySegment Library 136

3.20. EasyLocate Library 136
3.21. Legacy 137
EasyObject Library (Legacy) 137
4. Classes 138
4.1. E3DAligner Class 138
E3DAligner::Align 139
E3DAligner::ClearReprojectionPlane 141
E3DAligner::E3DAligner 141
E3DAligner::IsReprojectionPlaneSet 142
E3DAligner::Load 142
E3DAligner::operator= 143
E3DAligner::RetrieveReferencePoses 143
E3DAligner::RetrieveReferencePosesProjections 143
E3DAligner::Save 144
E3DAligner::GetScanReprojectionPlane 144
E3DAligner::SetScanReprojectionPlane 144
E3DAligner::SetFlatScan 145
E3DAligner::SetReference 146
4.2. E3DAlignment Class 148
E3DAlignment::E3DAlignment 149
E3DAlignment::GetError 149
E3DAlignment::operator= 149
E3DAlignment::GetPose 150
E3DAlignment::GetRefPoseMatchedIndex 150
4.3. E3DAnomaly Class 150
E3DAnomaly::GetArea 151
E3DAnomaly::GetBoundingBox 151
E3DAnomaly::GetCenterOfGravity 151
E3DAnomaly::GetCloud 151
E3DAnomaly::Create3DObject 151
E3DAnomaly::E3DAnomaly 152
E3DAnomaly::operator= 152
4.4. E3DAxisDisplay Class 152
E3DAxisDisplay::GetAxisGraduationColor 153
E3DAxisDisplay::SetAxisGraduationColor 153
E3DAxisDisplay::GetAxisOrigin 154
E3DAxisDisplay::SetAxisOrigin 154
E3DAxisDisplay::GetAxisOriginUserDefined 154
E3DAxisDisplay::SetAxisOriginUserDefined 154
E3DAxisDisplay::GetAxisSize 154
E3DAxisDisplay::SetAxisSize 154
E3DAxisDisplay::GetAxisXColor 155
E3DAxisDisplay::SetAxisXColor 155
E3DAxisDisplay::GetAxisYColor 155
E3DAxisDisplay::SetAxisYColor 155
E3DAxisDisplay::GetAxisZColor 156
E3DAxisDisplay::SetAxisZColor 156
E3DAxisDisplay::E3DAxisDisplay 156
E3DAxisDisplay::GetGridColor 156
E3DAxisDisplay::SetGridColor 156
E3DAxisDisplay::operator= 157
E3DAxisDisplay::operator== 157
E3DAxisDisplay::GetRenderAxis 157
E3DAxisDisplay::SetRenderAxis 157
E3DAxisDisplay::GetRenderGrid 158
E3DAxisDisplay::SetRenderGrid 158
E3DAxisDisplay::GetRenderGridStep 158
E3DAxisDisplay::SetRenderGridStep 158

4.5. E3DAxisSystem Class 158
E3DAxisSystem::GetAxisX 159
E3DAxisSystem::GetAxisY 159
E3DAxisSystem::GetAxisZ 160
E3DAxisSystem::CheckIsNormal 160
E3DAxisSystem::CheckIsOrthogonal 160
E3DAxisSystem::CheckIsRightHanded 161
E3DAxisSystem::E3DAxisSystem 161
E3DAxisSystem::IsNormal 162
E3DAxisSystem::IsOrthogonal 162
E3DAxisSystem::IsRightHanded 163
E3DAxisSystem::Load 163
E3DAxisSystem::GetNormX 163
E3DAxisSystem::GetNormY 163
E3DAxisSystem::GetNormZ 164
E3DAxisSystem::operator!= 164
E3DAxisSystem::operator= 164
E3DAxisSystem::operator== 165
E3DAxisSystem::GetOrigin 165
E3DAxisSystem::Save 165
4.6. E3DBox Class 166
E3DBox::GetAxes 167
E3DBox::SetAxes 167
E3DBox::GetCenter 167
E3DBox::E3DBox 167
E3DBox::Load 169
E3DBox::operator!= 169
E3DBox::operator= 169
E3DBox::operator== 170
E3DBox::Save 170
E3DBox::Transform 171
E3DBox::GetXAxis 171
E3DBox::GetXSize 171
E3DBox::SetXSize 171
E3DBox::GetXYQuadrangle 171
E3DBox::GetYAxis 172
E3DBox::GetYSize 172
E3DBox::SetYSize 172
E3DBox::GetZAxis 172
E3DBox::GetZSize 172
E3DBox::SetZSize 172
4.7. E3DComparer Class 173
E3DComparer::GetAutomaticCropFactor 175
E3DComparer::SetAutomaticCropFactor 175
E3DComparer::Compare 175
E3DComparer::GetComparisonDistanceMode 176
E3DComparer::SetComparisonDistanceMode 176
E3DComparer::ComputesAnomalies 176
E3DComparer::GetDontCare 176
E3DComparer::SetDontCare 176
E3DComparer::E3DComparer 176
E3DComparer::GetEnableAutomaticDecimation 177
E3DComparer::SetEnableAutomaticDecimation 177
E3DComparer::GetEnableAutomaticEdgeCropping 177
E3DComparer::SetEnableAutomaticEdgeCropping 177
E3DComparer::GetAnomalyHysteresis 178
E3DComparer::GetAnomalyThresholds 178
E3DComparer::GetComparisonPointCloud 179
E3DComparer::GetEdgeCroppingParameters 179

E3DComparer::Load ... 180
180
E3DComparer::SetMeshReference ... 180
E3DComparer::GetNoExtraMaterial ... 181
E3DComparer::SetNoExtraMaterial ... 181
E3DComparer::operator= ... 181
181
E3DComparer::SetPointCloudReference ... 181
E3DComparer::PrepareReference ... 182
E3DComparer::GetROI ... 182
E3DComparer::SetROI ... 182
E3DComparer::Save ... 182
E3DComparer::SetAnomalyHysteresis ... 183
E3DComparer::SetAnomalyThresholds ... 183
E3DComparer::SetEdgeCroppingParameters ... 184
4.8. E3DLine Class ... 184
E3DLine::Define ... 185
E3DLine::E3DLine ... 185
E3DLine::GetFirstPoint ... 186
E3DLine::Load ... 186
E3DLine::operator!= ... 186
E3DLine::operator= ... 187
E3DLine::operator== ... 187
E3DLine::Save ... 187
E3DLine::GetSecondPoint ... 188
4.9. E3DMatch Class ... 188
E3DMatch::GetAnomalies ... 188
E3DMatch::E3DMatch ... 189
E3DMatch::operator= ... 189
4.10. E3DMatcher Class ... 189
E3DMatcher::GetAllComparisonNoExtraMaterial ... 192
E3DMatcher::SetAllComparisonNoExtraMaterial ... 192
E3DMatcher::GetAllComparisonROI ... 193
E3DMatcher::SetAllComparisonROI ... 193
E3DMatcher::GetAutomaticCropFactor ... 193
E3DMatcher::SetAutomaticCropFactor ... 193
E3DMatcher::ClearComparisonNoExtraMaterial ... 193
E3DMatcher::ClearComparisonROI ... 194
E3DMatcher::GetComparisonDistanceMode ... 194
E3DMatcher::SetComparisonDistanceMode ... 194
E3DMatcher::E3DMatcher ... 194
E3DMatcher::GetEnableAutomaticDecimation ... 195
E3DMatcher::SetEnableAutomaticDecimation ... 195
E3DMatcher::GetEnableAutomaticEdgeCropping ... 195
E3DMatcher::SetEnableAutomaticEdgeCropping ... 195
E3DMatcher::GetEnableMissingPointAsAnomaly ... 195
E3DMatcher::SetEnableMissingPointAsAnomaly ... 195
E3DMatcher::GetAnomalyHysteresis ... 196
E3DMatcher::GetAnomalyThresholds ... 196
E3DMatcher::GetComparisonNoExtraMaterial ... 197
E3DMatcher::GetComparisonPointCloud ... 197
E3DMatcher::GetComparisonROI ... 198
E3DMatcher::GetEdgeCroppingParameters ... 198
E3DMatcher::Load ... 199
E3DMatcher::Match ... 199
E3DMatcher::operator= ... 201
E3DMatcher::PrepareReference ... 202
E3DMatcher::Save ... 202
E3DMatcher::SetAnomalyHysteresis ... 202

E3DMatcher::SetAnomalyThresholds ... 203
E3DMatcher::SetComparisonNoExtraMaterial ... 203
E3DMatcher::SetComparisonROI ... 204
E3DMatcher::SetEdgeCroppingParameters ... 204
4.11. E3DObject Class ... 205
E3DObject::GetArea ... 206
E3DObject::GetAspectRatio ... 207
E3DObject::GetAveragePosition ... 207
E3DObject::GetBasePlane ... 207
E3DObject::GetBaseTilt ... 207
E3DObject::GetBoundingBox ... 208
E3DObject::Draw ... 208
E3DObject::E3DObject ... 209
E3DObject::GetIsFeatureComputed ... 209
E3DObject::GetLength ... 210
E3DObject::Load ... 210
E3DObject::GetLocalHeight ... 211
E3DObject::GetLocalTilt ... 211
E3DObject::GetLocalTopPosition ... 211
E3DObject::GetNumPixels ... 211
E3DObject::operator= ... 211
E3DObject::operator== ... 212
E3DObject::GetOrientation ... 212
E3DObject::GetPlane ... 212
E3DObject::GetRectangleRegion ... 213
E3DObject::GetReferenceHeight ... 213
E3DObject::GetReferenceTilt ... 213
E3DObject::GetReferenceTopPosition ... 213
E3DObject::GetRegion ... 213
E3DObject::Save ... 214
E3DObject::GetSphere ... 214
E3DObject::Transform ... 214
E3DObject::GetVolume ... 215
E3DObject::GetWidth ... 215
4.12. E3DObjectExtractor Class ... 215
E3DObjectExtractor::AddToMesh ... 218
E3DObjectExtractor::GetAreaRange ... 219
E3DObjectExtractor::SetAreaRange ... 219
E3DObjectExtractor::GetAspectRatioRange ... 219
E3DObjectExtractor::SetAspectRatioRange ... 219
E3DObjectExtractor::GetBackgroundMask ... 219
E3DObjectExtractor::GetContourReinforce ... 220
E3DObjectExtractor::SetContourReinforce ... 220
E3DObjectExtractor::Draw ... 220
E3DObjectExtractor::E3DObjectExtractor ... 221
E3DObjectExtractor::Extract ... 221
E3DObjectExtractor::GetExtractionSensitivity ... 222
E3DObjectExtractor::SetExtractionSensitivity ... 222
E3DObjectExtractor::GetComputeFeature ... 223
E3DObjectExtractor::GetLengthRange ... 223
E3DObjectExtractor::SetLengthRange ... 223
E3DObjectExtractor::Load ... 223
E3DObjectExtractor::GetLocalHeightRange ... 224
E3DObjectExtractor::SetLocalHeightRange ... 224
E3DObjectExtractor::GetLocalTiltRange ... 224
E3DObjectExtractor::SetLocalTiltRange ... 224
E3DObjectExtractor::GetObjects ... 225
E3DObjectExtractor::GetObjectsMask ... 225
E3DObjectExtractor::operator= ... 225

E3DObjectExtractor::operator== 225
E3DObjectExtractor::GetOrientationRange 226
E3DObjectExtractor::SetOrientationRange 226
E3DObjectExtractor::GetOverlappedAreaRatio 226
E3DObjectExtractor::SetOverlappedAreaRatio 226
E3DObjectExtractor::GetOverlappedHeightDifference 226
E3DObjectExtractor::SetOverlappedHeightDifference 226
E3DObjectExtractor::GetOverlappedObject 227
E3DObjectExtractor::SetOverlappedObject 227
E3DObjectExtractor::GetReferenceHeightRange 227
E3DObjectExtractor::SetReferenceHeightRange 227
E3DObjectExtractor::GetReferenceTiltRange 228
E3DObjectExtractor::SetReferenceTiltRange 228
E3DObjectExtractor::Save 228
E3DObjectExtractor::SetAllComputeFeatures 228
E3DObjectExtractor::SetComputeFeature 229
E3DObjectExtractor::UnsetAllComputeFeatures 229
E3DObjectExtractor::GetVolumeRange 229
E3DObjectExtractor::SetVolumeRange 229
E3DObjectExtractor::GetWidthRange 230
E3DObjectExtractor::SetWidthRange 230
4.13. E3DOrthonormalAxisSystem Class 230
E3DOrthonormalAxisSystem::E3DOrthonormalAxisSystem 230
E3DOrthonormalAxisSystem::operator= 231
4.14. E3DPlane Class 231
E3DPlane::AngleWithPlane 233
E3DPlane::Define 233
E3DPlane::DistanceTo 234
E3DPlane::E3DPlane 234
E3DPlane::GetTransformationTo 235
E3DPlane::IntersectionWithTwoPlanes 236
E3DPlane::Load 236
E3DPlane::GetNormal 237
E3DPlane::SetNormal 237
E3DPlane::operator- 237
E3DPlane::operator+ 237
E3DPlane::operator= 238
E3DPlane::ProjectPoint 238
E3DPlane::Save 238
E3DPlane::GetSignedDistanceFromOrigin 239
E3DPlane::SetSignedDistanceFromOrigin 239
E3DPlane::Transform 239
E3DPlane::XPlane 239
E3DPlane::YPlane 240
E3DPlane::ZPlane 240
4.15. E3DRightOrthonormalAxisSystem Class 240
E3DRightOrthonormalAxisSystem::E3DRightOrthonormalAxisSystem 240
E3DRightOrthonormalAxisSystem::operator= 241
4.16. E3DSphere Class 242
E3DSphere::GetCenter 242
E3DSphere::Define 242
E3DSphere::DistanceTo 243
E3DSphere::E3DSphere 243
E3DSphere::GenerateMesh 244
E3DSphere::Load 244
E3DSphere::operator= 245
E3DSphere::GetRadius 245
E3DSphere::Save 245
E3DSphere::Transform 246

4.17. E3DTransformMatrix Class 246
E3DTransformMatrix::CreateAnisotropicScalingMatrix 247
E3DTransformMatrix::CreateIdentityMatrix 248
E3DTransformMatrix::CreateIsotropicScalingMatrix 248
E3DTransformMatrix::CreateOrthoBasis 248
E3DTransformMatrix::CreateOrthographicProjectionMatrix 249
E3DTransformMatrix::CreatePerspectiveProjectionMatrix 249
E3DTransformMatrix::CreateRotationMatrix 250
E3DTransformMatrix::CreateRotationXMatrix 250
E3DTransformMatrix::CreateRotationYMatrix 251
E3DTransformMatrix::CreateRotationZMatrix 251
E3DTransformMatrix::CreateTranslationMatrix 251
E3DTransformMatrix::E3DTransformMatrix 252
E3DTransformMatrix::GetEulerAngles 254
E3DTransformMatrix::GetAzimuthElevationAngles 254
E3DTransformMatrix::GetOrthoBasis 255
E3DTransformMatrix::GetValue 256
E3DTransformMatrix::Inverse 256
E3DTransformMatrix::IsRigid 256
E3DTransformMatrix::Load 257
E3DTransformMatrix::operator- 257
E3DTransformMatrix::operator!= 257
E3DTransformMatrix::operator* 258
E3DTransformMatrix::operator+ 258
E3DTransformMatrix::operator= 259
E3DTransformMatrix::operator== 259
E3DTransformMatrix::Save 259
E3DTransformMatrix::SetValue 260
E3DTransformMatrix::Transpose 260
4.18. E3DViewer Class 260
E3DViewer::AddRenderSource 266
E3DViewer::AddTextLabel 267
E3DViewer::GetAxisOrigin 269
E3DViewer::SetAxisOrigin 269
E3DViewer::GetBackgroundColor 269
E3DViewer::SetBackgroundColor 269
E3DViewer::ClearRenderSource 269
E3DViewer::ClearTextLabels 270
E3DViewer::GetColorRampGraduationColor 270
E3DViewer::SetColorRampGraduationColor 270
E3DViewer::GetColorRampMode 270
E3DViewer::SetColorRampMode 270
E3DViewer::ConfigureRenderSource 271
E3DViewer::DecPointSize 272
E3DViewer::DisableFixColorRampBounds 272
E3DViewer::E3DViewer 272
E3DViewer::EditTextLabel 273
E3DViewer::GetEDLShadingFactor 275
E3DViewer::SetEDLShadingFactor 275
E3DViewer::GetEnableEDLShading 275
E3DViewer::SetEnableEDLShading 275
E3DViewer::GetEnableFixColorRampBounds 275
E3DViewer::GetEnableSmartColorRamp 276
E3DViewer::SetEnableSmartColorRamp 276
E3DViewer::GetFieldOfView 276
E3DViewer::SetFieldOfView 276
E3DViewer::GetFontPath 276
E3DViewer::SetFontPath 276
E3DViewer::GenerateColors 277

E3DViewer::GetAutoRotate ... 277
E3DViewer::GetColorRampLocation ... 278
E3DViewer::GetFeatureStyleFor3DObject ... 278
E3DViewer::GetFixColorRampBounds ... 279
E3DViewer::GetRenderSourceColorMode ... 279
E3DViewer::GetRenderSourceConstantColor ... 280
E3DViewer::GetRenderSourceName ... 280
E3DViewer::GetRenderSourceOpacity ... 280
E3DViewer::GetRenderSourcePointSize ... 281
E3DViewer::GetRenderSourceWireFrame ... 281
E3DViewer::GetRenderSourceWireFrameColor ... 281
E3DViewer::GetRotationMatrix ... 282
E3DViewer::GetTextLabel ... 282
E3DViewer::GetViewAngle ... 283
E3DViewer::GetViewTarget ... 284
E3DViewer::Has3DObjects ... 284
E3DViewer::HasRenderSource ... 285
E3DViewer::HideColorRampLegend ... 285
E3DViewer::HideFeatureFor3DObject ... 285
E3DViewer::HideFeatureForAll3DObjects ... 286
E3DViewer::HideRenderSource ... 286
E3DViewer::IncPointSize ... 286
E3DViewer::InitRendering ... 287
E3DViewer::IsAutoRotate ... 287
E3DViewer::IsColorRampLegendVisible ... 287
E3DViewer::IsEDLShadingSupported ... 287
E3DViewer::IsFeatureVisibleFor3DObject ... 288
E3DViewer::IsRenderSourceVisible ... 288
E3DViewer::GetLastPickedPoint ... 289
E3DViewer::LockRotationFinalPosition ... 289
E3DViewer::LockRotationInitialPosition ... 289
E3DViewer::LockTranslationFinalPosition ... 290
E3DViewer::LockTranslationInitialPosition ... 290
E3DViewer::GetNumRenderSources ... 291
E3DViewer::Pick3DPoint ... 291
E3DViewer::GetPickingDisplay ... 292
E3DViewer::SetPickingDisplay ... 292
E3DViewer::GetPickingDistanceThreshold ... 292
E3DViewer::SetPickingDistanceThreshold ... 292
E3DViewer::GetPickingLabelColor ... 292
E3DViewer::SetPickingLabelColor ... 292
E3DViewer::GetPickingLabelFixed ... 293
E3DViewer::SetPickingLabelFixed ... 293
E3DViewer::GetPickingLabelSize ... 293
E3DViewer::SetPickingLabelSize ... 293
E3DViewer::GetPointSize ... 293
E3DViewer::SetPointSize ... 293
E3DViewer::GetProjectionType ... 294
E3DViewer::SetProjectionType ... 294
E3DViewer::Register3DObjects ... 294
E3DViewer::RemoveAllRenderSources ... 295
E3DViewer::RemoveCurrent3DObjects ... 295
E3DViewer::RemoveRenderSource ... 295
E3DViewer::RemoveTextLabel ... 295
E3DViewer::GetRenderAxis ... 296
E3DViewer::SetRenderAxis ... 296
E3DViewer::GetRenderAxisConfiguration ... 296
E3DViewer::SetRenderAxisConfiguration ... 296
E3DViewer::GetRenderDecimationLevel ... 297

E3DViewer::SetRenderDecimationLevel ... 297
E3DViewer::GetRenderGrid ... 297
E3DViewer::SetRenderGrid ... 297
E3DViewer::GetRenderGridStep ... 297
E3DViewer::SetRenderGridStep ... 297
E3DViewer::ResetPicking ... 298
E3DViewer::ResetView ... 298
E3DViewer::Resize ... 298
E3DViewer::SetAutoRotate ... 299
E3DViewer::SetColorRampLocation ... 299
E3DViewer::SetFeatureStyleFor3DObject ... 300
E3DViewer::SetFeatureStyleForAll3DObjects ... 300
E3DViewer::SetFixColorRampBounds ... 301
E3DViewer::SetFocus ... 301
E3DViewer::SetPickedPointCallBack ... 302
E3DViewer::SetPosition ... 302
E3DViewer::SetRenderSource ... 303
E3DViewer::SetRenderSourceColorMode ... 304
E3DViewer::SetRenderSourceConstantColor ... 305
E3DViewer::SetRenderSourceOpacity ... 305
E3DViewer::SetRenderSourcePointSize ... 306
E3DViewer::SetRenderSourceWireFrame ... 306
E3DViewer::SetRenderSourceWireFrameColor ... 306
E3DViewer::SetRotationMatrix ... 307
E3DViewer::SetViewAngle ... 307
E3DViewer::SetViewTarget ... 308
E3DViewer::Show ... 308
E3DViewer::ShowColorRampLegend ... 308
E3DViewer::ShowFeatureFor3DObject ... 309
E3DViewer::ShowFeatureForAll3DObjects ... 309
E3DViewer::ShowRenderSource ... 309
E3DViewer::StopAutoRotate ... 310
E3DViewer::ToggleRenderAxis ... 310
E3DViewer::ToggleWireframeMode ... 310
E3DViewer::UpdateRotationPosition ... 311
E3DViewer::UpdateTranslationPosition ... 311
E3DViewer::UpdateViewDistance ... 312
E3DViewer::GetViewDistance ... 312
E3DViewer::SetViewDistance ... 312
E3DViewer::GetWireframeMode ... 312
E3DViewer::SetWireframeMode ... 312
4.19. EAffineTransformer Class ... 313
EAffineTransformer::AddAnisotropicScalingTransform ... 314
EAffineTransformer::AddIsotropicScalingTransform ... 314
EAffineTransformer::AddOrthographicProjectionTransform ... 315
EAffineTransformer::AddPerspectiveProjectionTransform ... 315
EAffineTransformer::AddRotationXTransform ... 316
EAffineTransformer::AddRotationYTransform ... 316
EAffineTransformer::AddRotationZTransform ... 316
EAffineTransformer::AddTransform ... 317
EAffineTransformer::AddTranslationTransform ... 317
EAffineTransformer::ApplyMatrix ... 318
EAffineTransformer::ApplyTransform ... 318
EAffineTransformer::CreateAnisotropicScalingMatrix ... 319
EAffineTransformer::CreateIdentityMatrix ... 320
EAffineTransformer::CreateIsotropicScalingMatrix ... 320
EAffineTransformer::CreateOrthographicProjectionMatrix ... 320
EAffineTransformer::CreatePerspectiveProjectionMatrix ... 321
EAffineTransformer::CreateRotationXMatrix ... 321

EAffineTransformer::CreateRotationYMatrix ... 321
EAffineTransformer::CreateRotationZMatrix ... 322
EAffineTransformer::CreateTranslationMatrix ... 322
EAffineTransformer::EAffineTransformer ... 322
EAffineTransformer::operator= ... 323
EAffineTransformer::Reset ... 323
EAffineTransformer::GetTransform ... 323
EAffineTransformer::SetTransform ... 323
4.20. EAngleRectifier Class ... 324
EAngleRectifier::Rectify ... 324
4.21. Easy Class ... 324
Easy::GetAngleUnit ... 326
Easy::SetAngleUnit ... 326
Easy::CheckLicense ... 327
Easy::CheckLicenses ... 327
Easy::CheckOemKey ... 327
Easy::CloseImageGraphicContext ... 328
Easy::GetEnableEnhancedImageDisplay ... 329
Easy::SetEnableEnhancedImageDisplay ... 329
Easy::FromDegrees ... 329
Easy::FromRadians ... 329
Easy::GetBestMatchingImageType ... 330
Easy::GetDongleCount ... 330
Easy::GetDongleInternalSerialNumber ... 330
Easy::GetErrorText ... 331
Easy::GetGPUComputeCapability ... 331
Easy::GetGPUName ... 331
Easy::Initialize ... 332
Easy::IsGPUAvailable ... 332
Easy::LogInMemento ... 332
Easy::GetMaxNumberOfProcessingThreads ... 333
Easy::SetMaxNumberOfProcessingThreads ... 333
Easy::GetNumberOfAvailableProcessorCores ... 333
Easy::GetNumGPUs ... 333
Easy::OpenImageGraphicContext ... 334
Easy::Render3D ... 334
Easy::RenderColorHistogram ... 336
Easy::Resize ... 337
Easy::GetResourcesRootPath ... 338
Easy::GetSampleImagesRootPath ... 338
Easy::GetSampleProgramsRootPath ... 338
Easy::SetOemKey ... 338
Easy::StartTiming ... 339
Easy::StopTiming ... 339
Easy::Terminate ... 340
Easy::ToDegrees ... 340
Easy::ToRadians ... 340
Easy::TrueTimingResolution ... 341
Easy::GetVersion ... 341
4.22. EasyColor Class ... 341
EasyColor::AlphaBlend ... 344
EasyColor::AssignNearestClass ... 344
EasyColor::AssignNearestClassCenter ... 345
EasyColor::BayerToC24 ... 346
EasyColor::C24ToBayer ... 347
EasyColor::GetCieAB ... 348
EasyColor::GetCieAG ... 348
EasyColor::GetCieAR ... 349
EasyColor::GetCieD50B ... 349

EasyColor::GetCieD50G ... 349
EasyColor::GetCieD50R ... 349
EasyColor::GetCieD55B ... 349
EasyColor::GetCieD55G ... 350
EasyColor::GetCieD55R ... 350
EasyColor::GetCieD65B ... 350
EasyColor::GetCieD65G ... 350
EasyColor::GetCieD65R ... 350
EasyColor::GetCieFB ... 351
EasyColor::GetCieFG ... 351
EasyColor::GetCieFR ... 351
EasyColor::ClassAverages ... 351
EasyColor::ClassVariances ... 352
EasyColor::GetCompensateNtscGamma ... 353
EasyColor::GetCompensatePalGamma ... 353
EasyColor::GetCompensateSmpteGamma ... 353
EasyColor::Compose ... 354
EasyColor::Decompose ... 355
EasyColor::Dequantize ... 355
EasyColor::GetDstQuantization ... 357
EasyColor::SetDstQuantization ... 357
EasyColor::Format422To444 ... 357
EasyColor::Format444To422 ... 358
EasyColor::GetComponent ... 358
EasyColor::ImproveClassCenters ... 359
EasyColor::IshToRgb ... 360
EasyColor::LabToRgb ... 360
EasyColor::LabToXyz ... 361
EasyColor::LchToRgb ... 362
EasyColor::LshToRgb ... 362
EasyColor::LuvToRgb ... 363
EasyColor::LuvToXyz ... 363
EasyColor::GetNtscGamma ... 364
EasyColor::GetPalGamma ... 364
EasyColor::PseudoColor ... 364
EasyColor::Quantize ... 365
EasyColor::RegisterPlanes ... 366
EasyColor::GetRgbStandard ... 367
EasyColor::SetRgbStandard ... 367
EasyColor::RgbToIsh ... 368
EasyColor::RgbToLab ... 368
EasyColor::RgbToLch ... 369
EasyColor::RgbToLsh ... 370
EasyColor::RgbToLuv ... 370
EasyColor::RgbToReducedXyz ... 371
EasyColor::RgbToVsh ... 371
EasyColor::RgbToXyz ... 372
EasyColor::RgbToYiq ... 373
EasyColor::RgbToYsh ... 373
EasyColor::RgbToYuv ... 374
EasyColor::SetComponent ... 374
EasyColor::GetSmpteGamma ... 375
EasyColor::GetSrcQuantization ... 375
EasyColor::SetSrcQuantization ... 375
EasyColor::Transform ... 376
EasyColor::TransformBayer ... 376
EasyColor::VshToRgb ... 377
EasyColor::XyzToLab ... 378
EasyColor::XyzToLuv ... 379

EasyColor::XyzToRgb 379
EasyColor::YiqToRgb 380
EasyColor::YshToRgb 380
EasyColor::YuvToRgb 381
4.23. EasyImage Class 382
EasyImage::AdaptiveThreshold 388
EasyImage::AlphaBlend 389
EasyImage::AnalyseHistogram 390
EasyImage::AnalyseHistogramBW16 390
EasyImage::Area 391
EasyImage::AreaDoubleThreshold 392
EasyImage::ArgumentImage 393
EasyImage::AutoThreshold 394
EasyImage::BiLevelBlackTopHatBox 396
EasyImage::BiLevelBlackTopHatDisk 396
EasyImage::BiLevelCloseBox 397
EasyImage::BiLevelCloseDisk 397
EasyImage::BiLevelDilateBox 398
EasyImage::BiLevelDilateDisk 399
EasyImage::BiLevelErodeBox 399
EasyImage::BiLevelErodeDisk 400
EasyImage::BiLevelMedian 400
EasyImage::BiLevelMorphoGradientBox 401
EasyImage::BiLevelMorphoGradientDisk 402
EasyImage::BiLevelOpenBox 402
EasyImage::BiLevelOpenDisk 403
EasyImage::BiLevelThick 404
EasyImage::BiLevelThin 405
EasyImage::BiLevelWhiteTopHatBox 405
EasyImage::BiLevelWhiteTopHatDisk 406
EasyImage::BinaryMoments 407
EasyImage::BlackTopHatBox 410
EasyImage::BlackTopHatDisk 411
EasyImage::CloseBox 412
EasyImage::CloseDisk 414
EasyImage::Contour 415
EasyImage::Convert 417
EasyImage::ConvertTo422 421
EasyImage::ConvolGabor 422
EasyImage::ConvolGaussian 425
EasyImage::ConvolGradient 426
EasyImage::ConvolGradientX 427
EasyImage::ConvolGradientY 428
EasyImage::ConvolHighpass1 429
EasyImage::ConvolHighpass2 430
EasyImage::ConvolKernel 431
EasyImage::ConvolLaplacian4 433
EasyImage::ConvolLaplacian8 434
EasyImage::ConvolLaplacianX 435
EasyImage::ConvolLaplacianY 436
EasyImage::ConvolLowpass1 437
EasyImage::ConvolLowpass2 438
EasyImage::ConvolLowpass3 439
EasyImage::ConvolPrewitt 440
EasyImage::ConvolPrewittX 441
EasyImage::ConvolPrewittY 442
EasyImage::ConvolRoberts 443
EasyImage::ConvolSobel 444
EasyImage::ConvolSobelX 445

EasyImage::ConvolSobelY ... 446
EasyImage::ConvolSymmetricKernel ... 447
EasyImage::ConvolUniform ... 448
EasyImage::Copy ... 450
EasyImage::CumulateHistogram ... 453
EasyImage::DilateBox ... 454
EasyImage::DilateDisk ... 455
EasyImage::Distance ... 457
EasyImage::DoubleThreshold ... 457
EasyImage::Equalize ... 459
EasyImage::ErodeBox ... 460
EasyImage::ErodeDisk ... 461
EasyImage::Flip ... 462
EasyImage::Focusing ... 463
EasyImage::Gain ... 464
EasyImage::GainOffset ... 465
EasyImage::GetFrame ... 466
EasyImage::GetProfilePeaks ... 466
EasyImage::GradientScalar ... 467
EasyImage::GravityCenter ... 468
EasyImage::HDRFusion ... 470
EasyImage::Histogram ... 471
EasyImage::HistogramThreshold ... 472
EasyImage::HistogramThresholdBW16 ... 473
EasyImage::HitAndMiss ... 474
EasyImage::HorizontalMirror ... 475
EasyImage::ImageToLineSegment ... 475
EasyImage::ImageToPath ... 479
EasyImage::IsodataThreshold ... 481
EasyImage::IsodataThresholdBW16 ... 482
EasyImage::LinearTransform ... 483
EasyImage::LineSegmentToImage ... 484
EasyImage::LocalAverage ... 486
EasyImage::LocalDeviation ... 487
EasyImage::Lut ... 488
EasyImage::MatchFrames ... 489
EasyImage::Median ... 490
EasyImage::ModulusImage ... 491
EasyImage::MorphoGradientBox ... 492
EasyImage::MorphoGradientDisk ... 493
EasyImage::Normalize ... 495
EasyImage::Offset ... 496
EasyImage::OpenBox ... 496
EasyImage::OpenDisk ... 498
EasyImage::Oper ... 499
EasyImage::Overlay ... 503
EasyImage::GetOverlayColor ... 505
EasyImage::SetOverlayColor ... 505
EasyImage::PathToImage ... 505
EasyImage::PixelAverage ... 506
EasyImage::PixelCompare ... 507
EasyImage::PixelCount ... 509
EasyImage::PixelMax ... 512
EasyImage::PixelMaxBW16 ... 513
EasyImage::PixelMaxBW8 ... 513
EasyImage::PixelMin ... 514
EasyImage::PixelMinBW16 ... 515
EasyImage::PixelMinBW8 ... 515
EasyImage::PixelStat ... 515

EasyImage::PixelStatBW16 517
EasyImage::PixelStatBW8 517
EasyImage::PixelStdDev 518
EasyImage::PixelVariance 521
EasyImage::ProfileDerivative 523
EasyImage::ProjectOnAColumn 524
EasyImage::ProjectOnARow 525
EasyImage::RealignFrame 527
EasyImage::RebuildFrame 528
EasyImage::RecursiveAverage 529
EasyImage::Register 530
EasyImage::RmsNoise 534
EasyImage::Rotate 535
EasyImage::ScaleRotate 536
EasyImage::SetCircleWarp 539
EasyImage::SetFrame 540
EasyImage::SetInvCircleWarp 541
EasyImage::SetRecursiveAverageLUT 542
EasyImage::SetupEqualize 543
EasyImage::SetupInverseWarp 543
EasyImage::Shrink 544
EasyImage::SignalNoiseRatio 545
EasyImage::SwapFrames 546
EasyImage::Thick 547
EasyImage::Thin 548
EasyImage::ThreeLevelsMinResidueThreshold 549
EasyImage::Threshold 550
EasyImage::Transpose 554
EasyImage::TwoLevelsMinResidueThreshold 555
EasyImage::Uniformize 556
EasyImage::VerticalMirror 559
EasyImage::Warp 559
EasyImage::WeightedMoments 560
EasyImage::WhiteTopHatBox 568
EasyImage::WhiteTopHatDisk 570
4.24. EasyObject Class 571
EasyObject::ContourArea 572
EasyObject::ContourGravityCenter 572
EasyObject::ContourInertia 573
EasyObject::IsFloatFeature 573
EasyObject::IsIntegerFeature 574
EasyObject::IsUnsignedIntegerFeature 574
4.25. EBarCode Class 575
EBarCode::GetAdditionalSymbologies 577
EBarCode::SetAdditionalSymbologies 577
EBarCode::GetAngle 577
EBarCode::SetAngle 577
EBarCode::GetCenter 577
EBarCode::SetCenter 577
EBarCode::GetCenterX 578
EBarCode::GetCenterY 578
EBarCode::Decode 578
EBarCode::Detect 579
EBarCode::Drag 579
EBarCode::Draw 580
EBarCode::DrawWithCurrentPen 581
EBarCode::EBarCode 581
EBarCode::GetDecodedAngle 581
EBarCode::GetDecodedDirection 582

EBarCode::GetDecodedRectangle ... 583
EBarCode::GetDecodedSymbology ... 583
EBarCode::GetSymbologyName ... 584
EBarCode::HitTest ... 584
EBarCode::GetKnownLocation ... 585
EBarCode::SetKnownLocation ... 585
EBarCode::GetKnownModule ... 585
EBarCode::SetKnownModule ... 585
EBarCode::Load ... 585
EBarCode::GetModule ... 586
EBarCode::SetModule ... 586
EBarCode::GetNumDecodedSymbologies ... 586
EBarCode::GetNumEnabledSymbologies ... 587
EBarCode::Read ... 587
587
EBarCode::SetRectangle ... 587
EBarCode::GetRectangleShape ... 588
EBarCode::GetRelativeReadingSizeX ... 588
EBarCode::GetRelativeReadingSizeY ... 588
EBarCode::GetRelativeReadingX ... 588
EBarCode::GetRelativeReadingY ... 589
EBarCode::Save ... 589
EBarCode::SetCenterXY ... 589
EBarCode::SetReadingCenter ... 590
EBarCode::SetReadingSize ... 590
EBarCode::SetSize ... 591
EBarCode::GetSizeX ... 591
EBarCode::GetSizeY ... 591
EBarCode::GetStandardSymbologies ... 592
EBarCode::SetStandardSymbologies ... 592
EBarCode::GetThicknessRatio ... 592
EBarCode::SetThicknessRatio ... 592
EBarCode::GetVerifyChecksum ... 593
EBarCode::SetVerifyChecksum ... 593
4.26. EBarCode Class ... 593
EBarCode::DrawPosition ... 594
EBarCode::DrawPositionWithCurrentPen ... 594
EBarCode::EBarCode ... 595
EBarCode::GetChecksumOK ... 596
EBarCode::GetDecodedString ... 596
EBarCode::GetGradingParameters ... 597
EBarCode::HasGradingParameters ... 597
EBarCode::operator= ... 597
EBarCode::GetPosition ... 598
EBarCode::GetSymbologies ... 598
EBarCode::GetSymbology ... 598
4.27. EBarCodeGrid Class ... 598
EBarCodeGrid::EBarCodeGrid ... 599
599
EBarCodeGrid::SetEnableAll ... 599
EBarCodeGrid::GetCellEnabled ... 600
EBarCodeGrid::GetResults ... 600
EBarCodeGrid::GetNumCols ... 601
EBarCodeGrid::GetNumRows ... 601
EBarCodeGrid::operator= ... 601
EBarCodeGrid::SetEnableCell ... 601
EBarCodeGrid::SetEnableColumn ... 602
EBarCodeGrid::SetEnableRow ... 602
4.28. EBarCodeReader Class ... 603

EBarCodeReader::GetComputeGrading 606
EBarCodeReader::SetComputeGrading 606
EBarCodeReader::DisableAllSymbologies 606
EBarCodeReader::EBarCodeReader 606
EBarCodeReader::EnableAllSymbologies 607
EBarCodeReader::EnableDefaultSymbologies 607
EBarCodeReader::GetEnabledSymbologies 607
EBarCodeReader::GetEnablePermissiveDecoding 608
EBarCodeReader::SetEnablePermissiveDecoding 608
EBarCodeReader::EnableSymbologies 608
EBarCodeReader::EnableSymbology 608
EBarCodeReader::Learn 609
EBarCodeReader::GetLearningPerformed 609
EBarCodeReader::Load 610
EBarCodeReader::GetMaxNumCodes 610
EBarCodeReader::SetMaxNumCodes 610
EBarCodeReader::GetMinModuleSize 611
EBarCodeReader::SetMinModuleSize 611
EBarCodeReader::operator= 611
EBarCodeReader::Read 611
EBarCodeReader::GetReadingOrientation 613
EBarCodeReader::SetReadingOrientation 613
EBarCodeReader::ResetLearning 613
EBarCodeReader::Save 613
EBarCodeReader::GetTimeOut 614
EBarCodeReader::SetTimeOut 614
EBarCodeReader::GetUseMinModuleSize 614
EBarCodeReader::SetUseMinModuleSize 614
EBarCodeReader::GetValidateBarCode 615
EBarCodeReader::SetValidateBarCode 615
EBarCodeReader::GetValidateMandatoryChecksum 615
EBarCodeReader::SetValidateMandatoryChecksum 615
EBarCodeReader::GetValidateOptionalChecksum 616
EBarCodeReader::SetValidateOptionalChecksum 616
EBarCodeReader::GetValidatePharmacode 616
EBarCodeReader::SetValidatePharmacode 616
EBarCodeReader::GetValidateWithChecksum 617
EBarCodeReader::SetValidateWithChecksum 617
4.29. EBaseROI Class 617
EBaseROI::Attach 620
EBaseROI::GetAuthor 620
EBaseROI::SetAuthor 620
EBaseROI::GetBaseTopParent 621
EBaseROI::GetBitsPerPixel 621
EBaseROI::GetColorSystem 621
EBaseROI::SetColorSystem 621
EBaseROI::GetColPitch 621
EBaseROI::GetComment 622
EBaseROI::SetComment 622
EBaseROI::CopyTo 622
EBaseROI::CropToImage 623
EBaseROI::GetDate 623
EBaseROI::SetDate 623
EBaseROI::Drag 623
EBaseROI::Draw 624
EBaseROI::DrawFrame 626
EBaseROI::DrawFrameWithCurrentPen 628
EBaseROI::GetImagePtr 629
EBaseROI::GetSubBaseROIs 629

EBaseROI::HasSubROI 630
EBaseROI::GetHeight 630
EBaseROI::SetHeight 630
EBaseROI::HitTest 631
EBaseROI::IsAnROI 631
EBaseROI::GetIsVoid 632
EBaseROI::Load 632
EBaseROI::GetOrgX 633
EBaseROI::SetOrgX 633
EBaseROI::GetOrgY 633
EBaseROI::SetOrgY 633
EBaseROI::GetParent 633
EBaseROI::GetPlanesPerPixel 634
EBaseROI::GetRowPitch 634
EBaseROI::Save 634
EBaseROI::SaveJpeg 635
EBaseROI::SaveJpeg2K 635
EBaseROI::SavePng 636
EBaseROI::SetImagePtr 636
EBaseROI::SetPlacement 637
EBaseROI::SetSize 638
EBaseROI::GetTitle 639
EBaseROI::SetTitle 639
EBaseROI::GetTotalHeight 639
EBaseROI::GetTotalOrgX 640
EBaseROI::GetTotalOrgY 640
EBaseROI::GetTotalWidth 640
EBaseROI::GetType 640
EBaseROI::GetWidth 641
EBaseROI::SetWidth 641
4.30. EBinaryImageSegmenter Class 641
EBinaryImageSegmenter::EBinaryImageSegmenter 641
EBinaryImageSegmenter::operator== 642
4.31. EBW16PathVector Class 642
EBW16PathVector::AddElement 643
EBW16PathVector::GetClosed 644
EBW16PathVector::SetClosed 644
EBW16PathVector::Draw 644
EBW16PathVector::DrawClosedContour 645
EBW16PathVector::DrawWithCurrentPen 646
EBW16PathVector::EBW16PathVector 647
EBW16PathVector::GetElement 647
EBW16PathVector::operator[] 648
EBW16PathVector::operator= 648
EBW16PathVector::GetRawDataPtr 649
EBW16PathVector::SetElement 649
4.32. EBW16PixelAccessor Class 649
EBW16PixelAccessor::EBW16PixelAccessor 650
EBW16PixelAccessor::GetPixel 650
EBW16PixelAccessor::SetPixel 650
4.33. EBW16Vector Class 651
EBW16Vector::AddElement 651
EBW16Vector::Draw 652
EBW16Vector::DrawWithCurrentPen 653
EBW16Vector::EBW16Vector 654
EBW16Vector::GetElement 654
EBW16Vector::operator[] 655
EBW16Vector::operator= 655
EBW16Vector::GetRawDataPtr 655

EBW16Vector::SetElement 656
EBW16Vector::WeightedMoment 656
4.34. EBW32PixelAccessor Class 656
EBW32PixelAccessor::EBW32PixelAccessor 657
EBW32PixelAccessor::GetPixel 657
EBW32PixelAccessor::SetPixel 657
4.35. EBW32Vector Class 658
EBW32Vector::AddElement 659
EBW32Vector::Draw 659
EBW32Vector::DrawWithCurrentPen 660
EBW32Vector::EBW32Vector 661
EBW32Vector::GetElement 661
EBW32Vector::operator[] 662
EBW32Vector::operator= 662
EBW32Vector::GetRawDataPtr 662
EBW32Vector::SetElement 663
EBW32Vector::WeightedMoment 663
4.36. EBW8PathVector Class 663
EBW8PathVector::AddElement 664
EBW8PathVector::GetClosed 665
EBW8PathVector::SetClosed 665
EBW8PathVector::Draw 665
EBW8PathVector::DrawClosedContour 667
EBW8PathVector::DrawWithCurrentPen 667
EBW8PathVector::EBW8PathVector 668
EBW8PathVector::GetElement 669
EBW8PathVector::operator[] 669
EBW8PathVector::operator= 669
EBW8PathVector::GetRawDataPtr 670
EBW8PathVector::SetElement 670
4.37. EBW8PixelAccessor Class 670
EBW8PixelAccessor::EBW8PixelAccessor 671
EBW8PixelAccessor::GetPixel 671
EBW8PixelAccessor::SetPixel 671
4.38. EBW8Vector Class 672
EBW8Vector::AddElement 673
EBW8Vector::Draw 673
EBW8Vector::DrawWithCurrentPen 674
EBW8Vector::EBW8Vector 675
EBW8Vector::GetElement 675
EBW8Vector::operator[] 676
EBW8Vector::operator= 676
EBW8Vector::GetRawDataPtr 676
EBW8Vector::SetElement 677
EBW8Vector::WeightedMoment 677
4.39. EBWHistogramVector Class 677
EBWHistogramVector::AddElement 678
EBWHistogramVector::Draw 678
EBWHistogramVector::DrawWithCurrentPen 680
EBWHistogramVector::EBWHistogramVector 680
EBWHistogramVector::GetElement 681
EBWHistogramVector::operator[] 681
EBWHistogramVector::operator= 682
EBWHistogramVector::GetRawDataPtr 682
EBWHistogramVector::SetElement 682
4.40. EC15PixelAccessor Class 683
EC15PixelAccessor::EC15PixelAccessor 683
EC15PixelAccessor::GetPixel 683

EC15PixelAccessor::SetPixel 684
4.41. EC16PixelAccessor Class 684
EC16PixelAccessor::EC16PixelAccessor 684
EC16PixelAccessor::GetPixel 685
EC16PixelAccessor::SetPixel 685
4.42. EC24APixelAccessor Class 686
EC24APixelAccessor::EC24APixelAccessor 686
EC24APixelAccessor::GetPixel 686
EC24APixelAccessor::SetPixel 687
4.43. EC24PathVector Class 687
EC24PathVector::AddElement 688
EC24PathVector::GetClosed 688
EC24PathVector::SetClosed 688
EC24PathVector::Draw 689
EC24PathVector::DrawClosedContour 690
EC24PathVector::DrawWithCurrentPen 691
EC24PathVector::EC24PathVector 692
EC24PathVector::GetElement 692
EC24PathVector::operator[] 693
EC24PathVector::operator= 693
EC24PathVector::GetRawDataPtr 693
EC24PathVector::SetElement 694
4.44. EC24PixelAccessor Class 694
EC24PixelAccessor::EC24PixelAccessor 694
EC24PixelAccessor::GetPixel 695
EC24PixelAccessor::SetPixel 695
4.45. EC24Vector Class 696
EC24Vector::AddElement 696
EC24Vector::Draw 697
EC24Vector::EC24Vector 699
EC24Vector::GetElement 699
EC24Vector::operator[] 700
EC24Vector::operator= 700
EC24Vector::GetRawDataPtr 700
EC24Vector::SetElement 701
4.46. ECalibrationGenerator Class 701
4.47. ECalibrationModel Class 701
ECalibrationModel::Apply 702
ECalibrationModel::Create 702
ECalibrationModel::Save 703
ECalibrationModel::GetType 703
4.48. ECannyEdgeDetector Class 703
ECannyEdgeDetector::Apply 704
ECannyEdgeDetector::ECannyEdgeDetector 705
ECannyEdgeDetector::GetHighThreshold 705
ECannyEdgeDetector::SetHighThreshold 705
ECannyEdgeDetector::GetLowThreshold 705
ECannyEdgeDetector::SetLowThreshold 705
ECannyEdgeDetector::ResetSmoothingScale 705
ECannyEdgeDetector::GetSmoothingScale 706
ECannyEdgeDetector::SetSmoothingScale 706
ECannyEdgeDetector::GetThresholdingMode 706
ECannyEdgeDetector::SetThresholdingMode 706
4.49. EChecker Class 707
EChecker::AddPathName 708
EChecker::Attach 709
EChecker::GetAverage 709
EChecker::BatchLearn 709

EChecker::GetDarkGray ... 710
EChecker::SetDarkGray ... 710
EChecker::GetDegreesOfFreedom ... 710
EChecker::SetDegreesOfFreedom ... 710
EChecker::GetDeviation ... 711
EChecker::Drag ... 711
EChecker::Draw ... 711
EChecker::DrawWithCurrentPen ... 713
EChecker::EChecker ... 714
EChecker::EmptyPathNames ... 714
EChecker::GetHigh ... 714
EChecker::GetHitHandle ... 714
EChecker::GetHitRoi ... 715
EChecker::HitTest ... 715
EChecker::Learn ... 715
EChecker::GetLightGray ... 716
EChecker::SetLightGray ... 716
EChecker::Load ... 716
EChecker::GetLow ... 717
EChecker::GetNormalize ... 717
EChecker::SetNormalize ... 717
EChecker::GetNumAverageSamples ... 717
EChecker::GetNumDeviationSamples ... 718
EChecker::GetPanX ... 718
EChecker::GetPanY ... 718
EChecker::Register ... 718
EChecker::GetRegistered ... 719
EChecker::GetRelativeTolerance ... 719
EChecker::SetRelativeTolerance ... 719
EChecker::Save ... 719
EChecker::SetPan ... 720
EChecker::SetTolerance ... 720
EChecker::SetZoom ... 721
EChecker::GetToleranceX ... 721
EChecker::GetToleranceY ... 722
EChecker::GetZoomX ... 722
EChecker::GetZoomY ... 722
4.50. EChecker2 Class ... 722
EChecker2::AddTrainingImageFile ... 724
EChecker2::ClearTrainingImageFiles ... 725
EChecker2::DrawInspectionField ... 725
EChecker2::DrawReferenceInspectionField ... 726
EChecker2::DrawReferenceSearchFields ... 727
EChecker2::EChecker2 ... 728
EChecker2::GetFiducialHorizontalTolerance ... 729
EChecker2::SetFiducialHorizontalTolerance ... 729
EChecker2::GetFiducialMatchingMode ... 729
EChecker2::SetFiducialMatchingMode ... 729
EChecker2::GetFiducialVerticalTolerance ... 729
EChecker2::SetFiducialVerticalTolerance ... 729
EChecker2::GetHighThresholdImage ... 730
EChecker2::Initialize ... 730
EChecker2::Inspect ... 730
EChecker2::GetInspectionTolerance ... 731
EChecker2::SetInspectionTolerance ... 731
EChecker2::GetIsInitialized ... 731
EChecker2::GetIsTrained ... 731
EChecker2::GetLastRegisteredImage ... 731
EChecker2::Load ... 732

EChecker2::GetLowThresholdImage 732
EChecker2::GetNormalizationMode 732
EChecker2::SetNormalizationMode 732
EChecker2::ResetTraining 733
EChecker2::Save 733
EChecker2::Train 733
EChecker2::TrainFromImageFiles 734
EChecker2::GetTrainingMode 734
EChecker2::SetTrainingMode 734
4.51. ECircle Class 734
ECircle::GetAmplitude 736
ECircle::SetAmplitude 736
ECircle::GetApex 736
ECircle::GetApexAngle 736
ECircle::GetArcLength 737
ECircle::CopyTo 737
ECircle::GetDiameter 738
ECircle::SetDiameter 738
ECircle::GetDirect 738
ECircle::Distance 738
ECircle::ECircle 739
ECircle::GetEnd 740
ECircle::GetEndAngle 740
ECircle::GetFull 741
ECircle::GetDistanceBetweenLineAndCircle 741
ECircle::GetDistanceBetweenPointAndCircle 742
ECircle::GetIntersectionOfCircles 742
ECircle::GetIntersectionOfLineAndCircle 743
ECircle::GetPoint 744
ECircle::GetProjectionOfPointOnCircle 744
ECircle::operator= 744
ECircle::GetOrg 745
ECircle::GetOrgAngle 745
ECircle::GetRadius 745
ECircle::SetRadius 745
ECircle::SetFromCenterAndOrigin 746
ECircle::SetFromOriginMiddleEnd 746
4.52. ECircleGauge Class 747
750
ECircleGauge::SetActive 750
ECircleGauge::AddSkipRange 750
ECircleGauge::GetAverageDistance 751
751
ECircleGauge::SetCircle 751
ECircleGauge::CopyTo 751
ECircleGauge::DisableInnerFiltering 752
ECircleGauge::Drag 752
ECircleGauge::Draw 753
ECircleGauge::DrawWithCurrentPen 754
ECircleGauge::ECircleGauge 754
ECircleGauge::GetFilteringThreshold 755
ECircleGauge::SetFilteringThreshold 755
ECircleGauge::GetMeasuredPeak 755
ECircleGauge::GetMeasuredPoint 756
ECircleGauge::GetMinNumFitSamples 756
ECircleGauge::GetSample 757
ECircleGauge::GetSkipRange 758
ECircleGauge::HitTest 758
ECircleGauge::GetHVConstraint 759

ECircleGauge::SetHVConstraint 759
ECircleGauge::GetInnerFilteringEnabled 759
ECircleGauge::GetInnerFilteringThreshold 759
ECircleGauge::SetInnerFilteringThreshold 759
ECircleGauge::Measure 760
ECircleGauge::GetMeasuredCircle 760
ECircleGauge::MeasureSample 761
ECircleGauge::MeasureWithoutFitting 761
ECircleGauge::GetMinAmplitude 762
ECircleGauge::SetMinAmplitude 762
ECircleGauge::GetMinArea 762
ECircleGauge::SetMinArea 762
ECircleGauge::GetNumFilteringPasses 763
ECircleGauge::SetNumFilteringPasses 763
ECircleGauge::GetNumMeasuredPoints 763
ECircleGauge::GetNumSamples 763
ECircleGauge::GetNumSkipRanges 764
ECircleGauge::GetNumValidSamples 764
ECircleGauge::operator= 764
ECircleGauge::Plot 765
ECircleGauge::PlotWithCurrentPen 766
ECircleGauge::Process 767
ECircleGauge::GetRectangularSamplingArea 768
ECircleGauge::SetRectangularSamplingArea 768
ECircleGauge::RemoveAllSkipRanges 768
ECircleGauge::RemoveSkipRange 768
ECircleGauge::GetSamplingStep 769
ECircleGauge::SetSamplingStep 769
ECircleGauge::SetMinNumFitSamples 769
ECircleGauge::GetSmoothing 770
ECircleGauge::SetSmoothing 770
ECircleGauge::GetThickness 770
ECircleGauge::SetThickness 770
ECircleGauge::GetThreshold 771
ECircleGauge::SetThreshold 771
ECircleGauge::GetTolerance 771
ECircleGauge::SetTolerance 771
ECircleGauge::GetTransitionChoice 772
ECircleGauge::SetTransitionChoice 772
ECircleGauge::GetTransitionIndex 772
ECircleGauge::SetTransitionIndex 772
ECircleGauge::GetTransitionType 772
ECircleGauge::SetTransitionType 772
ECircleGauge::GetType 773
ECircleGauge::GetValid 773
4.53. ECircleRegion Class 773
ECircleRegion::GetCenter 774
ECircleRegion::SetCenter 774
ECircleRegion::Drag 774
ECircleRegion::ECircleRegion 775
ECircleRegion::HitTest 776
ECircleRegion::Load 777
ECircleRegion::operator!= 778
ECircleRegion::operator= 778
ECircleRegion::operator== 778
ECircleRegion::GetRadius 779
ECircleRegion::SetRadius 779
ECircleRegion::Save 779
ECircleRegion::Scale 779

ECircleRegion::Translate 780
4.54. ECircleShape Class 780
ECircleShape::GetAmplitude 782
ECircleShape::SetAmplitude 782
ECircleShape::GetAngle 782
ECircleShape::SetAngle 782
ECircleShape::GetApex 783
ECircleShape::GetApexAngle 783
ECircleShape::GetArcLength 783
ECircleShape::GetCenter 784
ECircleShape::SetCenter 784
ECircleShape::GetCenterX 784
ECircleShape::GetCenterY 784
ECircleShape::GetCircle 784
ECircleShape::SetCircle 784
ECircleShape::Closest 785
ECircleShape::CopyTo 785
ECircleShape::GetDiameter 786
ECircleShape::SetDiameter 786
ECircleShape::GetDirect 786
ECircleShape::Drag 786
ECircleShape::Draw 787
ECircleShape::DrawWithCurrentPen 788
ECircleShape::ECircleShape 788
ECircleShape::GetEnd 789
ECircleShape::GetEndAngle 790
ECircleShape::GetFull 790
ECircleShape::GetPoint 790
ECircleShape::HitTest 791
ECircleShape::operator= 791
ECircleShape::GetOrg 791
ECircleShape::GetOrgAngle 792
ECircleShape::GetRadius 792
ECircleShape::SetRadius 792
ECircleShape::GetScale 792
ECircleShape::SetScale 792
ECircleShape::SetCenterXY 793
ECircleShape::SetFromCenterAndOrigin 793
ECircleShape::SetFromOriginMiddleEnd 794
ECircleShape::GetType 794
4.55. EClassificationDataset Class 795
EClassificationDataset::GetAbsoluteMaxOverlap 805
EClassificationDataset::AddImage 805
EClassificationDataset::AddImageObject 807
EClassificationDataset::AddImages 808
EClassificationDataset::AddLabel 809
EClassificationDataset::AddObjectLabel 810
EClassificationDataset::AddRegionToSegment 810
EClassificationDataset::AddSegmentationLabel 811
811
EClassificationDataset::SetBasePath 811
EClassificationDataset::GetChannels 811
EClassificationDataset::Clear 811
EClassificationDataset::EClassificationDataset 812
EClassificationDataset::GetEnableDataAugmentation 812
EClassificationDataset::SetEnableDataAugmentation 812
EClassificationDataset::GetEnableHorizontalFlip 812
EClassificationDataset::SetEnableHorizontalFlip 812
EClassificationDataset::GetEnableVerticalFlip 813

EClassificationDataset::SetEnableVerticalFlip ... 813
EClassificationDataset::Export ... 813
EClassificationDataset::ExtractSplit ... 814
EClassificationDataset::GetGaussianNoiseMaximumStandardDeviation ... 815
EClassificationDataset::SetGaussianNoiseMaximumStandardDeviation ... 815
EClassificationDataset::GetGaussianNoiseMinimumStandardDeviation ... 815
EClassificationDataset::SetGaussianNoiseMinimumStandardDeviation ... 815
EClassificationDataset::GetAvailableImageAnnotationFormat ... 816
EClassificationDataset::GetImageCopy ... 816
EClassificationDataset::GetImageCopyWithDataAugmentation ... 817
EClassificationDataset::GetImageLabel ... 817
EClassificationDataset::GetImageNumObjects ... 818
EClassificationDataset::GetImageObject ... 818
EClassificationDataset::GetImageObjects ... 818
EClassificationDataset::GetImagePath ... 819
EClassificationDataset::GetImages ... 819
EClassificationDataset::GetImagesIndexesWithLabel ... 820
EClassificationDataset::GetLabel ... 820
EClassificationDataset::GetLabelWeight ... 820
EClassificationDataset::GetMask ... 821
EClassificationDataset::GetNumImagesForFile ... 821
EClassificationDataset::GetNumImagesForSegmentationLabel ... 821
EClassificationDataset::GetNumImagesWithObjectLabel ... 822
EClassificationDataset::GetNumObjectsWithLabel ... 822
EClassificationDataset::GetNumPixelsForSegmentationLabel ... 823
EClassificationDataset::GetNumSegmentedBlobs ... 823
EClassificationDataset::GetObjectLabel ... 824
EClassificationDataset::GetObjectLabelWeight ... 824
EClassificationDataset::GetRegionForSegment ... 825
EClassificationDataset::GetRegionOfInterestHeight ... 825
EClassificationDataset::GetRegionOfInterestOriginX ... 826
EClassificationDataset::GetRegionOfInterestOriginY ... 826
EClassificationDataset::GetRegionOfInterestWidth ... 826
EClassificationDataset::GetSegmentationLabel ... 827
EClassificationDataset::GetSegmentationLabelWeight ... 827
EClassificationDataset::GetSegmentationMap ... 827
EClassificationDataset::GetSplit ... 828
EClassificationDataset::HasForegroundSegments ... 829
EClassificationDataset::HasLabel ... 829
EClassificationDataset::HasObjectLabeling ... 830
EClassificationDataset::HasSegmentation ... 830
EClassificationDataset::GetHeight ... 830
EClassificationDataset::GetImagesIndexesWithNoLabel ... 830
EClassificationDataset::ImportPascalVOCXMLAnnotations ... 831
EClassificationDataset::ImportYOLOTXTAnnotations ... 831
EClassificationDataset::IsEmbeddedImage ... 832
EClassificationDataset::IsImageFile ... 832
EClassificationDataset::GetLabeledImagesIndexes ... 832
EClassificationDataset::Load ... 833
EClassificationDataset::GetMaxBrightnessOffset ... 833
EClassificationDataset::SetMaxBrightnessOffset ... 833
EClassificationDataset::GetMaxContrastGain ... 833
EClassificationDataset::SetMaxContrastGain ... 833
EClassificationDataset::GetMaxGamma ... 834
EClassificationDataset::SetMaxGamma ... 834
EClassificationDataset::GetMaxHorizontalShear ... 834
EClassificationDataset::SetMaxHorizontalShear ... 834
EClassificationDataset::GetMaxHorizontalShift ... 835
EClassificationDataset::SetMaxHorizontalShift ... 835

EClassificationDataset::GetMaxHueOffset ... 835
EClassificationDataset::SetMaxHueOffset ... 835
EClassificationDataset::GetMaxNumObjectPerImage ... 835
EClassificationDataset::GetMaxRotationAngle ... 836
EClassificationDataset::SetMaxRotationAngle ... 836
EClassificationDataset::GetMaxSaturationGain ... 836
EClassificationDataset::SetMaxSaturationGain ... 836
EClassificationDataset::GetMaxScale ... 836
EClassificationDataset::SetMaxScale ... 836
EClassificationDataset::GetMaxVerticalShear ... 837
EClassificationDataset::SetMaxVerticalShear ... 837
EClassificationDataset::GetMaxVerticalShift ... 837
EClassificationDataset::SetMaxVerticalShift ... 837
EClassificationDataset::GetMinContrastGain ... 837
EClassificationDataset::SetMinContrastGain ... 837
EClassificationDataset::GetMinGamma ... 838
EClassificationDataset::SetMinGamma ... 838
EClassificationDataset::GetMinSaturationGain ... 838
EClassificationDataset::SetMinSaturationGain ... 838
EClassificationDataset::GetMinScale ... 839
EClassificationDataset::SetMinScale ... 839
EClassificationDataset::GetNumImageFiles ... 839
EClassificationDataset::GetNumImages ... 839
EClassificationDataset::GetNumImagesWithForegroundSegments ... 839
EClassificationDataset::GetNumImagesWithObjectLabeling ... 840
EClassificationDataset::GetNumImagesWithObjects ... 840
EClassificationDataset::GetNumImagesWithoutForegroundSegments ... 840
EClassificationDataset::GetNumImagesWithoutObjectLabeling ... 840
EClassificationDataset::GetNumImagesWithoutObjects ... 840
EClassificationDataset::GetNumImagesWithoutSegmentation ... 841
EClassificationDataset::GetNumImagesWithSegmentation ... 841
EClassificationDataset::GetNumLabeledImages ... 841
EClassificationDataset::GetNumLabels ... 841
EClassificationDataset::GetNumObjectLabels ... 841
EClassificationDataset::GetNumSegmentationLabels ... 842
EClassificationDataset::GetNumUnlabeledImages ... 842
EClassificationDataset::GetObjectSize ... 842
EClassificationDataset::SetObjectSize ... 842
EClassificationDataset::operator= ... 842
EClassificationDataset::RemoveImage ... 843
EClassificationDataset::RemoveImageObject ... 843
EClassificationDataset::RemoveLabel ... 844
EClassificationDataset::RemoveObjectLabel ... 844
EClassificationDataset::RemoveSegmentationLabel ... 844
EClassificationDataset::ResetImageObjectLabeling ... 845
EClassificationDataset::ResetSegmentation ... 845
EClassificationDataset::GetSaltAndPepperNoiseMaximumDensity ... 846
EClassificationDataset::SetSaltAndPepperNoiseMaximumDensity ... 846
EClassificationDataset::GetSaltAndPepperNoiseMinimumDensity ... 846
EClassificationDataset::SetSaltAndPepperNoiseMinimumDensity ... 846
EClassificationDataset::GetSameLabelMaxOverlap ... 846
EClassificationDataset::Save ... 847
EClassificationDataset::SetImageLabel ... 847
EClassificationDataset::SetImageObject ... 848
EClassificationDataset::SetLabel ... 849
EClassificationDataset::SetLabelWeight ... 849
EClassificationDataset::SetMask ... 850
EClassificationDataset::SetObjectLabel ... 850
EClassificationDataset::SetObjectLabelWeight ... 851

EClassificationDataset::SetRegionOfInterest 851
EClassificationDataset::SetSegmentationLabel 852
EClassificationDataset::SetSegmentationLabelWeight 852
EClassificationDataset::SetSegmentationMap 853
EClassificationDataset::GetSpeckleNoiseMaximumStandardDeviation 853
EClassificationDataset::SetSpeckleNoiseMaximumStandardDeviation 853
EClassificationDataset::GetSpeckleNoiseMinimumStandardDeviation 854
EClassificationDataset::SetSpeckleNoiseMinimumStandardDeviation 854
EClassificationDataset::SplitDataset 854
EClassificationDataset::SplitDatasetForLocator 855
EClassificationDataset::SplitDatasetForSegmentation 855
EClassificationDataset::UnsetImageLabel 856
EClassificationDataset::UnsetImageObjectLabeling 856
EClassificationDataset::UnsetSegmentation 857
EClassificationDataset::GetWidth 857
4.56. EClassificationMetrics Class 857
EClassificationMetrics::GetAccuracy 858
EClassificationMetrics::AddMetrics 859
EClassificationMetrics::AddResult 859
EClassificationMetrics::GetBalancedAccuracy 859
EClassificationMetrics::GetBalancedError 860
EClassificationMetrics::CanComputeWeightedError 860
EClassificationMetrics::EClassificationMetrics 860
EClassificationMetrics::GetError 860
EClassificationMetrics::GetConfusion 861
EClassificationMetrics::GetLabelAccuracy 861
EClassificationMetrics::GetLabelError 862
EClassificationMetrics::GetWeightedAccuracy 862
EClassificationMetrics::GetWeightedError 863
EClassificationMetrics::IsValid 863
EClassificationMetrics::Load 863
EClassificationMetrics::operator= 864
EClassificationMetrics::Save 864
4.57. EClassificationResult Class 865
EClassificationResult::GetBestLabel 866
EClassificationResult::GetBestLabelId 866
EClassificationResult::GetBestProbability 866
EClassificationResult::DrawHeatmap 866
EClassificationResult::EClassificationResult 868
EClassificationResult::GetColorizedHeatmap 868
EClassificationResult::GetColorizedHeatmapWithTransparency 868
EClassificationResult::GetLabel 869
EClassificationResult::GetLabelColor 869
EClassificationResult::GetProbability 870
EClassificationResult::GetRanking 870
EClassificationResult::GetGroundtruthLabel 871
EClassificationResult::HasGroundtruth 871
EClassificationResult::HasHeatmap 871
EClassificationResult::GetHeatmap 871
EClassificationResult::IsValid 872
EClassificationResult::GetNumLabels 872
EClassificationResult::operator= 872
4.58. EClassifier Class 873
EClassifier::GetCapacity 875
EClassifier::SetCapacity 875
EClassifier::GetChannels 876
EClassifier::SetChannels 876
EClassifier::Classify 876
EClassifier::GetComputeHeatmapWithResult 877

EClassifier::SetComputeHeatmapWithResult 877
EClassifier::EClassifier 877
EClassifier::GetEnableAutomaticImageReformat 878
EClassifier::SetEnableAutomaticImageReformat 878
EClassifier::GetEnableHistogramEqualization 878
EClassifier::SetEnableHistogramEqualization 878
EClassifier::Evaluate 878
EClassifier::GetAvailableModelTypes 879
EClassifier::GetColorizedHeatMap 879
EClassifier::GetColorizedHeatmapWithTransparency 880
EClassifier::GetHeatMap 880
EClassifier::GetTrainingMetrics 881
EClassifier::GetValidationMetrics 881
EClassifier::HasPretrainedModel 881
EClassifier::GetHeight 882
EClassifier::SetHeight 882
EClassifier::LoadAsPretrained 882
EClassifier::GetMinimumHeight 882
EClassifier::GetMinimumWidth 883
EClassifier::GetModelType 883
EClassifier::SetModelType 883
EClassifier::GetNumAvailableModelTypes 883
EClassifier::operator= 883
EClassifier::SerializeSettings 884
EClassifier::GetToolType 884
EClassifier::GetUsePretrainedModel 884
EClassifier::SetUsePretrainedModel 884
EClassifier::GetWidth 885
EClassifier::SetWidth 885
4.59. ECode Class 885
ECode::GetBarCode 885
ECode::GetCodeType 886
ECode::GetDecodedString 886
ECode::DrawPosition 886
ECode::DrawPositionWithCurrentPen 887
ECode::ECode 888
ECode::GetMatrixCode 888
ECode::operator= 888
ECode::GetPosition 889
ECode::GetQRCode 889
4.60. ECodedElement Class 889
ECodedElement::GetArea 892
ECodedElement::AsHole 893
ECodedElement::AsObject 893
ECodedElement::GetBottomLimit 893
ECodedElement::GetBoundingBox 894
ECodedElement::GetBoundingBoxCenter 894
ECodedElement::GetBoundingBoxCenterX 894
ECodedElement::GetBoundingBoxCenterY 894
ECodedElement::GetBoundingBoxHeight 895
ECodedElement::GetBoundingBoxWidth 895
ECodedElement::ComputeConvexHull 895
ECodedElement::ComputeFeretBox 895
ECodedElement::ComputePixelGrayAverage 896
ECodedElement::ComputePixelGrayDeviation 897
ECodedElement::ComputePixelGrayVariance 897
ECodedElement::ComputePixelMax 897
ECodedElement::ComputePixelMin 898
ECodedElement::ComputeWeightedGravityCenter 898

ECodedElement::GetContour 898
ECodedElement::GetContourPath 899
ECodedElement::GetContourX 899
ECodedElement::GetContourY 899
ECodedElement::GetConvexHull 899
ECodedElement::GetEccentricity 899
ECodedElement::GetElementIndex 900
ECodedElement::GetEllipseAngle 900
ECodedElement::GetEllipseHeight 900
ECodedElement::GetEllipseWidth 901
ECodedElement::GetFeretBox22Box 901
ECodedElement::GetFeretBox22Center 901
ECodedElement::GetFeretBox22CenterX 901
ECodedElement::GetFeretBox22CenterY 901
ECodedElement::GetFeretBox22Height 902
ECodedElement::GetFeretBox22Width 902
ECodedElement::GetFeretBox45Box 902
ECodedElement::GetFeretBox45Center 902
ECodedElement::GetFeretBox45CenterX 902
ECodedElement::GetFeretBox45CenterY 903
ECodedElement::GetFeretBox45Height 903
ECodedElement::GetFeretBox45Width 903
ECodedElement::GetFeretBox68Box 903
ECodedElement::GetFeretBox68Center 904
ECodedElement::GetFeretBox68CenterX 904
ECodedElement::GetFeretBox68CenterY 904
ECodedElement::GetFeretBox68Height 904
ECodedElement::GetFeretBox68Width 904
ECodedElement::GetCentralMoment 905
ECodedElement::GetMoment 905
ECodedElement::GetNormalizedCentralMoment 906
ECodedElement::GetGravityCenter 906
ECodedElement::GetGravityCenterX 907
ECodedElement::GetGravityCenterY 907
ECodedElement::GetIsCodedElement 907
ECodedElement::GetIsHole 907
ECodedElement::GetIsObject 908
ECodedElement::GetLargestRun 908
ECodedElement::GetLayerIndex 908
ECodedElement::GetLeftLimit 908
ECodedElement::GetMinimumEnclosingRectangle 909
ECodedElement::GetMinimumEnclosingRectangleAngle 909
ECodedElement::GetMinimumEnclosingRectangleCenter 910
ECodedElement::GetMinimumEnclosingRectangleCenterX 910
ECodedElement::GetMinimumEnclosingRectangleCenterY 911
ECodedElement::GetMinimumEnclosingRectangleHeight 911
ECodedElement::GetMinimumEnclosingRectangleWidth 911
ECodedElement::operator== 911
ECodedElement::RenderMask 912
ECodedElement::GetRightLimit 912
ECodedElement::GetRunCount 912
ECodedElement::GetRunsIterator 913
ECodedElement::GetSigmaX 913
ECodedElement::GetSigmaXX 913
ECodedElement::GetSigmaXY 913
ECodedElement::GetSigmaY 913
ECodedElement::GetSigmaYY 914
ECodedElement::GetTopLimit 914
ECodedElement::ToRegion 914

4.61. ECodedImage Class 914
ECodedImage::AddFeat 919
ECodedImage::AnalyseObjects 919
ECodedImage::GetBlackClass 920
ECodedImage::SetBlackClass 920
ECodedImage::BlankFeatures 921
ECodedImage::BuildHoles 921
ECodedImage::BuildLabeledObjects 922
ECodedImage::BuildLabeledRuns 922
ECodedImage::BuildObjects 923
ECodedImage::BuildRuns 924
ECodedImage::GetConnexity 924
ECodedImage::SetConnexity 924
ECodedImage::GetContinuous 925
ECodedImage::SetContinuous 925
ECodedImage::GetCurrentObjPtr 925
ECodedImage::GetCurrentRunPtr 925
ECodedImage::GetDrawDiagonals 926
ECodedImage::SetDrawDiagonals 926
ECodedImage::DrawObject 926
ECodedImage::DrawObjectFeature 928
ECodedImage::DrawObjectFeatureWithCurrentPen 930
ECodedImage::DrawObjects 931
ECodedImage::DrawObjectsFeature 932
ECodedImage::DrawObjectsFeatureWithCurrentPen 934
ECodedImage::DrawObjectsWithCurrentPen 935
ECodedImage::DrawObjectWithCurrentPen 936
ECodedImage::ECodedImage 937
ECodedImage::FeatureAverage 937
ECodedImage::FeatureDeviation 937
ECodedImage::FeatureMaximum 938
ECodedImage::FeatureMinimum 938
ECodedImage::FeatureVariance 939
ECodedImage::GetFirstObjPtr 939
ECodedImage::GetCurrentObjData 940
ECodedImage::GetCurrentRunData 940
ECodedImage::GetFeatData 940
ECodedImage::GetFeatDataSize 941
ECodedImage::GetFeatDataType 941
ECodedImage::GetFeatNum 942
ECodedImage::GetFeatPtrByNum 942
ECodedImage::GetFeatSize 943
ECodedImage::GetFirstHole 943
ECodedImage::GetFirstObjData 944
ECodedImage::GetFirstRunData 944
ECodedImage::GetFirstRunPtr 944
ECodedImage::GetHoleParentObject 945
ECodedImage::GetLastObjData 945
ECodedImage::GetLastRunData 945
ECodedImage::GetLastRunPtr 946
ECodedImage::GetNextHole 946
ECodedImage::GetNextObjData 946
ECodedImage::GetNextObjPtr 947
ECodedImage::GetNextRunData 947
ECodedImage::GetNextRunPtr 948
ECodedImage::GetNumHoles 948
ECodedImage::GetNumObjectRuns 949
ECodedImage::GetObjDataPtr 949
ECodedImage::GetObjectData 949

ECodedImage::GetObjectFeature 950
ECodedImage::GetObjFirstRunPtr 952
ECodedImage::GetObjLastRunPtr 953
ECodedImage::GetObjPtr 953
ECodedImage::GetObjPtrByCoordinates 953
ECodedImage::GetObjPtrByPos 954
ECodedImage::GetPreviousObjData 954
ECodedImage::GetPreviousObjPtr 955
ECodedImage::GetPreviousRunData 955
ECodedImage::GetPreviousRunPtr 955
ECodedImage::GetRunData 956
ECodedImage::GetRunDataPtr 956
ECodedImage::GetRunPtr 957
ECodedImage::GetRunPtrByCoordinates 957
ECodedImage::GetHighColorThreshold 958
ECodedImage::SetHighColorThreshold 958
ECodedImage::GetHighImage 958
ECodedImage::SetHighImage 958
ECodedImage::GetHighThreshold 958
ECodedImage::SetHighThreshold 958
ECodedImage::IsHole 959
ECodedImage::IsObjectSelected 959
ECodedImage::GetLastObjPtr 960
ECodedImage::GetLimitAngle 960
ECodedImage::SetLimitAngle 960
ECodedImage::GetLowColorThreshold 960
ECodedImage::SetLowColorThreshold 960
ECodedImage::GetLowImage 961
ECodedImage::SetLowImage 961
ECodedImage::GetLowThreshold 961
ECodedImage::SetLowThreshold 961
ECodedImage::GetMaxObjects 962
ECodedImage::SetMaxObjects 962
ECodedImage::GetNeutralClass 962
ECodedImage::SetNeutralClass 962
ECodedImage::GetNumFeatures 962
ECodedImage::GetNumHoleRuns 963
ECodedImage::GetNumObjects 963
ECodedImage::GetNumRuns 963
ECodedImage::GetNumSelectedObjects 964
ECodedImage::SetNumSelectedObjects 964
ECodedImage::ObjectConvexHull 964
ECodedImage::RemoveAllFeats 964
ECodedImage::RemoveAllObjects 965
ECodedImage::RemoveAllRuns 965
ECodedImage::RemoveHoles 965
ECodedImage::RemoveObject 966
ECodedImage::RemoveRun 966
ECodedImage::ResetContinuousMode 967
ECodedImage::SelectAllObjects 967
ECodedImage::SelectHoles 967
ECodedImage::SelectObject 968
ECodedImage::SelectObjectsUsingFeature 968
ECodedImage::SelectObjectsUsingPosition 969
ECodedImage::SetFeatInfo 970
ECodedImage::SetFirstRunPtr 971
ECodedImage::SetLastRunPtr 971
ECodedImage::SortObjectsUsingFeature 972
ECodedImage::GetThreshold 972

ECodedImage::SetThreshold ... 972
973
ECodedImage::SetThresholdImage ... 973
ECodedImage::GetTrueThreshold ... 973
ECodedImage::UnselectAllObjects ... 973
ECodedImage::UnselectHoles ... 973
ECodedImage::UnselectObject ... 974
ECodedImage::GetWhiteClass ... 974
ECodedImage::SetWhiteClass ... 974
4.62. ECodedImage2 Class ... 975
ECodedImage2::ClearFeatureCache ... 976
ECodedImage2::Draw ... 976
ECodedImage2::DrawFeature ... 980
ECodedImage2::DrawFeatureWithCurrentPen ... 984
ECodedImage2::DrawHole ... 986
ECodedImage2::DrawHoleFeature ... 988
ECodedImage2::DrawHoleFeatureWithCurrentPen ... 991
ECodedImage2::DrawHoleWithCurrentPen ... 992
ECodedImage2::DrawObject ... 994
ECodedImage2::DrawObjectFeature ... 996
ECodedImage2::DrawObjectFeatureWithCurrentPen ... 998
ECodedImage2::DrawObjectWithCurrentPen ... 999
ECodedImage2::DrawWithCurrentPen ... 1000
ECodedImage2::ECodedImage2 ... 1002
ECodedImage2::FindObject ... 1002
ECodedImage2::GetObj ... 1003
ECodedImage2::GetObjCount ... 1004
ECodedImage2::GetParentObject ... 1004
ECodedImage2::GetHeight ... 1005
ECodedImage2::GetLayerCount ... 1005
ECodedImage2::RenderMask ... 1005
ECodedImage2::GetStartY ... 1006
ECodedImage2::ToRegion ... 1006
ECodedImage2::GetWidth ... 1007
4.63. ECodeGrid Class ... 1007
ECodeGrid::ECodeGrid ... 1007
1008
ECodeGrid::SetEnableAll ... 1008
ECodeGrid::GetCellEnabled ... 1008
ECodeGrid::GetResults ... 1009
ECodeGrid::GetNumCols ... 1009
ECodeGrid::GetNumRows ... 1009
ECodeGrid::operator= ... 1010
ECodeGrid::SetEnableCell ... 1010
ECodeGrid::SetEnableColumn ... 1011
ECodeGrid::SetEnableRow ... 1011
4.64. ECodeReader Class ... 1011
ECodeReader::GetBarCodeReader ... 1012
ECodeReader::ECodeReader ... 1012
ECodeReader::GetEnabledCodeTypes ... 1013
ECodeReader::SetEnabledCodeTypes ... 1013
ECodeReader::Load ... 1013
ECodeReader::GetMatrixCodeReader ... 1014
ECodeReader::GetMaxNumCodesPerType ... 1014
ECodeReader::SetMaxNumCodesPerType ... 1014
ECodeReader::operator= ... 1014
ECodeReader::GetQRCodeReader ... 1014
ECodeReader::Read ... 1015
ECodeReader::Save ... 1016

ECodeReader::GetTimeOut 1016
ECodeReader::SetTimeOut 1016
4.65. EColorLookup Class 1017
EColorLookup::AdjustGainOffset 1017
EColorLookup::Calibrate 1018
EColorLookup::GetColorSystemIn 1020
EColorLookup::GetColorSystemOut 1021
EColorLookup::ConvertFromRgb 1021
EColorLookup::ConvertToRgb 1022
EColorLookup::EColorLookup 1022
EColorLookup::GetIndexBits 1022
EColorLookup::SetIndexBits 1022
EColorLookup::GetInterpolation 1023
EColorLookup::SetInterpolation 1023
EColorLookup::Transform 1024
EColorLookup::WhiteBalance 1024
4.66. EColorRangeThresholdSegmenter Class 1025
EColorRangeThresholdSegmenter::GetHighThreshold 1026
EColorRangeThresholdSegmenter::SetHighThreshold 1026
EColorRangeThresholdSegmenter::Load 1026
EColorRangeThresholdSegmenter::GetLowThreshold 1027
EColorRangeThresholdSegmenter::SetLowThreshold 1027
EColorRangeThresholdSegmenter::operator== 1027
EColorRangeThresholdSegmenter::Save 1027
4.67. EColorSingleThresholdSegmenter Class 1028
EColorSingleThresholdSegmenter::Load 1028
EColorSingleThresholdSegmenter::operator== 1029
EColorSingleThresholdSegmenter::Save 1029
EColorSingleThresholdSegmenter::GetThreshold 1030
EColorSingleThresholdSegmenter::SetThreshold 1030
4.68. EColorVector Class 1030
EColorVector::AddElement 1030
EColorVector::EColorVector 1031
EColorVector::GetElement 1031
EColorVector::operator[] 1032
EColorVector::operator= 1032
EColorVector::GetRawDataPtr 1032
EColorVector::SetElement 1033
4.69. EConverter Class 1033
EConverter::Convert 1033
4.70. EDataAugmentation Class 1035
EDataAugmentation::CopyTo 1040
EDataAugmentation::EDataAugmentation 1040
EDataAugmentation::GetEnableHorizontalFlip 1041
EDataAugmentation::SetEnableHorizontalFlip 1041
EDataAugmentation::GetEnableVerticalFlip 1041
EDataAugmentation::SetEnableVerticalFlip 1041
EDataAugmentation::GetGaussianNoiseMaximumStandardDeviation 1041
EDataAugmentation::SetGaussianNoiseMaximumStandardDeviation 1041
EDataAugmentation::GetGaussianNoiseMinimumStandardDeviation 1042
EDataAugmentation::SetGaussianNoiseMinimumStandardDeviation 1042
EDataAugmentation::Generate 1042
EDataAugmentation::HasDataAugmentations 1043
EDataAugmentation::Load 1043
EDataAugmentation::GetMaxBrightnessOffset 1044
EDataAugmentation::SetMaxBrightnessOffset 1044
EDataAugmentation::GetMaxContrastGain 1044
EDataAugmentation::SetMaxContrastGain 1044

EDataAugmentation::GetMaxGamma 1045
EDataAugmentation::SetMaxGamma 1045
EDataAugmentation::GetMaxHorizontalShear 1045
EDataAugmentation::SetMaxHorizontalShear 1045
EDataAugmentation::GetMaxHorizontalShift 1045
EDataAugmentation::SetMaxHorizontalShift 1045
EDataAugmentation::GetMaxHueOffset 1046
EDataAugmentation::SetMaxHueOffset 1046
EDataAugmentation::GetMaxRotationAngle 1046
EDataAugmentation::SetMaxRotationAngle 1046
EDataAugmentation::GetMaxSaturationGain 1046
EDataAugmentation::SetMaxSaturationGain 1046
EDataAugmentation::GetMaxScale 1047
EDataAugmentation::SetMaxScale 1047
EDataAugmentation::GetMaxStainColor 1047
EDataAugmentation::SetMaxStainColor 1047
EDataAugmentation::GetMaxVerticalShear 1047
EDataAugmentation::SetMaxVerticalShear 1047
EDataAugmentation::GetMaxVerticalShift 1048
EDataAugmentation::SetMaxVerticalShift 1048
EDataAugmentation::GetMinContrastGain 1048
EDataAugmentation::SetMinContrastGain 1048
EDataAugmentation::GetMinGamma 1048
EDataAugmentation::SetMinGamma 1048
EDataAugmentation::GetMinSaturationGain 1049
EDataAugmentation::SetMinSaturationGain 1049
EDataAugmentation::GetMinScale 1049
EDataAugmentation::SetMinScale 1049
EDataAugmentation::GetMinStainColor 1050
EDataAugmentation::SetMinStainColor 1050
EDataAugmentation::operator= 1050
EDataAugmentation::operator== 1050
EDataAugmentation::GetSaltAndPepperNoiseMaximumDensity 1051
EDataAugmentation::SetSaltAndPepperNoiseMaximumDensity 1051
EDataAugmentation::GetSaltAndPepperNoiseMinimumDensity 1051
EDataAugmentation::SetSaltAndPepperNoiseMinimumDensity 1051
EDataAugmentation::Save 1051
EDataAugmentation::GetSpeckleNoiseMaximumStandardDeviation 1052
EDataAugmentation::SetSpeckleNoiseMaximumStandardDeviation 1052
EDataAugmentation::GetSpeckleNoiseMinimumStandardDeviation 1052
EDataAugmentation::SetSpeckleNoiseMinimumStandardDeviation 1052
EDataAugmentation::GetStainBlur 1053
EDataAugmentation::SetStainBlur 1053
EDataAugmentation::GetStainColorVariation 1053
EDataAugmentation::SetStainColorVariation 1053
EDataAugmentation::GetStainDisruptionMaxAnchorPoints 1053
EDataAugmentation::SetStainDisruptionMaxAnchorPoints 1053
EDataAugmentation::GetStainDisruptionMaxIntensity 1054
EDataAugmentation::SetStainDisruptionMaxIntensity 1054
EDataAugmentation::GetStainEllipseMaxRadius 1054
EDataAugmentation::SetStainEllipseMaxRadius 1054
EDataAugmentation::GetStainEllipseMinRadius 1054
EDataAugmentation::SetStainEllipseMinRadius 1054
EDataAugmentation::GetStainInterpolationSiteFactor 1055
EDataAugmentation::SetStainInterpolationSiteFactor 1055
EDataAugmentation::GetStainProbability 1055
EDataAugmentation::SetStainProbability 1055
4.71. EDatasetSplit Class 1055
EDatasetSplit::CopyTo 1056

EDatasetSplit::GetDatasetMapping 1056
EDatasetSplit::EDatasetSplit 1057
EDatasetSplit::GetImageType 1057
EDatasetSplit::GetImageTypeLocked 1058
EDatasetSplit::GetNumImages 1058
EDatasetSplit::GetSplit 1058
EDatasetSplit::HasLockedImages 1059
EDatasetSplit::Load 1059
EDatasetSplit::GetNumLockedImages 1059
EDatasetSplit::operator= 1060
EDatasetSplit::Save 1060
EDatasetSplit::SetImageType 1060
EDatasetSplit::SetImageTypeLocked 1061
EDatasetSplit::SetTypeLocked 1061
4.72. EDecimator Class 1062
EDecimator::Decimate 1062
EDecimator::EDecimator 1062
EDecimator::Load 1063
EDecimator::operator!= 1063
EDecimator::operator== 1063
EDecimator::Save 1064
4.73. EDeepLearningBenchmark Class 1064
EDeepLearningBenchmark::GetBenchmarkHeight 1066
EDeepLearningBenchmark::GetBenchmarkSettings 1066
EDeepLearningBenchmark::GetBenchmarkWidth 1066
EDeepLearningBenchmark::GetDatasetHeight 1067
EDeepLearningBenchmark::GetDatasetWidth 1067
EDeepLearningBenchmark::EDeepLearningBenchmark 1067
EDeepLearningBenchmark::GetErrorCode 1067
EDeepLearningBenchmark::GetErrorDescription 1068
EDeepLearningBenchmark::IsValid 1068
EDeepLearningBenchmark::Load 1068
EDeepLearningBenchmark::GetMaxLatency 1069
EDeepLearningBenchmark::GetMaxTimePerImage 1069
EDeepLearningBenchmark::GetMaxTimePerInference 1069
EDeepLearningBenchmark::GetMeanLatency 1069
EDeepLearningBenchmark::GetMeanTimePerImage 1069
EDeepLearningBenchmark::GetMeanTimePerInference 1070
EDeepLearningBenchmark::GetMinLatency 1070
EDeepLearningBenchmark::GetMinTimePerImage 1070
EDeepLearningBenchmark::GetMinTimePerInference 1070
EDeepLearningBenchmark::GetNumDroppedInference 1071
EDeepLearningBenchmark::GetNumImagesByInference 1071
EDeepLearningBenchmark::GetNumInferences 1071
EDeepLearningBenchmark::operator= 1071
EDeepLearningBenchmark::Save 1072
EDeepLearningBenchmark::GetStdTimePerImage 1072
EDeepLearningBenchmark::GetStdTimePerInference 1072
EDeepLearningBenchmark::GetThroughput 1073
EDeepLearningBenchmark::GetTimePerInference 1073
4.74. EDeepLearningBenchmarkSettings Class 1073
EDeepLearningBenchmarkSettings::GetBatchSize 1074
EDeepLearningBenchmarkSettings::SetBatchSize 1074
EDeepLearningBenchmarkSettings::GetCameraSpeed 1075
EDeepLearningBenchmarkSettings::SetCameraSpeed 1075
EDeepLearningBenchmarkSettings::GetDevice 1075
EDeepLearningBenchmarkSettings::SetDevice 1075
EDeepLearningBenchmarkSettings::EDeepLearningBenchmarkSettings 1075
EDeepLearningBenchmarkSettings::GetEngine 1076

EDeepLearningBenchmarkSettings::SetEngine ... 1076
EDeepLearningBenchmarkSettings::GetInferencePrecision ... 1076
EDeepLearningBenchmarkSettings::SetInferencePrecision ... 1076
EDeepLearningBenchmarkSettings::GetInternalResizeDisabled ... 1076
EDeepLearningBenchmarkSettings::SetInternalResizeDisabled ... 1076
EDeepLearningBenchmarkSettings::Load ... 1077
EDeepLearningBenchmarkSettings::GetName ... 1077
EDeepLearningBenchmarkSettings::SetName ... 1077
EDeepLearningBenchmarkSettings::GetNumExternalThreads ... 1077
EDeepLearningBenchmarkSettings::SetNumExternalThreads ... 1077
EDeepLearningBenchmarkSettings::GetNumImages ... 1078
EDeepLearningBenchmarkSettings::SetNumImages ... 1078
EDeepLearningBenchmarkSettings::GetNumInternalThreads ... 1078
EDeepLearningBenchmarkSettings::SetNumInternalThreads ... 1078
EDeepLearningBenchmarkSettings::operator= ... 1078
EDeepLearningBenchmarkSettings::Save ... 1079
4.75. EDeepLearningDefectDetectionMetrics Class ... 1080
EDeepLearningDefectDetectionMetrics::GetAreaUnderROCCurve ... 1083
EDeepLearningDefectDetectionMetrics::GetAveragePrecision ... 1084
EDeepLearningDefectDetectionMetrics::GetBestAccuracy ... 1084
EDeepLearningDefectDetectionMetrics::GetBestAccuracyClassificationThreshold ... 1084
EDeepLearningDefectDetectionMetrics::GetBestBalancedAccuracy ... 1084
EDeepLearningDefectDetectionMetrics::GetBestBalancedAccuracyClassificationThreshold ... 1085
EDeepLearningDefectDetectionMetrics::GetBestFScore ... 1085
EDeepLearningDefectDetectionMetrics::GetBestFScoreThreshold ... 1085
EDeepLearningDefectDetectionMetrics::GetClassificationThreshold ... 1086
EDeepLearningDefectDetectionMetrics::SetClassificationThreshold ... 1086
EDeepLearningDefectDetectionMetrics::EDeepLearningDefectDetectionMetrics ... 1086
EDeepLearningDefectDetectionMetrics::GetAccuracy ... 1087
EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracy ... 1087
EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracyClassificationThreshold ... 1088
EDeepLearningDefectDetectionMetrics::GetConfusion ... 1088
EDeepLearningDefectDetectionMetrics::GetFScore ... 1089
EDeepLearningDefectDetectionMetrics::GetPrecision ... 1089
EDeepLearningDefectDetectionMetrics::GetPrecisionRecallCurvePoint ... 1089
EDeepLearningDefectDetectionMetrics::GetRecall ... 1090
EDeepLearningDefectDetectionMetrics::GetROCPoint ... 1090
EDeepLearningDefectDetectionMetrics::IsDefectDetectionMetricsValid ... 1091
EDeepLearningDefectDetectionMetrics::Load ... 1091
EDeepLearningDefectDetectionMetrics::GetNumberOfClassifiers ... 1092
EDeepLearningDefectDetectionMetrics::GetNumDefectiveSample ... 1092
EDeepLearningDefectDetectionMetrics::GetNumGoodSample ... 1092
EDeepLearningDefectDetectionMetrics::GetNumPrecisionRecallCurvePoint ... 1092
EDeepLearningDefectDetectionMetrics::operator= ... 1093
EDeepLearningDefectDetectionMetrics::GetPrecisionRecallCurveIndex ... 1093
EDeepLearningDefectDetectionMetrics::Save ... 1093
4.76. EDeepLearningDevice Class ... 1094
EDeepLearningDevice::GetDefaultPrecision ... 1095
EDeepLearningDevice::GetDeviceId ... 1095
EDeepLearningDevice::GetDeviceType ... 1095
EDeepLearningDevice::GetDeviceTypeDescription ... 1095
EDeepLearningDevice::EDeepLearningDevice ... 1096
EDeepLearningDevice::GetEngineName ... 1096
EDeepLearningDevice::GetFullName ... 1096
EDeepLearningDevice::HasInferenceCapability ... 1096
EDeepLearningDevice::HasTrainingCapability ... 1097
EDeepLearningDevice::IsPrecisionSupported ... 1097
EDeepLearningDevice::IsValid ... 1097
EDeepLearningDevice::GetName ... 1097

EDeepLearningDevice::operator!= 1098
EDeepLearningDevice::operator= 1098
EDeepLearningDevice::operator== 1098
EDeepLearningDevice::GetSupportedPrecisions 1099
4.77. EDeepLearningExecutionSettings Class 1099
EDeepLearningExecutionSettings::GetAvailableDevices 1101
EDeepLearningExecutionSettings::GetAvailableEngines 1101
EDeepLearningExecutionSettings::GetBatchSize 1101
EDeepLearningExecutionSettings::SetBatchSize 1101
EDeepLearningExecutionSettings::CopyTo 1101
1102
EDeepLearningExecutionSettings::SetDeviceById 1102
1102
EDeepLearningExecutionSettings::SetDeviceByName 1102
EDeepLearningExecutionSettings::GetDevices 1102
EDeepLearningExecutionSettings::SetDevices 1102
EDeepLearningExecutionSettings::EDeepLearningExecutionSettings 1103
EDeepLearningExecutionSettings::GetEngine 1103
EDeepLearningExecutionSettings::SetEngine 1103
EDeepLearningExecutionSettings::GetAvailableDevice 1103
EDeepLearningExecutionSettings::GetAvailableDeviceName 1104
EDeepLearningExecutionSettings::GetAvailableDevicesForEngine 1104
EDeepLearningExecutionSettings::GetAvailableEngineName 1104
EDeepLearningExecutionSettings::GetDevice 1105
EDeepLearningExecutionSettings::GetInferencePrecision 1105
EDeepLearningExecutionSettings::SetInferencePrecision 1105
EDeepLearningExecutionSettings::Load 1105
EDeepLearningExecutionSettings::GetNumAvailableDevices 1106
EDeepLearningExecutionSettings::GetNumAvailableEngines 1106
EDeepLearningExecutionSettings::GetNumDevices 1106
EDeepLearningExecutionSettings::operator= 1107
EDeepLearningExecutionSettings::GetOptimizeBatchSize 1107
EDeepLearningExecutionSettings::SetOptimizeBatchSize 1107
EDeepLearningExecutionSettings::Save 1107
EDeepLearningExecutionSettings::SetDevice 1108
4.78. EDeepLearningProject Class 1108
EDeepLearningProject::AddBenchmark 1111
EDeepLearningProject::AddDataAugmentation 1111
EDeepLearningProject::AddSplit 1112
EDeepLearningProject::AddTool 1112
EDeepLearningProject::CopyTo 1112
EDeepLearningProject::GetCreationDate 1113
EDeepLearningProject::GetDataset 1113
EDeepLearningProject::EDeepLearningProject 1113
EDeepLearningProject::GetFileStructureUpdateDescription 1114
EDeepLearningProject::GetBenchmark 1114
EDeepLearningProject::GetDataAugmentation 1115
EDeepLearningProject::GetDataAugmentationCreationDate 1115
EDeepLearningProject::GetDataAugmentationCreationISODate 1115
EDeepLearningProject::GetDataAugmentationName 1116
EDeepLearningProject::GetNumBenchmarks 1116
EDeepLearningProject::GetResultFilename 1116
EDeepLearningProject::GetSplit 1117
EDeepLearningProject::GetSplitCreationDate 1117
EDeepLearningProject::GetSplitCreationISODate 1117
EDeepLearningProject::GetSplitName 1118
EDeepLearningProject::GetToolCopy 1118
EDeepLearningProject::GetToolCreationDate 1119
EDeepLearningProject::GetToolDataAugmentation 1119

EDeepLearningProject::GetToolISOCreationDate ... 1119
EDeepLearningProject::GetToolName ... 1120
EDeepLearningProject::GetToolSplit ... 1120
EDeepLearningProject::GetGoodLabel ... 1120
EDeepLearningProject::SetGoodLabel ... 1120
EDeepLearningProject::HasFileStructureUpdates ... 1121
EDeepLearningProject::GetImageDirectory ... 1121
EDeepLearningProject::GetImportImageIntoProjectDirectory ... 1121
EDeepLearningProject::SetImportImageIntoProjectDirectory ... 1121
EDeepLearningProject::ImportTool ... 1121
EDeepLearningProject::GetISOCreationDate ... 1122
EDeepLearningProject::IsToolNameValid ... 1122
EDeepLearningProject::Load ... 1123
EDeepLearningProject::GetName ... 1123
EDeepLearningProject::SetName ... 1123
EDeepLearningProject::GetNumDataAugmentations ... 1123
EDeepLearningProject::GetNumSplits ... 1123
EDeepLearningProject::GetNumTools ... 1124
EDeepLearningProject::operator= ... 1124
EDeepLearningProject::GetProjectDirectory ... 1124
EDeepLearningProject::SetProjectDirectory ... 1124
EDeepLearningProject::GetProjectFile ... 1125
EDeepLearningProject::RemoveBenchmark ... 1125
EDeepLearningProject::RemoveDataAugmentaton ... 1125
EDeepLearningProject::RemoveSplit ... 1126
EDeepLearningProject::RemoveTool ... 1126
EDeepLearningProject::Save ... 1126
EDeepLearningProject::SaveProject ... 1127
EDeepLearningProject::SetBenchmark ... 1127
EDeepLearningProject::SetDataAugmentationName ... 1127
EDeepLearningProject::SetSplitName ... 1128
EDeepLearningProject::SetToolDataAugmentation ... 1128
EDeepLearningProject::SetToolName ... 1129
EDeepLearningProject::SetToolSplit ... 1129
EDeepLearningProject::GetType ... 1129
EDeepLearningProject::SetType ... 1129
EDeepLearningProject::UnsetGoodLabel ... 1130
EDeepLearningProject::UpdateProjectFileStructure ... 1130
4.79. EDeepLearningTool Class ... 1130
1136
EDeepLearningTool::SetActiveDeviceById ... 1136
1136
EDeepLearningTool::SetActiveDeviceByName ... 1136
EDeepLearningTool::GetActiveDevices ... 1137
EDeepLearningTool::SetActiveDevices ... 1137
1137
EDeepLearningTool::SetActiveDevicesById ... 1137
EDeepLearningTool::GetAutoSavePeriod ... 1137
EDeepLearningTool::SetAutoSavePeriod ... 1137
EDeepLearningTool::GetAvailableDevices ... 1138
EDeepLearningTool::GetAvailableEngines ... 1138
EDeepLearningTool::GetBatchSize ... 1138
EDeepLearningTool::SetBatchSize ... 1138
EDeepLearningTool::GetBatchSizeForMaximumInferenceSpeed ... 1139
EDeepLearningTool::GetBestIteration ... 1139
EDeepLearningTool::Create ... 1139
EDeepLearningTool::GetCurrentTrainingFinishedIterations ... 1140
EDeepLearningTool::GetCurrentTrainingNumIterations ... 1140
EDeepLearningTool::GetCurrentTrainingProgression ... 1140

EDeepLearningTool::GetDeterministicTrainingRandomSeed 1141
EDeepLearningTool::SetDeterministicTrainingRandomSeed 1141
EDeepLearningTool::GetEnableDeterministicTraining 1141
EDeepLearningTool::SetEnableDeterministicTraining 1141
EDeepLearningTool::GetEnableGPU 1141
EDeepLearningTool::SetEnableGPU 1141
EDeepLearningTool::GetEngine 1142
EDeepLearningTool::SetEngine 1142
EDeepLearningTool::GetExecutionSettings 1142
EDeepLearningTool::SetExecutionSettings 1142
EDeepLearningTool::GetActiveDevice 1142
EDeepLearningTool::GetAvailableDevice 1143
EDeepLearningTool::GetAvailableDeviceName 1143
EDeepLearningTool::GetAvailableDevicesForEngine 1143
EDeepLearningTool::GetAvailableEngineName 1144
EDeepLearningTool::GetLabel 1144
EDeepLearningTool::GetLabelColor 1144
EDeepLearningTool::GetLabelOpacity 1145
EDeepLearningTool::GetLabelWeight 1145
EDeepLearningTool::GetNumPatchesForImage 1146
EDeepLearningTool::GetOptimalNumImagesForBatchSize 1146
EDeepLearningTool::GetGPUIndexes 1147
EDeepLearningTool::SetGPUIndexes 1147
EDeepLearningTool::HasInferenceModel 1147
EDeepLearningTool::HasTrainingModel 1147
EDeepLearningTool::GetImageCacheSize 1148
EDeepLearningTool::SetImageCacheSize 1148
EDeepLearningTool::GetInferenceModelPath 1148
EDeepLearningTool::GetInferencePrecision 1148
EDeepLearningTool::SetInferencePrecision 1148
EDeepLearningTool::InitializeInference 1149
EDeepLearningTool::IsTrained 1150
EDeepLearningTool::IsTraining 1150
EDeepLearningTool::Load 1150
EDeepLearningTool::LoadInferenceModel 1151
EDeepLearningTool::LoadSettings 1151
EDeepLearningTool::LoadTrainingModel 1151
EDeepLearningTool::GetNumActiveDevices 1152
EDeepLearningTool::GetNumAvailableDevices 1152
EDeepLearningTool::GetNumAvailableEngines 1152
EDeepLearningTool::GetNumGPUs 1152
EDeepLearningTool::GetNumLabels 1152
EDeepLearningTool::GetNumTrainedIterations 1153
EDeepLearningTool::GetOptimizeBatchSize 1153
EDeepLearningTool::SetOptimizeBatchSize 1153
EDeepLearningTool::GetPath 1153
EDeepLearningTool::Save 1154
EDeepLearningTool::SaveInferenceModel 1154
EDeepLearningTool::SaveSettings 1155
EDeepLearningTool::SaveTrainingModel 1155
EDeepLearningTool::SerializeInferenceModel 1155
EDeepLearningTool::SerializeSettings 1156
EDeepLearningTool::SerializeTrainingModel 1156
EDeepLearningTool::SetActiveDevice 1156
EDeepLearningTool::SetLabel 1157
EDeepLearningTool::SetLabelColor 1157
EDeepLearningTool::SetLabelOpacity 1158
EDeepLearningTool::SetLabelWeight 1158
EDeepLearningTool::GetSettingsPath 1159

EDeepLearningTool::StopTraining 1159
EDeepLearningTool::GetToolType 1159
EDeepLearningTool::Train 1159
EDeepLearningTool::GetTrainingModelPath 1160
EDeepLearningTool::UnloadInferenceModel 1161
EDeepLearningTool::UnloadModels 1161
EDeepLearningTool::UnloadTrainingModel 1161
EDeepLearningTool::WaitForIterationCompletion 1161
EDeepLearningTool::WaitForTrainingCompletion 1162
4.80. EDepthMap Class 1162
EDepthMap::AddMetadata 1163
EDepthMap::GetAxisSystemType 1164
EDepthMap::SetAxisSystemType 1164
EDepthMap::Clear 1164
EDepthMap::ClearMetadata 1164
EDepthMap::ConvertCoordinatesMapToPixel 1165
EDepthMap::ConvertCoordinatesPixelToMap 1165
EDepthMap::DeleteMetadata 1166
EDepthMap::Draw 1166
EDepthMap::DrawImage 1169
EDepthMap::GetBufferPtr 1171
EDepthMap::GetCheckedBufferPtr 1172
EDepthMap::GetMetadata 1172
EDepthMap::GetZValue 1173
EDepthMap::GetHeight 1173
EDepthMap::SetHeight 1173
EDepthMap::IsVoid 1173
EDepthMap::Load 1174
EDepthMap::LoadImage 1174
EDepthMap::LoadImageAndMetadata 1175
EDepthMap::LoadMetadata 1175
EDepthMap::ModifyMetadata 1176
EDepthMap::GetRowPitch 1176
EDepthMap::Save 1176
EDepthMap::SaveImage 1177
EDepthMap::SaveImageAndMetadata 1177
EDepthMap::SaveJpeg 1178
EDepthMap::SaveJpeg2K 1178
EDepthMap::SaveMetadata 1178
EDepthMap::SetBufferPtr 1179
EDepthMap::SetSize 1179
EDepthMap::GetType 1180
EDepthMap::GetWidth 1180
EDepthMap::SetWidth 1180
EDepthMap::GetZResolution 1181
EDepthMap::SetZResolution 1181
4.81. EDepthMap16 Class 1181
EDepthMap16::AddMetadata 1183
EDepthMap16::AsEImage 1183
EDepthMap16::GetAxisSystemType 1183
EDepthMap16::SetAxisSystemType 1183
EDepthMap16::Clear 1183
EDepthMap16::ClearMetadata 1184
EDepthMap16::ConvertCoordinatesMapToPixel 1184
EDepthMap16::ConvertCoordinatesPixelToMap 1184
EDepthMap16::CopyMetadataTo 1185
EDepthMap16::DeleteMetadata 1185
EDepthMap16::Draw 1186
EDepthMap16::DrawImage 1189

EDepthMap16::EDepthMap16 1191
EDepthMap16::FillUndefinedPixels 1191
EDepthMap16::FillUndefinedPixelsWithMedian 1192
EDepthMap16::GetBufferPtr 1192
EDepthMap16::GetCheckedBufferPtr 1193
EDepthMap16::GetMetadata 1194
EDepthMap16::GetPixel 1194
EDepthMap16::GetZValue 1194
EDepthMap16::GetHeight 1195
EDepthMap16::SetHeight 1195
EDepthMap16::IsVoid 1195
EDepthMap16::Load 1195
EDepthMap16::LoadImage 1196
EDepthMap16::LoadImageAndMetadata 1196
EDepthMap16::LoadMetadata 1197
EDepthMap16::ModifyMetadata 1197
EDepthMap16::operator= 1198
EDepthMap16::GetRowPitch 1198
EDepthMap16::Save 1198
EDepthMap16::SaveImage 1199
EDepthMap16::SaveImageAndMetadata 1199
EDepthMap16::SaveJpeg 1200
EDepthMap16::SaveJpeg2K 1200
EDepthMap16::SaveMetadata 1201
EDepthMap16::SetBufferPtr 1201
EDepthMap16::SetPixel 1201
EDepthMap16::SetSize 1202
EDepthMap16::GetType 1203
EDepthMap16::GetUndefinedValue 1203
EDepthMap16::GetWidth 1203
EDepthMap16::SetWidth 1203
EDepthMap16::GetZResolution 1203
EDepthMap16::SetZResolution 1203
4.82. EDepthMap32f Class 1204
EDepthMap32f::AddMetadata 1205
EDepthMap32f::AsEImage 1206
EDepthMap32f::GetAxisSystemType 1206
EDepthMap32f::SetAxisSystemType 1206
EDepthMap32f::Clear 1206
EDepthMap32f::ClearMetadata 1206
EDepthMap32f::ConvertCoordinatesMapToPixel 1207
EDepthMap32f::ConvertCoordinatesPixelToMap 1207
EDepthMap32f::CopyMetadataTo 1208
EDepthMap32f::DeleteMetadata 1208
EDepthMap32f::Draw 1208
EDepthMap32f::DrawImage 1211
EDepthMap32f::EDepthMap32f 1213
EDepthMap32f::FillUndefinedPixels 1214
EDepthMap32f::FillUndefinedPixelsWithMedian 1214
EDepthMap32f::GetBufferPtr 1215
EDepthMap32f::GetCheckedBufferPtr 1216
EDepthMap32f::GetMetadata 1216
EDepthMap32f::GetPixel 1216
EDepthMap32f::GetZValue 1217
EDepthMap32f::GetHeight 1217
EDepthMap32f::SetHeight 1217
EDepthMap32f::IsVoid 1218
EDepthMap32f::Load 1218
EDepthMap32f::LoadImage 1218

EDepthMap32f::LoadImageAndMetadata 1219
EDepthMap32f::LoadMetadata 1219
EDepthMap32f::ModifyMetadata 1220
EDepthMap32f::operator= 1220
EDepthMap32f::GetRowPitch 1220
EDepthMap32f::Save 1221
EDepthMap32f::SaveImage 1221
EDepthMap32f::SaveImageAndMetadata 1222
EDepthMap32f::SaveJpeg 1222
EDepthMap32f::SaveJpeg2K 1223
EDepthMap32f::SaveMetadata 1223
EDepthMap32f::SetBufferPtr 1223
EDepthMap32f::SetPixel 1224
EDepthMap32f::SetSize 1224
EDepthMap32f::GetType 1225
EDepthMap32f::GetUndefinedValue 1225
EDepthMap32f::GetWidth 1226
EDepthMap32f::SetWidth 1226
EDepthMap32f::GetZResolution 1226
EDepthMap32f::SetZResolution 1226
4.83. EDepthMap8 Class 1226
EDepthMap8::AddMetadata 1228
EDepthMap8::AsEImage 1228
EDepthMap8::GetAxisSystemType 1229
EDepthMap8::SetAxisSystemType 1229
EDepthMap8::Clear 1229
EDepthMap8::ClearMetadata 1229
EDepthMap8::ConvertCoordinatesMapToPixel 1229
EDepthMap8::ConvertCoordinatesPixelToMap 1230
EDepthMap8::CopyMetadataTo 1230
EDepthMap8::DeleteMetadata 1231
EDepthMap8::Draw 1231
EDepthMap8::DrawImage 1234
EDepthMap8::EDepthMap8 1236
EDepthMap8::FillUndefinedPixels 1237
EDepthMap8::FillUndefinedPixelsWithMedian 1237
EDepthMap8::GetBufferPtr 1238
EDepthMap8::GetCheckedBufferPtr 1239
EDepthMap8::GetMetadata 1239
EDepthMap8::GetPixel 1239
EDepthMap8::GetZValue 1240
EDepthMap8::GetHeight 1240
EDepthMap8::SetHeight 1240
EDepthMap8::IsVoid 1241
EDepthMap8::Load 1241
EDepthMap8::LoadImage 1241
EDepthMap8::LoadImageAndMetadata 1242
EDepthMap8::LoadMetadata 1242
EDepthMap8::ModifyMetadata 1243
EDepthMap8::operator= 1243
EDepthMap8::GetRowPitch 1243
EDepthMap8::Save 1244
EDepthMap8::SaveImage 1244
EDepthMap8::SaveImageAndMetadata 1245
EDepthMap8::SaveJpeg 1245
EDepthMap8::SaveJpeg2K 1246
EDepthMap8::SaveMetadata 1246
EDepthMap8::SetBufferPtr 1246
EDepthMap8::SetPixel 1247

EDepthMap8::SetSize 1247
EDepthMap8::GetType 1248
EDepthMap8::GetUndefinedValue 1248
EDepthMap8::GetWidth 1249
EDepthMap8::SetWidth 1249
EDepthMap8::GetZResolution 1249
EDepthMap8::SetZResolution 1249
4.84. EDepthMapToMeshConverter Class 1249
EDepthMapToMeshConverter::GetCalibrationModel 1250
EDepthMapToMeshConverter::SetCalibrationModel 1250
EDepthMapToMeshConverter::Convert 1250
EDepthMapToMeshConverter::EDepthMapToMeshConverter 1251
EDepthMapToMeshConverter::Load 1251
EDepthMapToMeshConverter::operator= 1252
EDepthMapToMeshConverter::Save 1252
4.85. EDepthMapToPointCloudConverter Class 1252
EDepthMapToPointCloudConverter::GetCalibrationModel 1253
EDepthMapToPointCloudConverter::SetCalibrationModel 1253
EDepthMapToPointCloudConverter::Convert 1253
EDepthMapToPointCloudConverter::EDepthMapToPointCloudConverter 1254
EDepthMapToPointCloudConverter::Load 1255
EDepthMapToPointCloudConverter::operator= 1255
EDepthMapToPointCloudConverter::Save 1255
4.86. EDrawableExtent Class 1256
EDrawableExtent::GetBottomExclusive 1256
EDrawableExtent::DoesContainHorizontalLine 1257
EDrawableExtent::DoesContainPoint 1257
EDrawableExtent::DoesContainRectangle 1257
EDrawableExtent::EDrawableExtent 1258
EDrawableExtent::GetHeight 1259
EDrawableExtent::IsInfinite 1259
EDrawableExtent::GetLeft 1259
EDrawableExtent::operator= 1260
EDrawableExtent::GetRightExclusive 1260
EDrawableExtent::GetTop 1260
EDrawableExtent::GetWidth 1260
4.87. EDrawAdapter Class 1261
EDrawAdapter::Arc 1262
EDrawAdapter::BackedText 1262
EDrawAdapter::GetBrush 1263
EDrawAdapter::SetBrush 1263
EDrawAdapter::DrawMask 1264
EDrawAdapter::DrawPoint 1264
EDrawAdapter::Ellipse 1265
EDrawAdapter::FilledEllipse 1265
EDrawAdapter::FilledPolygon 1266
EDrawAdapter::FilledRectangle 1267
EDrawAdapter::FilledRotatedEllipse 1267
EDrawAdapter::GetFont 1268
EDrawAdapter::SetFont 1268
EDrawAdapter::GetTextSize 1269
EDrawAdapter::Image 1269
EDrawAdapter::Line 1270
EDrawAdapter::GetPen 1271
EDrawAdapter::SetPen 1271
EDrawAdapter::Polygon 1271
EDrawAdapter::Rectangle 1272
EDrawAdapter::RotatedEllipse 1272
EDrawAdapter::Text 1273

EDrawAdapter::UseCurrentBrush 1274
EDrawAdapter::UseCurrentPen 1274
4.88. EEllipseRegion Class 1274
EEllipseRegion::GetAngle 1275
EEllipseRegion::SetAngle 1275
EEllipseRegion::GetCenter 1276
EEllipseRegion::SetCenter 1276
EEllipseRegion::Drag 1276
EEllipseRegion::EEllipseRegion 1277
EEllipseRegion::HitTest 1278
EEllipseRegion::GetHorizontalRadius 1279
EEllipseRegion::SetHorizontalRadius 1279
EEllipseRegion::Load 1279
EEllipseRegion::operator!= 1280
EEllipseRegion::operator= 1280
EEllipseRegion::operator== 1281
EEllipseRegion::Rotate 1281
EEllipseRegion::Save 1281
EEllipseRegion::Scale 1282
EEllipseRegion::Translate 1282
EEllipseRegion::GetVerticalRadius 1283
EEllipseRegion::SetVerticalRadius 1283
4.89. EErrorStatistics Class 1283
EErrorStatistics::EErrorStatistics 1284
EErrorStatistics::Load 1284
EErrorStatistics::GetMax 1285
EErrorStatistics::SetMax 1285
EErrorStatistics::GetMean 1285
EErrorStatistics::SetMean 1285
EErrorStatistics::GetMin 1285
EErrorStatistics::SetMin 1285
EErrorStatistics::GetNumOfErrors 1286
EErrorStatistics::SetNumOfErrors 1286
EErrorStatistics::GetNumOfValidSamples 1286
EErrorStatistics::SetNumOfValidSamples 1286
EErrorStatistics::operator= 1286
EErrorStatistics::Save 1287
EErrorStatistics::GetStdDev 1287
EErrorStatistics::SetStdDev 1287
4.90. EException Class 1287
EException::EException 1288
EException::GetError 1288
EException::SetError 1288
EException::operator= 1289
EException::What 1289
4.91. EExplicitGeometricCalibrationModel Class 1289
EExplicitGeometricCalibrationModel::GetCameraAngle 1290
EExplicitGeometricCalibrationModel::GetCameraHeight 1290
EExplicitGeometricCalibrationModel::EExplicitGeometricCalibrationModel 1291
EExplicitGeometricCalibrationModel::GetFocalLength 1292
EExplicitGeometricCalibrationModel::IsInitialized 1292
EExplicitGeometricCalibrationModel::GetLaserPlaneAngle 1292
EExplicitGeometricCalibrationModel::Load 1293
EExplicitGeometricCalibrationModel::GetMotionIncrement 1293
EExplicitGeometricCalibrationModel::operator= 1293
EExplicitGeometricCalibrationModel::operator== 1294
EExplicitGeometricCalibrationModel::GetRoiBottomLine 1294
EExplicitGeometricCalibrationModel::GetRoiLeftColumn 1294
EExplicitGeometricCalibrationModel::Save 1294

EExplicitGeometricCalibrationModel::GetSensorHeight ... 1295
EExplicitGeometricCalibrationModel::GetSensorWidth ... 1295
EExplicitGeometricCalibrationModel::GetSensorXResolution ... 1295
EExplicitGeometricCalibrationModel::GetSensorYResolution ... 1296
EExplicitGeometricCalibrationModel::GetType ... 1296
4.92. EExternalDrawAdapter Class ... 1296
EExternalDrawAdapter::Arc ... 1298
1298
EExternalDrawAdapter::SetArcFunctionPtr ... 1298
EExternalDrawAdapter::BackedText ... 1299
1299
EExternalDrawAdapter::SetBackedTextFunctionPtr ... 1299
1299
EExternalDrawAdapter::SetBrush ... 1299
EExternalDrawAdapter::DrawPoint ... 1300
1300
EExternalDrawAdapter::SetDrawPointFunctionPtr ... 1300
EExternalDrawAdapter::EExternalDrawAdapter ... 1300
EExternalDrawAdapter::Ellipse ... 1301
1301
EExternalDrawAdapter::SetEllipseFunctionPtr ... 1301
EExternalDrawAdapter::GetExtent ... 1302
EExternalDrawAdapter::FilledEllipse ... 1302
EExternalDrawAdapter::FilledPolygon ... 1302
EExternalDrawAdapter::FilledRectangle ... 1303
1304
EExternalDrawAdapter::SetFillEllipseFunctionPtr ... 1304
1304
EExternalDrawAdapter::SetFillPolygonFunctionPtr ... 1304
1304
EExternalDrawAdapter::SetFillRectangleFunctionPtr ... 1304
1304
EExternalDrawAdapter::SetFont ... 1304
1304
EExternalDrawAdapter::SetGetExtentFunctionPtr ... 1304
EExternalDrawAdapter::GetTextSize ... 1305
1305
EExternalDrawAdapter::SetGetTextSizeFunctionPtr ... 1305
EExternalDrawAdapter::Image ... 1305
1306
EExternalDrawAdapter::SetImageFunctionPtr ... 1306
1306
EExternalDrawAdapter::SetImageWithColorScaleFunctionPtr ... 1306
1307
EExternalDrawAdapter::SetImageWithGrayscaleScaleFunctionPtr ... 1307
EExternalDrawAdapter::Line ... 1307
1307
EExternalDrawAdapter::SetLineFunctionPtr ... 1307
EExternalDrawAdapter::operator= ... 1308
1308
EExternalDrawAdapter::SetPen ... 1308
EExternalDrawAdapter::Polygon ... 1308
1309
EExternalDrawAdapter::SetPolygonFunctionPtr ... 1309
EExternalDrawAdapter::Rectangle ... 1309
1310
EExternalDrawAdapter::SetRectangleFunctionPtr ... 1310
1310
EExternalDrawAdapter::SetSetBrushFunctionPtr ... 1310

1310
EExternalDrawAdapter::SetSetFontFunctionPtr ... 1310
1310
EExternalDrawAdapter::SetSetPenFunctionPtr ... 1310
EExternalDrawAdapter::Text ... 1310
1311
EExternalDrawAdapter::SetTextFunctionPtr ... 1311
EExternalDrawAdapter::UseCurrentBrush ... 1311
1312
EExternalDrawAdapter::SetUseCurrentBrushFunctionPtr ... 1312
EExternalDrawAdapter::UseCurrentPen ... 1312
1312
EExternalDrawAdapter::SetUseCurrentPenFunctionPtr ... 1312
4.93. EFeaturesAligner Class ... 1312
EFeaturesAligner::Compute ... 1313
EFeaturesAligner::EFeaturesAligner ... 1313
EFeaturesAligner::Load ... 1314
EFeaturesAligner::GetModelPoints ... 1314
EFeaturesAligner::SetModelPoints ... 1314
EFeaturesAligner::operator= ... 1315
EFeaturesAligner::GetPolarityTransform ... 1315
EFeaturesAligner::SetPolarityTransform ... 1315
EFeaturesAligner::Save ... 1315
4.94. EFilePointerSerializer Class ... 1316
EFilePointerSerializer::Close ... 1316
EFilePointerSerializer::GetWriting ... 1316
4.95. EFileSerializer Class ... 1317
EFileSerializer::Close ... 1317
EFileSerializer::GetWriting ... 1317
4.96. EFilters Class ... 1317
EFilters::Median ... 1318
EFilters::RemoveNoise ... 1321
4.97. EFindFeaturePoint Class ... 1322
EFindFeaturePoint::EFindFeaturePoint ... 1323
EFindFeaturePoint::GetGradientX ... 1323
EFindFeaturePoint::GetGradientY ... 1324
EFindFeaturePoint::operator!= ... 1324
EFindFeaturePoint::operator= ... 1324
EFindFeaturePoint::operator== ... 1324
EFindFeaturePoint::GetPosition ... 1325
4.98. EFloatRange Class ... 1325
EFloatRange::GetCenter ... 1326
EFloatRange::EFloatRange ... 1326
EFloatRange::IsInRange ... 1326
EFloatRange::IsValid ... 1327
EFloatRange::GetLowerBound ... 1327
EFloatRange::operator= ... 1327
EFloatRange::operator== ... 1328
EFloatRange::SetBounds ... 1328
EFloatRange::SetFromBaseAndAbsoluteTolerance ... 1328
EFloatRange::SetFromBaseAndRelativeTolerance ... 1329
EFloatRange::GetSize ... 1329
EFloatRange::Update ... 1330
EFloatRange::GetUpperBound ... 1330
4.99. EFont Class ... 1330
EFont::EFont ... 1331
EFont::GetFamily ... 1332
EFont::SetFamily ... 1332

EFont::IsValid ... 1332
EFont::Load ... 1332
EFont::operator!= ... 1333
EFont::operator= ... 1333
EFont::operator== ... 1333
EFont::Save ... 1334
EFont::Serialize ... 1334
EFont::GetSize ... 1334
EFont::SetSize ... 1334
EFont::GetStyle ... 1335
EFont::SetStyle ... 1335
4.100. EFoundPattern Class ... 1335
EFoundPattern::GetAngle ... 1336
EFoundPattern::GetCenter ... 1336
EFoundPattern::Draw ... 1336
EFoundPattern::GetDrawBoundingBox ... 1338
EFoundPattern::SetDrawBoundingBox ... 1338
EFoundPattern::GetDrawCenter ... 1338
EFoundPattern::SetDrawCenter ... 1338
EFoundPattern::GetDrawFeaturePoints ... 1338
EFoundPattern::SetDrawFeaturePoints ... 1338
EFoundPattern::DrawWithCurrentPen ... 1339
EFoundPattern::EFoundPattern ... 1339
EFoundPattern::operator!= ... 1340
EFoundPattern::operator= ... 1340
EFoundPattern::operator== ... 1340
EFoundPattern::GetQuadrangle ... 1341
EFoundPattern::GetScale ... 1341
EFoundPattern::GetScore ... 1341
EFoundPattern::ToRegion ... 1342
4.101. EFourierTransformer Class ... 1342
EFourierTransformer::DirectTransform ... 1342
EFourierTransformer::EFourierTransformer ... 1343
EFourierTransformer::GetFrequentialDomainFormat ... 1344
EFourierTransformer::SetFrequentialDomainFormat ... 1344
EFourierTransformer::InverseTransform ... 1344
EFourierTransformer::Load ... 1345
EFourierTransformer::operator!= ... 1345
EFourierTransformer::operator= ... 1345
EFourierTransformer::operator== ... 1346
EFourierTransformer::Save ... 1346
4.102. EFrame Class ... 1346
EFrame::GetAngle ... 1347
EFrame::SetAngle ... 1347
EFrame::GetCenterX ... 1347
EFrame::GetCenterY ... 1348
EFrame::CopyTo ... 1348
EFrame::EFrame ... 1348
EFrame::GlobalToLocal ... 1349
EFrame::Load ... 1350
EFrame::LocalToGlobal ... 1350
EFrame::operator= ... 1351
EFrame::Save ... 1351
EFrame::GetScale ... 1351
EFrame::SetScale ... 1351
4.103. EFrameShape Class ... 1352
EFrameShape::GetAngle ... 1353
EFrameShape::SetAngle ... 1353
EFrameShape::GetCenter ... 1353

EFrameShape::SetCenter 1353
EFrameShape::GetCenterX 1353
EFrameShape::GetCenterY 1353
EFrameShape::Closest 1354
EFrameShape::CopyTo 1354
EFrameShape::Drag 1354
EFrameShape::Draw 1355
EFrameShape::DrawWithCurrentPen 1356
EFrameShape::EFrameShape 1356
EFrameShape::HitTest 1357
EFrameShape::operator= 1357
EFrameShape::GetScale 1358
EFrameShape::SetScale 1358
EFrameShape::Set 1358
EFrameShape::SetCenterXY 1358
EFrameShape::SetSize 1359
EFrameShape::GetSizeX 1359
EFrameShape::GetSizeY 1360
EFrameShape::GetType 1360
4.104. EGDIPlusDrawAdapter Class 1360
EGDIPlusDrawAdapter::EGDIPlusDrawAdapter 1360
4.105. EGenericDrawAdapter Class 1361
EGenericDrawAdapter::Arc 1362
EGenericDrawAdapter::BackedText 1362
EGenericDrawAdapter::DrawPoint 1363
EGenericDrawAdapter::EGenericDrawAdapter 1364
EGenericDrawAdapter::Ellipse 1364
EGenericDrawAdapter::FilledEllipse 1365
EGenericDrawAdapter::FilledPolygon 1365
EGenericDrawAdapter::FilledRectangle 1366
EGenericDrawAdapter::GetFont 1367
EGenericDrawAdapter::SetFont 1367
EGenericDrawAdapter::GetTextSize 1367
EGenericDrawAdapter::Image 1367
EGenericDrawAdapter::Line 1368
EGenericDrawAdapter::Polygon 1369
EGenericDrawAdapter::Rectangle 1369
EGenericDrawAdapter::Text 1370
EGenericDrawAdapter::GetUseAntialiasing 1371
EGenericDrawAdapter::SetUseAntialiasing 1371
EGenericDrawAdapter::UseCurrentBrush 1371
EGenericDrawAdapter::UseCurrentPen 1371
4.106. EGrabberDepthMap16 Class 1372
EGrabberDepthMap16::EGrabberDepthMap16 1372
4.107. EGrabberDepthMap8 Class 1372
EGrabberDepthMap8::EGrabberDepthMap8 1373
4.108. EGrabberImageBW16 Class 1373
EGrabberImageBW16::EGrabberImageBW16 1373
4.109. EGrabberImageBW8 Class 1374
EGrabberImageBW8::EGrabberImageBW8 1374
4.110. EGrabberImageC24 Class 1375
EGrabberImageC24::EGrabberImageC24 1375
4.111. EGrayscaleDoubleThresholdSegmenter Class 1376
EGrayscaleDoubleThresholdSegmenter::GetHighThreshold 1376
EGrayscaleDoubleThresholdSegmenter::SetHighThreshold 1376
EGrayscaleDoubleThresholdSegmenter::Load 1376
EGrayscaleDoubleThresholdSegmenter::GetLowThreshold 1377
EGrayscaleDoubleThresholdSegmenter::SetLowThreshold 1377

EGrayscaleDoubleThresholdSegmenter::operator== 1377
EGrayscaleDoubleThresholdSegmenter::Save 1378
4.112. EGrayscaleSingleThresholdSegmenter Class 1378
EGrayscaleSingleThresholdSegmenter::GetAbsoluteThreshold 1379
EGrayscaleSingleThresholdSegmenter::SetAbsoluteThreshold 1379
EGrayscaleSingleThresholdSegmenter::IsFirstApplication 1379
EGrayscaleSingleThresholdSegmenter::GetLastThreshold 1379
EGrayscaleSingleThresholdSegmenter::Load 1380
EGrayscaleSingleThresholdSegmenter::GetMode 1380
EGrayscaleSingleThresholdSegmenter::SetMode 1380
EGrayscaleSingleThresholdSegmenter::operator== 1381
EGrayscaleSingleThresholdSegmenter::GetRelativeThreshold 1381
EGrayscaleSingleThresholdSegmenter::SetRelativeThreshold 1381
EGrayscaleSingleThresholdSegmenter::Save 1381
4.113. EGridDecimator Class 1382
EGridDecimator::GetCellSize 1382
EGridDecimator::SetCellSize 1382
EGridDecimator::Decimate 1383
EGridDecimator::EGridDecimator 1383
EGridDecimator::operator!= 1384
EGridDecimator::operator= 1384
EGridDecimator::operator== 1384
4.114. EGs1Translator Class 1385
EGs1Translator::GetHumanReadableCode 1385
4.115. EHarrisCornerDetector Class 1385
EHarrisCornerDetector::Apply 1386
EHarrisCornerDetector::GetDerivationScale 1387
EHarrisCornerDetector::SetDerivationScale 1387
EHarrisCornerDetector::EHarrisCornerDetector 1387
EHarrisCornerDetector::GetGradientNormalizationEnabled 1387
EHarrisCornerDetector::SetGradientNormalizationEnabled 1387
EHarrisCornerDetector::GetIntegrationScale 1388
EHarrisCornerDetector::SetIntegrationScale 1388
EHarrisCornerDetector::GetSubpixelPrecisionEnabled 1388
EHarrisCornerDetector::SetSubpixelPrecisionEnabled 1388
EHarrisCornerDetector::GetThreshold 1388
EHarrisCornerDetector::SetThreshold 1388
EHarrisCornerDetector::GetThresholdingMode 1389
EHarrisCornerDetector::SetThresholdingMode 1389
4.116. EHarrisInterestPoints Class 1389
EHarrisInterestPoints::Draw 1390
EHarrisInterestPoints::DrawCorner 1391
EHarrisInterestPoints::DrawCornerWithCurrentPen 1392
EHarrisInterestPoints::DrawWithCurrentPen 1393
EHarrisInterestPoints::EHarrisInterestPoints 1394
EHarrisInterestPoints::GetCornerness 1394
EHarrisInterestPoints::GetGradientMagnitude 1394
EHarrisInterestPoints::GetGradientOrientation 1394
EHarrisInterestPoints::GetGradientX 1395
EHarrisInterestPoints::GetGradientY 1395
EHarrisInterestPoints::GetPoint 1395
EHarrisInterestPoints::GetX 1396
EHarrisInterestPoints::GetY 1396
EHarrisInterestPoints::GetPointCount 1396
4.117. EHDRColorFuser Class 1397
EHDRColorFuser::EHDRColorFuser 1397
EHDRColorFuser::Fuse 1398
EHDRColorFuser::GetFusedImage 1398

EHDRColorFuser::operator= 1398
4.118. EHDRFuser Class 1399
EHDRFuser::EHDRFuser 1399
EHDRFuser::Fuse 1400
EHDRFuser::GetFusedImage 1400
EHDRFuser::operator= 1401
4.119. EHitAndMissKernel Class 1401
EHitAndMissKernel::EHitAndMissKernel 1401
EHitAndMissKernel::GetEndX 1402
EHitAndMissKernel::GetEndY 1403
EHitAndMissKernel::GetValue 1403
EHitAndMissKernel::SetSize 1403
EHitAndMissKernel::SetValue 1404
EHitAndMissKernel::GetStartX 1405
EHitAndMissKernel::GetStartY 1405
4.120. EHole Class 1405
EHole::EHole 1406
EHole::operator= 1406
EHole::GetParentObjectIndex 1406
4.121. EImageBW1 Class 1406
EImageBW1::EImageBW1 1407
EImageBW1::GetBitIndex 1408
EImageBW1::InitializeFromUnalignedBuffer 1408
EImageBW1::operator= 1409
4.122. EImageBW16 Class 1409
EImageBW16::EImageBW16 1410
EImageBW16::InitializeFromUnalignedBuffer 1411
EImageBW16::operator= 1411
4.123. EImageBW32 Class 1412
EImageBW32::EImageBW32 1412
EImageBW32::InitializeFromUnalignedBuffer 1413
EImageBW32::operator= 1414
4.124. EImageBW32f Class 1414
EImageBW32f::EImageBW32f 1414
EImageBW32f::InitializeFromUnalignedBuffer 1415
EImageBW32f::operator= 1416
4.125. EImageBW8 Class 1416
EImageBW8::EImageBW8 1417
EImageBW8::InitializeFromUnalignedBuffer 1418
EImageBW8::operator= 1418
4.126. EImageC15 Class 1419
EImageC15::EImageC15 1419
EImageC15::InitializeFromUnalignedBuffer 1420
EImageC15::operator= 1421
4.127. EImageC16 Class 1421
EImageC16::EImageC16 1421
EImageC16::InitializeFromUnalignedBuffer 1422
EImageC16::operator= 1423
4.128. EImageC24 Class 1423
EImageC24::EImageC24 1424
EImageC24::InitializeFromUnalignedBuffer 1425
EImageC24::operator= 1425
4.129. EImageC24A Class 1426
EImageC24A::EImageC24A 1426
EImageC24A::InitializeFromUnalignedBuffer 1427
EImageC24A::operator= 1428
4.130. EImageC48 Class 1428
EImageC48::EImageC48 1428

EImageC48::InitializeFromUnalignedBuffer 1429
EImageC48::operator= 1430
4.131. EImageEncoder Class 1430
EImageEncoder::GetBinaryImageSegmenter 1432
EImageEncoder::GetColorRangeThresholdSegmenter 1432
EImageEncoder::GetColorSingleThresholdSegmenter 1432
EImageEncoder::GetContinuousModeEnabled 1433
EImageEncoder::SetContinuousModeEnabled 1433
EImageEncoder::GetContinuousModeMaxHeight 1433
EImageEncoder::SetContinuousModeMaxHeight 1433
EImageEncoder::CopyTo 1433
EImageEncoder::EImageEncoder 1434
EImageEncoder::Encode 1434
EImageEncoder::GetEncodingConnexity 1436
EImageEncoder::SetEncodingConnexity 1436
EImageEncoder::FlushContinuousMode 1436
EImageEncoder::GetGrayscaleDoubleThresholdSegmenter 1437
EImageEncoder::GetGrayscaleSingleThresholdSegmenter 1437
EImageEncoder::GetImageRangeSegmenter 1437
EImageEncoder::GetLabeledImageSegmenter 1437
EImageEncoder::Load 1438
EImageEncoder::operator!= 1438
EImageEncoder::operator= 1438
EImageEncoder::operator== 1439
EImageEncoder::GetReferenceImageSegmenter 1439
EImageEncoder::ResetContinuousMode 1439
EImageEncoder::Save 1439
EImageEncoder::GetSegmentationMethod 1440
EImageEncoder::SetSegmentationMethod 1440
EImageEncoder::Serialize 1440
4.132. EImageRangeSegmenter Class 1441
1442
EImageRangeSegmenter::SetBlackLayerEncoded 1442
1442
EImageRangeSegmenter::SetBlackLayerIndex 1442
EImageRangeSegmenter::GetHighImageBW16 1442
EImageRangeSegmenter::SetHighImageBW16 1442
EImageRangeSegmenter::GetHighImageBW8 1443
EImageRangeSegmenter::SetHighImageBW8 1443
EImageRangeSegmenter::GetHighImageC24 1443
EImageRangeSegmenter::SetHighImageC24 1443
EImageRangeSegmenter::Load 1443
EImageRangeSegmenter::GetLowImageBW16 1444
EImageRangeSegmenter::SetLowImageBW16 1444
EImageRangeSegmenter::GetLowImageBW8 1444
EImageRangeSegmenter::SetLowImageBW8 1444
EImageRangeSegmenter::GetLowImageC24 1444
EImageRangeSegmenter::SetLowImageC24 1444
EImageRangeSegmenter::operator== 1445
EImageRangeSegmenter::Save 1445
1445
EImageRangeSegmenter::SetWhiteLayerEncoded 1445
1446
EImageRangeSegmenter::SetWhiteLayerIndex 1446
4.133. EImageSegmenter Class 1446
4.134. EIntegerRange Class 1446
EIntegerRange::GetCenter 1447
EIntegerRange::EIntegerRange 1447
EIntegerRange::IsInRange 1447

EIntegerRange::GetLowerBound 1448
EIntegerRange::operator= 1448
EIntegerRange::SetBounds 1449
EIntegerRange::SetFromBaseAndAbsoluteTolerance 1449
EIntegerRange::SetFromBaseAndRelativeTolerance 1449
EIntegerRange::GetSize 1450
EIntegerRange::Update 1450
EIntegerRange::GetUpperBound 1450
4.135. EInterestPointLocator Class 1451
EInterestPointLocator::GetAbsoluteMinDistance 1452
EInterestPointLocator::SetAbsoluteMinDistance 1452
EInterestPointLocator::EInterestPointLocator 1452
EInterestPointLocator::GetObjectSize 1453
EInterestPointLocator::SetObjectSize 1453
EInterestPointLocator::operator= 1453
EInterestPointLocator::GetSameLabelMinDistance 1454
EInterestPointLocator::SetSameLabelMinDistance 1454
EInterestPointLocator::SerializeSettings 1454
EInterestPointLocator::GetToolType 1454
4.136. EKernel Class 1454
EKernel::EKernel 1455
EKernel::GetGain 1456
EKernel::SetGain 1456
EKernel::GetKernelData 1457
EKernel::GetOffset 1457
EKernel::SetOffset 1457
EKernel::GetOutsideValue 1457
EKernel::SetOutsideValue 1457
EKernel::GetRawDataPtr 1458
EKernel::GetRectifier 1458
EKernel::SetRectifier 1458
EKernel::SetKernelData 1458
EKernel::SetSize 1463
EKernel::GetSizeX 1463
EKernel::GetSizeY 1464
4.137. ELabeledImageSegmenter Class 1464
ELabeledImageSegmenter::Load 1465
ELabeledImageSegmenter::GetMaxLayer 1465
ELabeledImageSegmenter::SetMaxLayer 1465
ELabeledImageSegmenter::GetMinLayer 1465
ELabeledImageSegmenter::SetMinLayer 1465
ELabeledImageSegmenter::operator== 1466
ELabeledImageSegmenter::Save 1466
4.138. ELandmark Class 1466
ELandmark::ELandmark 1467
ELandmark::operator= 1467
ELandmark::GetSensorX 1468
ELandmark::SetSensorX 1468
ELandmark::GetSensorY 1468
ELandmark::SetSensorY 1468
ELandmark::GetWorldX 1468
ELandmark::SetWorldX 1468
ELandmark::GetWorldY 1469
ELandmark::SetWorldY 1469
4.139. ELaserLineExtractor Class 1469
ELaserLineExtractor::GetAnalysisMode 1470
ELaserLineExtractor::SetAnalysisMode 1470
ELaserLineExtractor::GetAnalysisThreshold 1470

ELaserLineExtractor::SetAnalysisThreshold 1470
ELaserLineExtractor::GetDepthMap 1470
ELaserLineExtractor::ELaserLineExtractor 1471
ELaserLineExtractor::GetEnableSmoothing 1471
ELaserLineExtractor::SetEnableSmoothing 1471
ELaserLineExtractor::ExtractProfileFromFrame 1472
ELaserLineExtractor::operator= 1472
ELaserLineExtractor::GetProfile 1472
ELaserLineExtractor::SetSmoothingParameters 1473
4.140. ELine Class 1473
ELine::CopyTo 1474
ELine::ELine 1474
ELine::GetEnd 1475
ELine::GetAngleBetweenLines 1475
ELine::GetDistanceBetweenPointAndLine 1476
ELine::GetIntersectionOfLines 1476
ELine::GetPoint 1477
ELine::GetProjectionOfPointOnLine 1477
ELine::GetLength 1478
ELine::SetLength 1478
ELine::Load 1478
ELine::operator= 1479
ELine::GetOrg 1479
ELine::Save 1479
ELine::SetFromOriginAndEnd 1480
ELine::SetFromTwoPoints 1480
4.141. ELineGauge Class 1481
1483
ELineGauge::SetActive 1483
ELineGauge::AddSkipRange 1484
ELineGauge::GetAverageDistance 1484
ELineGauge::GetClippingMode 1485
ELineGauge::SetClippingMode 1485
ELineGauge::CopyTo 1485
ELineGauge::Drag 1486
ELineGauge::Draw 1486
ELineGauge::DrawWithCurrentPen 1487
ELineGauge::ELineGauge 1487
ELineGauge::GetFilteringThreshold 1488
ELineGauge::SetFilteringThreshold 1488
ELineGauge::GetMeasuredPeak 1488
ELineGauge::GetMeasuredPoint 1489
ELineGauge::GetMinNumFitSamples 1490
ELineGauge::GetSample 1490
ELineGauge::GetSkipRange 1491
ELineGauge::HitTest 1491
ELineGauge::GetHVConstraint 1492
ELineGauge::SetHVConstraint 1492
ELineGauge::GetKnownAngle 1492
ELineGauge::SetKnownAngle 1492
1493
ELineGauge::SetLine 1493
ELineGauge::Measure 1493
ELineGauge::GetMeasuredLine 1493
ELineGauge::MeasureSample 1494
ELineGauge::MeasureWithoutFitting 1494
ELineGauge::GetMinAmplitude 1495
ELineGauge::SetMinAmplitude 1495
ELineGauge::GetMinArea 1495

ELineGauge::SetMinArea 1495
ELineGauge::GetNumFilteringPasses 1496
ELineGauge::SetNumFilteringPasses 1496
ELineGauge::GetNumMeasuredPoints 1496
ELineGauge::GetNumSamples 1497
ELineGauge::GetNumSkipRanges 1497
ELineGauge::GetNumValidSamples 1497
ELineGauge::operator= 1497
ELineGauge::Plot 1498
ELineGauge::PlotWithCurrentPen 1499
ELineGauge::Process 1500
ELineGauge::GetRectangularSamplingArea 1501
ELineGauge::SetRectangularSamplingArea 1501
ELineGauge::RemoveAllSkipRanges 1501
ELineGauge::RemoveSkipRange 1501
ELineGauge::GetSamplingStep 1502
ELineGauge::SetSamplingStep 1502
ELineGauge::SetMinNumFitSamples 1502
ELineGauge::GetSmoothing 1503
ELineGauge::SetSmoothing 1503
ELineGauge::GetThickness 1503
ELineGauge::SetThickness 1503
ELineGauge::GetThreshold 1504
ELineGauge::SetThreshold 1504
ELineGauge::GetTolerance 1504
ELineGauge::SetTolerance 1504
ELineGauge::GetTransitionChoice 1505
ELineGauge::SetTransitionChoice 1505
ELineGauge::GetTransitionIndex 1505
ELineGauge::SetTransitionIndex 1505
ELineGauge::GetTransitionType 1505
ELineGauge::SetTransitionType 1505
ELineGauge::GetType 1506
ELineGauge::GetValid 1506
4.142. ELineShape Class 1506
ELineShape::GetAngle 1508
ELineShape::SetAngle 1508
ELineShape::GetCenter 1508
ELineShape::SetCenter 1508
ELineShape::GetCenterX 1508
ELineShape::GetCenterY 1508
ELineShape::Closest 1509
ELineShape::CopyTo 1509
ELineShape::Drag 1509
ELineShape::Draw 1510
ELineShape::DrawWithCurrentPen 1511
ELineShape::ELineShape 1511
ELineShape::GetEnd 1512
ELineShape::GetPoint 1512
ELineShape::HitTest 1513
ELineShape::GetLength 1513
ELineShape::SetLength 1513
1514
ELineShape::SetLine 1514
ELineShape::operator= 1514
ELineShape::GetOrg 1514
ELineShape::GetScale 1514
ELineShape::SetScale 1514
ELineShape::SetCenterXY 1515

ELineShape::SetFromOriginAndEnd 1515
ELineShape::SetFromTwoPoints 1515
ELineShape::GetType 1516
4.143. EListItem Class 1516
4.144. ELocator Class 1517
ELocator::GetAbsoluteMaxOverlap 1519
ELocator::SetAbsoluteMaxOverlap 1519
ELocator::ELocator 1519
ELocator::GenerateAnchors 1519
ELocator::operator= 1520
ELocator::GetSameLabelMaxOverlap 1521
ELocator::SetSameLabelMaxOverlap 1521
ELocator::SerializeSettings 1521
ELocator::GetToolType 1521
4.145. ELocatorBase Class 1522
ELocatorBase::GetAbsoluteMaxObjectProximity 1524
ELocatorBase::SetAbsoluteMaxObjectProximity 1524
ELocatorBase::Apply 1525
ELocatorBase::GetCapacity 1526
ELocatorBase::SetCapacity 1526
ELocatorBase::GetChannels 1526
ELocatorBase::SetChannels 1526
ELocatorBase::GetDetectionThreshold 1527
ELocatorBase::SetDetectionThreshold 1527
ELocatorBase::Evaluate 1527
ELocatorBase::GetTrainingMetrics 1527
ELocatorBase::GetValidationMetrics 1528
ELocatorBase::HasFeature 1528
ELocatorBase::GetHeight 1528
ELocatorBase::SetHeight 1528
ELocatorBase::GetLocatorFeatures 1529
ELocatorBase::GetMaxNumberOfObjects 1529
ELocatorBase::SetMaxNumberOfObjects 1529
ELocatorBase::GetPredictionAnchors 1529
ELocatorBase::SetPredictionAnchors 1529
ELocatorBase::GetSameLabelMaxObjectProximity 1530
ELocatorBase::SetSameLabelMaxObjectProximity 1530
ELocatorBase::SerializeSettings 1530
ELocatorBase::GetWidth 1531
ELocatorBase::SetWidth 1531
4.146. ELocatorMetrics Class 1531
ELocatorMetrics::GetAveragePrecision 1533
ELocatorMetrics::GetAveragePrecision50 1533
ELocatorMetrics::GetAverageProximity 1534
ELocatorMetrics::GetDetectionThreshold 1534
ELocatorMetrics::SetDetectionThreshold 1534
ELocatorMetrics::ELocatorMetrics 1534
ELocatorMetrics::GetError 1535
ELocatorMetrics::GetFScore 1535
ELocatorMetrics::GetBestWeightedFScore 1535
ELocatorMetrics::GetBestWeightedFScoreAndThreshold 1535
ELocatorMetrics::GetBestWeightedFScoreThreshold 1536
ELocatorMetrics::GetBestWeightedPrecision 1536
ELocatorMetrics::GetBestWeightedPrecisionAndThreshold 1537
ELocatorMetrics::GetBestWeightedPrecisionThreshold 1537
ELocatorMetrics::GetBestWeightedRecall 1538
ELocatorMetrics::GetBestWeightedRecallAndThreshold 1538
ELocatorMetrics::GetBestWeightedRecallThreshold 1539
ELocatorMetrics::GetLabel 1539

ELocatorMetrics::GetLabelAveragePrecision 1539
ELocatorMetrics::GetLabelAverageProximity 1540
ELocatorMetrics::GetLabelFScore 1540
ELocatorMetrics::GetLabelIntersectionOverUnion 1540
ELocatorMetrics::GetLabelPrecision 1541
ELocatorMetrics::GetLabelRecall 1541
ELocatorMetrics::GetNumCorrectlyDetectedObjects 1542
ELocatorMetrics::GetNumDetectedObjects 1542
ELocatorMetrics::GetNumUndetectedObjects 1543
ELocatorMetrics::GetWeightedFScore 1543
ELocatorMetrics::GetWeightedPrecision 1544
ELocatorMetrics::GetWeightedRecall 1544
ELocatorMetrics::GetImageAccuracy 1545
ELocatorMetrics::GetIntersectionOverUnion 1545
ELocatorMetrics::IsValid 1545
ELocatorMetrics::Load 1545
ELocatorMetrics::GetNumBadlyPredictedImagesWithObjects 1546
ELocatorMetrics::GetNumBadlyPredictedImagesWithoutObjects 1546
ELocatorMetrics::GetNumCorrectlyPredictedImagesWithObjects 1546
ELocatorMetrics::GetNumCorrectlyPredictedImagesWithoutObjects 1546
ELocatorMetrics::GetNumLabels 1547
ELocatorMetrics::operator= 1547
ELocatorMetrics::GetPrecision 1547
ELocatorMetrics::GetRecall 1547
ELocatorMetrics::Save 1548
4.147. ELocatorObject Class 1548
ELocatorObject::Draw 1549
ELocatorObject::ELocatorObject 1550
ELocatorObject::GetFeatures 1551
ELocatorObject::HasFeature 1552
ELocatorObject::GetHeight 1552
ELocatorObject::SetHeight 1552
ELocatorObject::IsValid 1552
ELocatorObject::GetLabel 1552
ELocatorObject::SetLabel 1552
ELocatorObject::GetObjectSize 1553
ELocatorObject::operator!= 1553
ELocatorObject::operator= 1553
ELocatorObject::operator== 1554
ELocatorObject::GetOrgX 1554
ELocatorObject::SetOrgX 1554
ELocatorObject::GetOrgY 1554
ELocatorObject::SetOrgY 1554
ELocatorObject::GetPositionX 1555
ELocatorObject::SetPositionX 1555
ELocatorObject::GetPositionY 1555
ELocatorObject::SetPositionY 1555
ELocatorObject::GetRectangleRegion 1555
ELocatorObject::SetRectangleRegion 1555
ELocatorObject::SetOriginAndSize 1556
ELocatorObject::GetWidth 1556
ELocatorObject::SetWidth 1556
4.148. ELocatorPredictedObject Class 1556
ELocatorPredictedObject::Draw 1557
ELocatorPredictedObject::ELocatorPredictedObject 1558
ELocatorPredictedObject::operator= 1558
ELocatorPredictedObject::GetProbability 1559
4.149. ELocatorResult Class 1559
ELocatorResult::GetDetectedObjects 1560

ELocatorResult::GetDetectionThreshold 1560
ELocatorResult::Draw 1560
ELocatorResult::ELocatorResult 1562
ELocatorResult::GetLabel 1562
ELocatorResult::GetLabelColor 1562
ELocatorResult::GetNumDetectedObjects 1563
ELocatorResult::GetGroundtruthObjects 1563
ELocatorResult::HasGroundtruth 1563
ELocatorResult::IsValid 1564
ELocatorResult::Load 1564
ELocatorResult::GetLocatorFeatures 1564
ELocatorResult::GetNumLabels 1565
ELocatorResult::GetObjectSize 1565
ELocatorResult::operator= 1565
ELocatorResult::RemoveGroundtruth 1565
ELocatorResult::Save 1566
4.150. EMailBarcode Class 1566
EMailBarcode::GetChecksumOk 1566
EMailBarcode::GetComponentStrings 1567
EMailBarcode::Draw 1567
EMailBarcode::EMailBarcode 1568
EMailBarcode::operator= 1568
EMailBarcode::GetOrientation 1569
EMailBarcode::GetPosition 1569
EMailBarcode::GetSymbology 1569
EMailBarcode::GetText 1569
4.151. EMailBarcodeReader Class 1570
EMailBarcodeReader::EMailBarcodeReader 1570
EMailBarcodeReader::GetEnableClutteredBarcodes 1571
EMailBarcodeReader::SetEnableClutteredBarcodes 1571
EMailBarcodeReader::GetEnableDottedBarcodes 1571
EMailBarcodeReader::SetEnableDottedBarcodes 1571
EMailBarcodeReader::GetExpectedOrientations 1571
EMailBarcodeReader::SetExpectedOrientations 1571
EMailBarcodeReader::GetExpectedSymbologies 1572
EMailBarcodeReader::SetExpectedSymbologies 1572
EMailBarcodeReader::Load 1572
EMailBarcodeReader::operator= 1572
EMailBarcodeReader::Read 1573
EMailBarcodeReader::Save 1573
EMailBarcodeReader::GetValidateChecksum 1574
EMailBarcodeReader::SetValidateChecksum 1574
4.152. EMatcher Class 1574
EMatcher::GetAdvancedLearning 1577
EMatcher::SetAdvancedLearning 1577
EMatcher::ClearImage 1578
EMatcher::GetContrastMode 1578
EMatcher::SetContrastMode 1578
EMatcher::CopyLearntPattern 1578
EMatcher::CopyTo 1579
EMatcher::GetCorrelationMode 1579
EMatcher::SetCorrelationMode 1579
EMatcher::GetDontCareThreshold 1579
EMatcher::SetDontCareThreshold 1579
EMatcher::DrawPosition 1580
EMatcher::DrawPositions 1581
EMatcher::DrawPositionsWithCurrentPen 1583
EMatcher::DrawPositionWithCurrentPen 1584
EMatcher::EMatcher 1584

EMatcher::GetEnableEarlyCandidateRejection 1585
EMatcher::SetEnableEarlyCandidateRejection 1585
EMatcher::GetFilteringMode 1586
EMatcher::SetFilteringMode 1586
EMatcher::GetFinalReduction 1586
EMatcher::SetFinalReduction 1586
EMatcher::GetAngleStep 1587
EMatcher::GetExtension 1587
EMatcher::GetPixelDimensions 1588
EMatcher::GetPosition 1588
EMatcher::GetScaleStep 1588
EMatcher::GetScaleXStep 1589
EMatcher::GetScaleYStep 1589
EMatcher::GetInitialMinScore 1589
EMatcher::SetInitialMinScore 1589
EMatcher::GetInterpolate 1590
EMatcher::SetInterpolate 1590
EMatcher::GetIsotropicScale 1590
EMatcher::LearnPattern 1591
EMatcher::Load 1591
EMatcher::Match 1592
EMatcher::GetMaxAngle 1593
EMatcher::SetMaxAngle 1593
EMatcher::GetMaxInitialPositions 1593
EMatcher::SetMaxInitialPositions 1593
EMatcher::GetMaxOverlap 1593
EMatcher::SetMaxOverlap 1593
EMatcher::GetMaxPositions 1594
EMatcher::SetMaxPositions 1594
EMatcher::GetMaxScale 1594
EMatcher::SetMaxScale 1594
EMatcher::GetMaxScaleX 1595
EMatcher::SetMaxScaleX 1595
EMatcher::GetMaxScaleY 1595
EMatcher::SetMaxScaleY 1595
EMatcher::GetMinAngle 1595
EMatcher::SetMinAngle 1595
EMatcher::GetMinReducedArea 1596
EMatcher::SetMinReducedArea 1596
EMatcher::GetMinScale 1596
EMatcher::SetMinScale 1596
EMatcher::GetMinScaleX 1597
EMatcher::SetMinScaleX 1597
EMatcher::GetMinScaleY 1597
EMatcher::SetMinScaleY 1597
EMatcher::GetMinScore 1598
EMatcher::SetMinScore 1598
EMatcher::GetNumPositions 1598
EMatcher::GetNumReductions 1598
EMatcher::operator= 1599
EMatcher::GetPatternHeight 1599
EMatcher::GetPatternLearnt 1599
EMatcher::GetPatternType 1599
EMatcher::GetPatternWidth 1600
EMatcher::GetPositions 1600
EMatcher::Save 1600
EMatcher::SetExtension 1601
EMatcher::SetPixelDimensions 1601
EMatcher::GetVersion 1602

4.153. EMatrixCode Class ... 1602
EMatrixCode::GetAngle ... 1604
EMatrixCode::GetAxialNonUniformity ... 1605
EMatrixCode::GetAxialNonUniformityGrade ... 1605
EMatrixCode::GetCellDefects ... 1605
EMatrixCode::GetCenter ... 1606
EMatrixCode::GetContrast ... 1606
EMatrixCode::GetContrastGrade ... 1606
EMatrixCode::GetContrastType ... 1606
EMatrixCode::GetDataMatrixCellHeight ... 1607
EMatrixCode::GetDataMatrixCellWidth ... 1607
EMatrixCode::GetDecodedString ... 1607
EMatrixCode::Draw ... 1608
EMatrixCode::DrawErrors ... 1609
EMatrixCode::DrawErrorsWithCurrentPen ... 1610
EMatrixCode::DrawWithCurrentPen ... 1610
EMatrixCode::EMatrixCode ... 1611
EMatrixCode::GetFamily ... 1612
EMatrixCode::GetFinderPatternDefects ... 1612
EMatrixCode::GetFlipping ... 1612
EMatrixCode::GetFound ... 1613
EMatrixCode::GetCorner ... 1613
EMatrixCode::GetDecodedDataElement ... 1614
EMatrixCode::GetHorizontalMarkGrowth ... 1614
EMatrixCode::GetHorizontalMarkMisplacement ... 1615
EMatrixCode::IsGS1 ... 1615
EMatrixCode::GetIso15415GradingParameters ... 1615
EMatrixCode::GetIso29158GradingParameters ... 1616
EMatrixCode::Load ... 1616
EMatrixCode::GetLocationThreshold ... 1616
EMatrixCode::GetLogicalSize ... 1617
EMatrixCode::GetLogicalSizeHeight ... 1617
EMatrixCode::GetLogicalSizeWidth ... 1617
EMatrixCode::GetMeasuredPrintGrowth ... 1617
EMatrixCode::GetNumErrors ... 1618
EMatrixCode::operator= ... 1618
EMatrixCode::GetOverallGrade ... 1618
EMatrixCode::GetPrintGrowth ... 1619
EMatrixCode::GetPrintGrowthGrade ... 1619
EMatrixCode::GetReadingThreshold ... 1619
EMatrixCode::Save ... 1620
EMatrixCode::GetSemiT10GradingParameters ... 1620
EMatrixCode::SetCorner ... 1620
EMatrixCode::GetSymbolContrastSNR ... 1621
EMatrixCode::GetUnusedErrorCorrection ... 1621
EMatrixCode::GetUnusedErrorCorrectionGrade ... 1621
EMatrixCode::GetVerticalMarkGrowth ... 1622
EMatrixCode::GetVerticalMarkMisplacement ... 1622
4.154. EMatrixCode Class ... 1622
EMatrixCode::GetDecodedString ... 1624
EMatrixCode::GetDecodedStringStream ... 1624
EMatrixCode::DrawErrors ... 1624
EMatrixCode::DrawErrorsWithCurrentPen ... 1625
EMatrixCode::DrawGrid ... 1626
EMatrixCode::DrawGridWithCurrentPen ... 1626
EMatrixCode::DrawPosition ... 1627
EMatrixCode::DrawPositionWithCurrentPen ... 1628
EMatrixCode::GetECC000Family ... 1629
EMatrixCode::EMatrixCode ... 1629

EMatrixCode::GetErrors 1629
EMatrixCode::GetCellColor 1630
EMatrixCode::GetCellCorrectedColor 1630
EMatrixCode::GetCellPosition 1631
EMatrixCode::GetIsECC200 1631
EMatrixCode::GetIsGraded 1632
EMatrixCode::GetIsGS1 1632
EMatrixCode::GetIso15415GradingParameters 1632
EMatrixCode::GetIso29158GradingParameters 1632
EMatrixCode::GetIsReliable 1632
EMatrixCode::operator= 1633
EMatrixCode::GetPosition 1633
EMatrixCode::GetReliabilityScore 1634
EMatrixCode::GetSemiT10GradingParameters 1634
EMatrixCode::GetSymbolHeight 1634
EMatrixCode::GetSymbolPolarity 1634
EMatrixCode::GetSymbolWidth 1634
4.155. EMatrixCodeGrid Class 1635
EMatrixCodeGrid::EMatrixCodeGrid 1635
1636
EMatrixCodeGrid::SetEnableAll 1636
EMatrixCodeGrid::GetCellEnabled 1636
EMatrixCodeGrid::GetResults 1636
EMatrixCodeGrid::GetNumCols 1637
EMatrixCodeGrid::GetNumRows 1637
EMatrixCodeGrid::operator= 1637
EMatrixCodeGrid::SetEnableCell 1638
EMatrixCodeGrid::SetEnableColumn 1638
EMatrixCodeGrid::SetEnableRow 1639
4.156. EMatrixCodeReader Class 1639
EMatrixCodeReader::GetComputeGrading 1641
EMatrixCodeReader::SetComputeGrading 1641
EMatrixCodeReader::EMatrixCodeReader 1641
EMatrixCodeReader::GetLearnMaskElement 1641
EMatrixCodeReader::Learn 1642
EMatrixCodeReader::LearnMore 1642
EMatrixCodeReader::Load 1643
EMatrixCodeReader::GetMaxHeightWidthRatio 1643
EMatrixCodeReader::SetMaxHeightWidthRatio 1643
EMatrixCodeReader::GetMaximumPrintGrowth 1644
EMatrixCodeReader::SetMaximumPrintGrowth 1644
EMatrixCodeReader::GetMinimumPrintGrowth 1644
EMatrixCodeReader::SetMinimumPrintGrowth 1644
EMatrixCodeReader::GetNominalPrintGrowth 1645
EMatrixCodeReader::SetNominalPrintGrowth 1645
EMatrixCodeReader::Read 1645
EMatrixCodeReader::Reset 1646
EMatrixCodeReader::Save 1646
EMatrixCodeReader::GetSearchParams 1647
EMatrixCodeReader::SetIso29158CalibrationParameters 1647
EMatrixCodeReader::SetLearnMaskElement 1648
EMatrixCodeReader::GetTimeOut 1648
EMatrixCodeReader::SetTimeOut 1648
4.157. EMatrixCodeReader Class 1649
EMatrixCodeReader::GetComputeGrading 1650
EMatrixCodeReader::SetComputeGrading 1650
EMatrixCodeReader::EMatrixCodeReader 1651
EMatrixCodeReader::GetEnableDMRE 1651
EMatrixCodeReader::SetEnableDMRE 1651

EMatrixCodeReader::GetEnablePermissiveDecoding 1651
EMatrixCodeReader::SetEnablePermissiveDecoding 1651
EMatrixCodeReader::GetEnableReturnUnreliableCodes 1652
EMatrixCodeReader::SetEnableReturnUnreliableCodes 1652
EMatrixCodeReader::GetIso29158CalibrationParameters 1652
EMatrixCodeReader::SetIso29158CalibrationParameters 1652
EMatrixCodeReader::Learn 1652
EMatrixCodeReader::GetLearnPerformed 1653
EMatrixCodeReader::Load 1653
EMatrixCodeReader::GetMatrixCodeDimensionsRange 1654
EMatrixCodeReader::SetMatrixCodeDimensionsRange 1654
EMatrixCodeReader::GetMaxNumCodes 1654
EMatrixCodeReader::SetMaxNumCodes 1654
EMatrixCodeReader::operator= 1654
EMatrixCodeReader::Read 1655
EMatrixCodeReader::GetReadMode 1656
EMatrixCodeReader::SetReadMode 1656
EMatrixCodeReader::GetReadResults 1656
EMatrixCodeReader::ResetLearning 1657
EMatrixCodeReader::Save 1657
EMatrixCodeReader::StopProcess 1657
EMatrixCodeReader::GetTimeOut 1658
EMatrixCodeReader::SetTimeOut 1658
EMatrixCodeReader::UnsetMatrixCodeDimensionsRange 1658
4.158. EMeasurementUnit Class 1658
EMeasurementUnit::ConversionFactorTo 1659
EMeasurementUnit::EMeasurementUnit 1659
EMeasurementUnit::GetStockMeasurementUnit 1660
EMeasurementUnit::GetMagnitude 1660
EMeasurementUnit::SetMagnitude 1660
EMeasurementUnit::GetName 1660
EMeasurementUnit::SetName 1660
4.159. EMemorySerializer Class 1661
EMemorySerializer::GetBuffer 1661
EMemorySerializer::GetBufferSize 1661
EMemorySerializer::Close 1661
EMemorySerializer::GetCurrentPosition 1662
EMemorySerializer::GetWriting 1662
4.160. EMesh Class 1662
EMesh::Clear 1663
EMesh::ComputePlaneBehind 1663
EMesh::EMesh 1664
EMesh::Load 1665
EMesh::LoadSTL 1665
EMesh::GetNormals 1665
EMesh::operator= 1666
EMesh::GetPointCloud 1666
EMesh::Save 1666
EMesh::SaveSTL 1667
EMesh::GetTriangleCount 1667
EMesh::GetTriangleIndexes 1667
4.161. EMeshToZMapConverter Class 1668
EMeshToZMapConverter::Convert 1671
EMeshToZMapConverter::EMeshToZMapConverter 1672
EMeshToZMapConverter::EnableFillMode 1673
EMeshToZMapConverter::GetExtension 1673
EMeshToZMapConverter::SetExtension 1673
EMeshToZMapConverter::GetFillUndefinedPixelsDirection 1673
EMeshToZMapConverter::GetFillUndefinedPixelsMethod 1674

EMeshToZMapConverter::IsFillModeEnabled 1674
EMeshToZMapConverter::Load 1674
EMeshToZMapConverter::GetMapHeight 1675
EMeshToZMapConverter::GetMapWidth 1675
EMeshToZMapConverter::GetMapXResolution 1675
EMeshToZMapConverter::GetMapYResolution 1675
EMeshToZMapConverter::GetMapZResolution 1676
EMeshToZMapConverter::SetMapZResolution 1676
EMeshToZMapConverter::operator= 1676
EMeshToZMapConverter::operator== 1676
EMeshToZMapConverter::GetOrientationVector 1677
EMeshToZMapConverter::SetOrientationVector 1677
EMeshToZMapConverter::GetOrientationVectorMode 1677
EMeshToZMapConverter::SetOrientationVectorMode 1677
EMeshToZMapConverter::GetOrigin 1678
EMeshToZMapConverter::SetOrigin 1678
EMeshToZMapConverter::GetReferencePlane 1678
EMeshToZMapConverter::SetReferencePlane 1678
EMeshToZMapConverter::GetReferencePlaneMode 1678
EMeshToZMapConverter::SetReferencePlaneMode 1678
EMeshToZMapConverter::Save 1679
EMeshToZMapConverter::SetFillMode 1679
EMeshToZMapConverter::SetFillModeMedian 1680
EMeshToZMapConverter::SetMapSize 1680
EMeshToZMapConverter::SetMapXYResolution 1680
EMeshToZMapConverter::UnsetMapSize 1681
EMeshToZMapConverter::UnsetMapXYResolution 1681
EMeshToZMapConverter::UnsetMapZResolution 1682
EMeshToZMapConverter::UnsetOrigin 1682
EMeshToZMapConverter::UnsetWorldToZMapTransform 1682
EMeshToZMapConverter::GetWorldToZMapTransform 1683
EMeshToZMapConverter::SetWorldToZMapTransform 1683
EMeshToZMapConverter::GetZMaptoWorldTransform 1683
EMeshToZMapConverter::SetZMaptoWorldTransform 1683
4.162. EMovingAverage Class 1683
EMovingAverage::Average 1684
EMovingAverage::EMovingAverage 1684
EMovingAverage::GetSize 1685
EMovingAverage::Reset 1686
EMovingAverage::SetSize 1686
EMovingAverage::GetSrcImage 1687
4.163. EObject Class 1687
EObject::EObject 1688
EObject::GetHole 1688
EObject::GetHoleCount 1688
EObject::operator= 1689
4.164. EObjectBasedCalibrationGenerator Class 1689
EObjectBasedCalibrationGenerator::GetCalibrationObjectScaleX 1692
EObjectBasedCalibrationGenerator::GetCalibrationObjectScaleY 1692
EObjectBasedCalibrationGenerator::GetCalibrationObjectScaleZ 1692
EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeA 1692
EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeB 1693
EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeC 1693
EObjectBasedCalibrationGenerator::Compute 1693
EObjectBasedCalibrationGenerator::EObjectBasedCalibrationGenerator 1693
EObjectBasedCalibrationGenerator::GetCalibrationObjectType 1694
EObjectBasedCalibrationGenerator::Load 1694
EObjectBasedCalibrationGenerator::GetNumCalibrationPasses 1695
EObjectBasedCalibrationGenerator::SetNumCalibrationPasses 1695

EObjectBasedCalibrationGenerator::operator= 1695
EObjectBasedCalibrationGenerator::GetPrecisionVsSpeedTradeOff 1695
EObjectBasedCalibrationGenerator::SetPrecisionVsSpeedTradeOff 1695
EObjectBasedCalibrationGenerator::GetRangeX 1696
EObjectBasedCalibrationGenerator::SetRangeX 1696
EObjectBasedCalibrationGenerator::GetRangeY 1696
EObjectBasedCalibrationGenerator::SetRangeY 1696
EObjectBasedCalibrationGenerator::GetRangeZ 1696
EObjectBasedCalibrationGenerator::SetRangeZ 1696
EObjectBasedCalibrationGenerator::Save 1697
EObjectBasedCalibrationGenerator::SetCalibrationObjectScale 1697
EObjectBasedCalibrationGenerator::SetCalibrationObjectType 1698
4.165. EObjectBasedCalibrationModel Class 1699
EObjectBasedCalibrationModel::GetCalibrationError 1699
EObjectBasedCalibrationModel::GetCalibrationRelativeError 1700
EObjectBasedCalibrationModel::EObjectBasedCalibrationModel 1700
EObjectBasedCalibrationModel::IsInitialized 1700
EObjectBasedCalibrationModel::Load 1700
EObjectBasedCalibrationModel::operator= 1701
EObjectBasedCalibrationModel::Save 1701
EObjectBasedCalibrationModel::GetType 1702
4.166. EObjectRunsIterator Class 1702
EObjectRunsIterator::GetEndX 1703
EObjectRunsIterator::EObjectRunsIterator 1703
EObjectRunsIterator::First 1703
EObjectRunsIterator::GetIsDone 1704
EObjectRunsIterator::GetLength 1704
EObjectRunsIterator::Next 1704
EObjectRunsIterator::operator= 1704
EObjectRunsIterator::GetStartX 1705
EObjectRunsIterator::GetY 1705
4.167. EObjectSelection Class 1705
EObjectSelection::Add 1708
EObjectSelection::AddHole 1708
EObjectSelection::AddHoles 1709
EObjectSelection::AddHolesOfSelectedObjects 1710
EObjectSelection::AddLayer 1710
EObjectSelection::AddObject 1710
EObjectSelection::AddObjects 1711
EObjectSelection::AddObjectsUsingFloatFeature 1711
EObjectSelection::AddObjectsUsingIntegerFeature 1713
EObjectSelection::AddObjectsUsingRectangle 1714
EObjectSelection::AddObjectsUsingRegion 1715
EObjectSelection::AddObjectsUsingUnsignedIntegerFeature 1716
EObjectSelection::AddObjectUsingPosition 1717
EObjectSelection::GetAttachedImage 1717
EObjectSelection::SetAttachedImage 1717
EObjectSelection::Clear 1718
EObjectSelection::ClearFeatureCache 1718
EObjectSelection::GetElementCount 1718
EObjectSelection::EObjectSelection 1718
EObjectSelection::FeatureAverage 1719
EObjectSelection::FeatureDeviation 1719
EObjectSelection::FeatureVariance 1719
EObjectSelection::GetFeretAngle 1720
EObjectSelection::SetFeretAngle 1720
EObjectSelection::FloatFeatureMaximum 1720
EObjectSelection::FloatFeatureMinimum 1720
EObjectSelection::GetElement 1721

EObjectSelection::GetFloatFeature ... 1721
EObjectSelection::GetIndexOfElement ... 1722
EObjectSelection::GetIntegerFeature ... 1722
EObjectSelection::GetUnsignedIntegerFeature ... 1722
EObjectSelection::IntegerFeatureMaximum ... 1723
EObjectSelection::IntegerFeatureMinimum ... 1723
EObjectSelection::IsSelected ... 1723
EObjectSelection::operator== ... 1724
EObjectSelection::Remove ... 1725
EObjectSelection::RemoveHole ... 1725
EObjectSelection::RemoveHoles ... 1726
EObjectSelection::RemoveLayer ... 1727
EObjectSelection::RemoveObject ... 1727
EObjectSelection::RemoveObjectsUsingRectangle ... 1728
EObjectSelection::RemoveObjectsUsingRegion ... 1729
EObjectSelection::RemoveObjectUsingPosition ... 1730
EObjectSelection::RemoveSelectedHoles ... 1730
EObjectSelection::RemoveUsingFloatFeature ... 1730
EObjectSelection::RemoveUsingIntegerFeature ... 1731
EObjectSelection::RemoveUsingUnsignedIntegerFeature ... 1732
EObjectSelection::RenderMask ... 1733
EObjectSelection::Sort ... 1733
EObjectSelection::ToRegion ... 1734
EObjectSelection::UnsignedIntegerFeatureMaximum ... 1734
EObjectSelection::UnsignedIntegerFeatureMinimum ... 1734
4.168. EObjectTemplateMatcher Class ... 1735
EObjectTemplateMatcher::BuildTemplate ... 1736
EObjectTemplateMatcher::GetEnableAlignment ... 1737
EObjectTemplateMatcher::SetEnableAlignment ... 1737
EObjectTemplateMatcher::EObjectTemplateMatcher ... 1737
EObjectTemplateMatcher::GetUnpairedObjects ... 1737
EObjectTemplateMatcher::Load ... 1738
EObjectTemplateMatcher::GetMaximumDistance ... 1738
EObjectTemplateMatcher::SetMaximumDistance ... 1738
EObjectTemplateMatcher::GetNumberOfPairedObjects ... 1738
EObjectTemplateMatcher::operator= ... 1739
EObjectTemplateMatcher::Save ... 1739
EObjectTemplateMatcher::GetSelectionIndexes ... 1739
EObjectTemplateMatcher::SortPositions ... 1740
EObjectTemplateMatcher::SortSelection ... 1740
EObjectTemplateMatcher::GetTemplateIndexes ... 1741
4.169. EOCR Class ... 1741
EOCR::AddChar ... 1744
EOCR::AddPatternFromImage ... 1745
EOCR::BuildObjects ... 1746
EOCR::CharGetDstX ... 1746
EOCR::CharGetDstY ... 1747
EOCR::CharGetHeight ... 1747
EOCR::CharGetOrgX ... 1747
EOCR::CharGetOrgY ... 1748
EOCR::CharGetTotalDstX ... 1748
EOCR::CharGetTotalDstY ... 1748
EOCR::CharGetTotalOrgX ... 1749
EOCR::CharGetTotalOrgY ... 1749
EOCR::CharGetWidth ... 1750
EOCR::GetCharSpacing ... 1750
EOCR::SetCharSpacing ... 1750
EOCR::GetCompareAspectRatio ... 1750
EOCR::SetCompareAspectRatio ... 1750

EOCR::GetCutLargeChars ... 1751
EOCR::SetCutLargeChars ... 1751
EOCR::DrawChar ... 1751
EOCR::DrawChars ... 1752
EOCR::DrawCharsWithCurrentPen ... 1754
EOCR::DrawCharWithCurrentPen ... 1754
EOCR::DrawObjects ... 1755
EOCR::EmptyChars ... 1756
EOCR::EOCR ... 1756
EOCR::FindAllChars ... 1757
EOCR::GetConfidenceRatio ... 1757
EOCR::GetFirstCharCode ... 1758
EOCR::GetFirstCharDistance ... 1758
EOCR::GetPatternBitmap ... 1758
EOCR::GetPatternClass ... 1759
EOCR::GetPatternCode ... 1759
EOCR::GetSecondCharCode ... 1759
EOCR::GetSecondCharDistance ... 1760
EOCR::HitChar ... 1760
EOCR::HitChars ... 1761
EOCR::LearnPattern ... 1762
EOCR::LearnPatterns ... 1763
EOCR::GetLineSpacingMode ... 1764
EOCR::SetLineSpacingMode ... 1764
EOCR::Load ... 1764
EOCR::MatchChar ... 1765
EOCR::GetMatchingMode ... 1766
EOCR::SetMatchingMode ... 1766
EOCR::GetMaxCharHeight ... 1766
EOCR::SetMaxCharHeight ... 1766
EOCR::GetMaxCharWidth ... 1766
EOCR::SetMaxCharWidth ... 1766
EOCR::GetMinCharHeight ... 1767
EOCR::SetMinCharHeight ... 1767
EOCR::GetMinCharWidth ... 1767
EOCR::SetMinCharWidth ... 1767
EOCR::NewFont ... 1768
EOCR::GetNoiseArea ... 1768
EOCR::SetNoiseArea ... 1768
EOCR::GetNumChars ... 1768
EOCR::GetNumPatterns ... 1769
EOCR::operator= ... 1769
EOCR::GetPatternHeight ... 1769
EOCR::GetPatternWidth ... 1769
EOCR::ReadText ... 1770
EOCR::ReadTextWide ... 1770
EOCR::Recognize ... 1771
EOCR::RecognizeWide ... 1772
EOCR::GetRelativeSpacing ... 1773
EOCR::SetRelativeSpacing ... 1773
EOCR::GetRelativeThreshold ... 1773
EOCR::SetRelativeThreshold ... 1773
EOCR::GetRemoveBorder ... 1774
EOCR::SetRemoveBorder ... 1774
EOCR::GetRemoveNarrowOrFlat ... 1774
EOCR::SetRemoveNarrowOrFlat ... 1774
EOCR::RemovePattern ... 1774
EOCR::Save ... 1775
EOCR::GetSegmentationMode ... 1775

EOCR::SetSegmentationMode 1775
EOCR::SetPatternClass 1776
EOCR::SetPatternCode 1776
EOCR::GetShiftingMode 1776
EOCR::SetShiftingMode 1776
EOCR::GetShiftXTolerance 1777
EOCR::SetShiftXTolerance 1777
EOCR::GetShiftYTolerance 1777
EOCR::SetShiftYTolerance 1777
EOCR::GetTextColor 1778
EOCR::SetTextColor 1778
EOCR::GetThreshold 1778
EOCR::SetThreshold 1778
EOCR::GetTrueThreshold 1778
4.170. EOCR2 Class 1779
EOCR2::AddCharactersToDatabase 1785
EOCR2::AddClassifierForSymbol 1786
EOCR2::GetAllowedCharacterTypes 1786
EOCR2::SetAllowedCharacterTypes 1786
EOCR2::GetCharacterDatabase 1787
EOCR2::SetCharacterDatabase 1787
EOCR2::GetCharsHeight 1787
EOCR2::SetCharsHeight 1787
EOCR2::GetCharsMaxFragmentation 1787
EOCR2::SetCharsMaxFragmentation 1787
EOCR2::GetCharsSpacingBias 1788
EOCR2::SetCharsSpacingBias 1788
EOCR2::GetCharsWidthBias 1788
EOCR2::SetCharsWidthBias 1788
EOCR2::GetCharsWidthRange 1789
EOCR2::SetCharsWidthRange 1789
EOCR2::GetClassifier 1789
EOCR2::SetClassifier 1789
EOCR2::ClearCharacterDatabase 1789
EOCR2::ClearResult 1790
EOCR2::Detect 1790
EOCR2::GetDetectionDelta 1791
EOCR2::SetDetectionDelta 1791
EOCR2::GetDetectionMethod 1791
EOCR2::SetDetectionMethod 1791
EOCR2::DrawDetection 1791
EOCR2::DrawDetectionWithCurrentPen 1793
EOCR2::DrawRecognition 1793
EOCR2::DrawRecognitionWithCurrentPen 1795
EOCR2::DrawSegmentation 1796
EOCR2::DrawSegmentationWithCurrentPen 1797
EOCR2::GetEnableCutLargeCharacter 1798
EOCR2::SetEnableCutLargeCharacter 1798
EOCR2::GetEnabledTopology 1798
EOCR2::SetEnabledTopology 1798
EOCR2::GetEnableGPU 1798
EOCR2::SetEnableGPU 1798
EOCR2::GetEnableOffSizeCharacter 1799
EOCR2::SetEnableOffSizeCharacter 1799
EOCR2::GetEnableSecondPassGlobalSegmentation 1799
EOCR2::SetEnableSecondPassGlobalSegmentation 1799
EOCR2::EOCR2 1799
EOCR2::GetClassifierForSymbol 1800
EOCR2::GetGlobalSegmentationRelativeThreshold 1800

EOCR2::SetGlobalSegmentationRelativeThreshold 1800
EOCR2::GetGlobalSegmentationThresholdMode 1801
EOCR2::SetGlobalSegmentationThresholdMode 1801
EOCR2::GetGPUIndexes 1801
EOCR2::SetGPUIndexes 1801
EOCR2::HitTestChar 1801
EOCR2::HitTestLine 1802
EOCR2::HitTestText 1803
EOCR2::HitTestWord 1804
EOCR2::Learn 1805
EOCR2::Load 1806
EOCR2::GetMaxVariation 1806
EOCR2::SetMaxVariation 1806
EOCR2::GetNumDetectionPasses 1807
EOCR2::SetNumDetectionPasses 1807
EOCR2::operator= 1807
EOCR2::Read 1808
EOCR2::GetReadText 1808
EOCR2::Recognize 1808
EOCR2::GetRelativeSpacesWidthRange 1809
EOCR2::SetRelativeSpacesWidthRange 1809
EOCR2::RemoveClassifierForSymbol 1809
EOCR2::GetRepasteObjects 1810
EOCR2::SetRepasteObjects 1810
EOCR2::Save 1810
EOCR2::SaveCharacterDatabase 1811
EOCR2::GetSegmentationMethod 1811
EOCR2::SetSegmentationMethod 1811
EOCR2::GetTextAngleRange 1812
EOCR2::SetTextAngleRange 1812
EOCR2::GetTextPolarity 1812
EOCR2::SetTextPolarity 1812
EOCR2::GetTimeOut 1812
EOCR2::SetTimeOut 1812
EOCR2::GetTopology 1813
EOCR2::SetTopology 1813
4.171. EOCR2Char Class 1813
EOCR2Char::GetBitmap 1814
EOCR2Char::GetBoundingBox 1814
EOCR2Char::GetCandidates 1814
EOCR2Char::EOCR2Char 1815
EOCR2Char::operator= 1815
EOCR2Char::GetText 1815
EOCR2Char::SetText 1815
1816
EOCR2Char::SetTextCode 1816
4.172. EOCR2CharacterCluster Class 1816
EOCR2CharacterCluster::AddCharacter 1816
EOCR2CharacterCluster::GetCharacterCount 1817
EOCR2CharacterCluster::GetCharacters 1817
EOCR2CharacterCluster::Clear 1817
EOCR2CharacterCluster::GetCode 1817
EOCR2CharacterCluster::SetCode 1817
EOCR2CharacterCluster::EOCR2CharacterCluster 1818
EOCR2CharacterCluster::GetCharacter 1818
EOCR2CharacterCluster::operator= 1818
EOCR2CharacterCluster::RemoveCharacter 1819
4.173. EOCR2CharacterDatabase Class 1819
EOCR2CharacterDatabase::AddCharacter 1820

EOCR2CharacterDatabase::AddCharacters 1820
EOCR2CharacterDatabase::AddCluster 1821
EOCR2CharacterDatabase::AddClusters 1821
EOCR2CharacterDatabase::GetCharacters 1822
EOCR2CharacterDatabase::ClearDatabase 1822
EOCR2CharacterDatabase::ClusterDatabase 1822
EOCR2CharacterDatabase::EOCR2CharacterDatabase 1823
EOCR2CharacterDatabase::GetCharacter 1823
EOCR2CharacterDatabase::operator= 1823
EOCR2CharacterDatabase::RemoveCharacter 1824
EOCR2CharacterDatabase::Save 1824
4.174. EOCR2DatabaseCharacter Class 1824
EOCR2DatabaseCharacter::GetBitmap 1825
EOCR2DatabaseCharacter::GetCharacterCode 1825
EOCR2DatabaseCharacter::SetCharacterCode 1825
EOCR2DatabaseCharacter::EOCR2DatabaseCharacter 1825
EOCR2DatabaseCharacter::operator= 1826
4.175. EOCR2Line Class 1826
EOCR2Line::GetBoundingBox 1826
EOCR2Line::EOCR2Line 1827
EOCR2Line::operator= 1827
EOCR2Line::GetText 1827
EOCR2Line::SetText 1827
EOCR2Line::GetWords 1828
EOCR2Line::SetWords 1828
4.176. EOCR2Text Class 1828
EOCR2Text::GetBoundingBox 1828
EOCR2Text::EOCR2Text 1829
EOCR2Text::GetLines 1829
EOCR2Text::SetLines 1829
EOCR2Text::operator= 1829
EOCR2Text::GetText 1830
EOCR2Text::SetText 1830
4.177. EOCR2Word Class 1830
EOCR2Word::GetBoundingBox 1831
EOCR2Word::GetCharacters 1831
EOCR2Word::SetCharacters 1831
EOCR2Word::EOCR2Word 1831
EOCR2Word::operator= 1831
EOCR2Word::GetText 1832
EOCR2Word::SetText 1832
4.178. EPathVector Class 1832
EPathVector::AddElement 1833
EPathVector::GetClosed 1833
EPathVector::SetClosed 1833
EPathVector::Draw 1834
EPathVector::DrawClosedContour 1835
EPathVector::DrawWithCurrentPen 1836
EPathVector::EPathVector 1837
EPathVector::GetElement 1837
EPathVector::operator[] 1838
EPathVector::operator= 1838
EPathVector::GetRawDataPtr 1838
EPathVector::SetElement 1839
4.179. EPatternFinder Class 1839
EPatternFinder::GetAngleBias 1842
EPatternFinder::SetAngleBias 1842
EPatternFinder::GetAngleSearchExtent 1842

EPatternFinder::SetAngleSearchExtent ... 1842
EPatternFinder::GetAngleTolerance ... 1842
EPatternFinder::SetAngleTolerance ... 1842
EPatternFinder::GetContrastMode ... 1843
EPatternFinder::SetContrastMode ... 1843
EPatternFinder::CopyLearntPattern ... 1843
EPatternFinder::DrawModel ... 1843
EPatternFinder::DrawModelWithCurrentPen ... 1844
EPatternFinder::EPatternFinder ... 1845
EPatternFinder::GetFeaturePoints ... 1846
EPatternFinder::Find ... 1846
EPatternFinder::GetFindExtension ... 1846
EPatternFinder::SetFindExtension ... 1846
EPatternFinder::GetInterpolate ... 1847
EPatternFinder::SetInterpolate ... 1847
EPatternFinder::Learn ... 1847
EPatternFinder::GetLearningDone ... 1848
EPatternFinder::GetLightBalance ... 1848
EPatternFinder::SetLightBalance ... 1848
EPatternFinder::Load ... 1849
EPatternFinder::GetLocalSearchMode ... 1850
EPatternFinder::SetLocalSearchMode ... 1850
EPatternFinder::GetMaxFeaturePoints ... 1850
EPatternFinder::SetMaxFeaturePoints ... 1850
EPatternFinder::GetMaxInitialCandidates ... 1850
EPatternFinder::SetMaxInitialCandidates ... 1850
EPatternFinder::GetMaxInstances ... 1851
EPatternFinder::SetMaxInstances ... 1851
EPatternFinder::GetMaxOverlap ... 1851
EPatternFinder::SetMaxOverlap ... 1851
EPatternFinder::GetMinFeaturePoints ... 1852
EPatternFinder::SetMinFeaturePoints ... 1852
EPatternFinder::GetMinScore ... 1852
EPatternFinder::SetMinScore ... 1852
EPatternFinder::operator= ... 1852
EPatternFinder::GetPatternType ... 1853
EPatternFinder::SetPatternType ... 1853
EPatternFinder::GetPivot ... 1853
EPatternFinder::SetPivot ... 1853
EPatternFinder::GetPointShape ... 1853
EPatternFinder::GetReductionMode ... 1854
EPatternFinder::SetReductionMode ... 1854
EPatternFinder::GetReductionStrength ... 1854
EPatternFinder::SetReductionStrength ... 1854
EPatternFinder::Save ... 1855
EPatternFinder::GetScaleBias ... 1855
EPatternFinder::SetScaleBias ... 1855
EPatternFinder::GetScaleSearchExtent ... 1856
EPatternFinder::SetScaleSearchExtent ... 1856
EPatternFinder::GetScaleTolerance ... 1856
EPatternFinder::SetScaleTolerance ... 1856
EPatternFinder::GetThinStructureMode ... 1856
EPatternFinder::SetThinStructureMode ... 1856
EPatternFinder::GetXSearchExtent ... 1857
EPatternFinder::SetXSearchExtent ... 1857
EPatternFinder::GetYSearchExtent ... 1857
EPatternFinder::SetYSearchExtent ... 1857
4.180. EPeakVector Class ... 1857
EPeakVector::AddElement ... 1858

EPeakVector::EPeakVector ... 1858
EPeakVector::GetElement ... 1859
EPeakVector::operator[] ... 1859
EPeakVector::operator= ... 1859
EPeakVector::GetRawDataPtr ... 1860
EPeakVector::SetElement ... 1860
4.181. EPhotometricStereoImager Class ... 1860
EPhotometricStereoImager::CalibrateFromSphere ... 1862
EPhotometricStereoImager::GetCalibrationAzimuthAngles ... 1863
EPhotometricStereoImager::GetCalibrationElevationAngles ... 1864
EPhotometricStereoImager::Compute ... 1866
EPhotometricStereoImager::ComputeGaussianCurvatures ... 1867
EPhotometricStereoImager::ComputeHeightMap ... 1868
EPhotometricStereoImager::ComputeMeanCurvatures ... 1868
EPhotometricStereoImager::ConfigureNonUniformLightingCorrection ... 1869
EPhotometricStereoImager::GetEnableNonUniformLightingCorrection ... 1870
EPhotometricStereoImager::SetEnableNonUniformLightingCorrection ... 1870
EPhotometricStereoImager::EPhotometricStereoImager ... 1870
EPhotometricStereoImager::GetAlbedos ... 1870
EPhotometricStereoImager::GetCalibrationAngles ... 1871
EPhotometricStereoImager::GetGradientsX ... 1873
EPhotometricStereoImager::GetGradientsY ... 1873
EPhotometricStereoImager::Load ... 1873
EPhotometricStereoImager::GetNormals ... 1874
EPhotometricStereoImager::operator= ... 1874
EPhotometricStereoImager::Save ... 1874
EPhotometricStereoImager::SetCalibrationAngles ... 1875
4.182. EPlaneCropper Class ... 1877
EPlaneCropper::Crop ... 1877
EPlaneCropper::EPlaneCropper ... 1878
EPlaneCropper::Load ... 1878
EPlaneCropper::operator= ... 1879
EPlaneCropper::GetPlane ... 1879
EPlaneCropper::SetPlane ... 1879
EPlaneCropper::Save ... 1880
4.183. EPlaneFinder Class ... 1880
EPlaneFinder::DisableDecimator ... 1882
EPlaneFinder::EnableDecimator ... 1882
EPlaneFinder::EPlaneFinder ... 1883
EPlaneFinder::GetExpectedCloudInliersRatio ... 1883
EPlaneFinder::SetExpectedCloudInliersRatio ... 1883
EPlaneFinder::GetExpectedCloudInliersRatioRange ... 1884
EPlaneFinder::SetExpectedCloudInliersRatioRange ... 1884
EPlaneFinder::Find ... 1884
EPlaneFinder::GetNormal ... 1885
EPlaneFinder::GetPoint ... 1885
EPlaneFinder::IsDecimatorEnabled ... 1886
EPlaneFinder::IsNormalSet ... 1886
EPlaneFinder::IsSeedSet ... 1886
EPlaneFinder::Load ... 1886
EPlaneFinder::GetMaxDeviation ... 1887
EPlaneFinder::SetMaxDeviation ... 1887
EPlaneFinder::GetNormalTolerance ... 1887
EPlaneFinder::GetNumberOfPointsAfterDecimation ... 1888
EPlaneFinder::SetNumberOfPointsAfterDecimation ... 1888
EPlaneFinder::GetNumberOfPointsSet ... 1888
1888
EPlaneFinder::SetOnePoint ... 1888
EPlaneFinder::operator= ... 1888

EPlaneFinder::Save ... 1889
EPlaneFinder::GetSeed ... 1889
EPlaneFinder::SetSeed ... 1889
EPlaneFinder::SetNormal ... 1890
EPlaneFinder::SetTwoPoints ... 1891
EPlaneFinder::UnsetNormal ... 1891
EPlaneFinder::UnsetPoints ... 1892
EPlaneFinder::UnsetSeed ... 1892
4.184. EPlaneFitter Class ... 1892
EPlaneFitter::EPlaneFitter ... 1893
EPlaneFitter::Fit ... 1893
EPlaneFitter::Load ... 1893
EPlaneFitter::GetMinSampleCount ... 1894
EPlaneFitter::SetMinSampleCount ... 1894
EPlaneFitter::operator= ... 1894
EPlaneFitter::Save ... 1895
4.185. EPoint Class ... 1895
EPoint::Area ... 1896
EPoint::Argument ... 1897
EPoint::GetCenter ... 1897
EPoint::SetCenter ... 1897
EPoint::CopyTo ... 1897
EPoint::Cross ... 1898
EPoint::Distance ... 1898
EPoint::Dot ... 1899
EPoint::EPoint ... 1899
EPoint::Load ... 1900
EPoint::MidPoint ... 1900
EPoint::Modulus ... 1901
EPoint::operator- ... 1901
EPoint::operator!= ... 1901
EPoint::operator* ... 1902
EPoint::operator/ ... 1902
EPoint::operator+ ... 1902
EPoint::operator= ... 1903
EPoint::operator== ... 1903
EPoint::Project ... 1904
EPoint::Rotate ... 1904
EPoint::Save ... 1904
EPoint::SetCenterXY ... 1905
EPoint::Square ... 1905
EPoint::SquaredDistance ... 1906
EPoint::GetX ... 1906
EPoint::GetY ... 1906
4.186. EPointCloud Class ... 1906
EPointCloud::AddCustomAttributeBuffer ... 1908
EPointCloud::AddPoint ... 1910
EPointCloud::AddPointAndAttributesTo ... 1910
EPointCloud::AddPointCloud ... 1911
EPointCloud::AddPoints ... 1911
EPointCloud::AllocateAttributeBuffer ... 1912
EPointCloud::AllocateCustomAttributeBuffer ... 1913
EPointCloud::Clear ... 1914
EPointCloud::ClearAttributeBuffer ... 1914
EPointCloud::ComputeNormalsAndCurvatures ... 1914
EPointCloud::ComputePlaneBehind ... 1915
EPointCloud::CopyAllAttributesTo ... 1915
EPointCloud::DistanceToLine ... 1916
EPointCloud::DistanceToSegment ... 1917

EPointCloud::GetEnableSpacePartition 1918
EPointCloud::SetEnableSpacePartition 1918
EPointCloud::EPointCloud 1918
EPointCloud::FillAttributeBuffer 1918
EPointCloud::FillPointsBuffer 1920
EPointCloud::GetAttribute 1921
EPointCloud::GetAttributeBuffer 1922
EPointCloud::GetAttributeBufferType 1923
EPointCloud::GetInitializedAttributes 1923
EPointCloud::GetPoint 1923
EPointCloud::HasAttributeBuffer 1924
EPointCloud::Load 1924
EPointCloud::LoadCSV 1925
EPointCloud::LoadOBJ 1925
EPointCloud::LoadPCD 1926
EPointCloud::LoadPLY 1927
EPointCloud::LoadXYZ 1928
EPointCloud::GetNumPoints 1929
EPointCloud::operator= 1929
EPointCloud::GetPointsBuffer 1930
EPointCloud::PrepareSpacePartition 1930
EPointCloud::RemovePoint 1930
EPointCloud::Save 1930
EPointCloud::SaveCSV 1931
EPointCloud::SaveOBJ 1931
EPointCloud::SavePCD 1932
EPointCloud::SavePLY 1932
EPointCloud::SaveXYZ 1933
EPointCloud::SetAttribute 1933
4.187. EPointCloudFactory Class 1935
EPointCloudFactory::CreateCubicPointCloud 1935
EPointCloudFactory::CreateRectangularPointCloud 1936
EPointCloudFactory::CreateSphericPointCloud 1937
4.188. EPointCloudFilter Class 1937
EPointCloudFilter::EPointCloudFilter 1938
EPointCloudFilter::Filter 1939
EPointCloudFilter::GetFilteringMethod 1939
EPointCloudFilter::SetFilteringMethod 1939
EPointCloudFilter::Load 1940
EPointCloudFilter::GetNumberOfNeighbors 1940
EPointCloudFilter::SetNumberOfNeighbors 1940
EPointCloudFilter::operator= 1940
EPointCloudFilter::GetReplaceByAverage 1941
EPointCloudFilter::SetReplaceByAverage 1941
EPointCloudFilter::Save 1941
EPointCloudFilter::GetThresholdMultiplier 1942
EPointCloudFilter::SetThresholdMultiplier 1942
4.189. EPointCloudMerger Class 1942
EPointCloudMerger::Calibrate 1943
EPointCloudMerger::GetEnableDecimation 1943
EPointCloudMerger::SetEnableDecimation 1943
EPointCloudMerger::EPointCloudMerger 1944
EPointCloudMerger::Load 1944
EPointCloudMerger::Merge 1945
EPointCloudMerger::GetMergedCloudResolution 1945
EPointCloudMerger::SetMergedCloudResolution 1945
EPointCloudMerger::operator= 1946
EPointCloudMerger::Save 1946
EPointCloudMerger::GetTransformations 1946

EPointCloudMerger::SetTransformations ... 1946
4.190. EPointCloudStatistics Class ... 1947
EPointCloudStatistics::GetPointCloudBounds ... 1947
EPointCloudStatistics::GetPointCloudCentroid ... 1948
4.191. EPointCloudToMeshConverter Class ... 1949
EPointCloudToMeshConverter::Convert ... 1950
EPointCloudToMeshConverter::EPointCloudToMeshConverter ... 1950
EPointCloudToMeshConverter::Load ... 1951
EPointCloudToMeshConverter::GetMaxEdgeLength ... 1951
EPointCloudToMeshConverter::SetMaxEdgeLength ... 1951
EPointCloudToMeshConverter::operator= ... 1951
EPointCloudToMeshConverter::GetProjectionPlane ... 1952
EPointCloudToMeshConverter::SetProjectionPlane ... 1952
EPointCloudToMeshConverter::GetResolution ... 1952
EPointCloudToMeshConverter::SetResolution ... 1952
EPointCloudToMeshConverter::Save ... 1952
4.192. EPointCloudToZMapConverter Class ... 1953
EPointCloudToZMapConverter::Convert ... 1957
EPointCloudToZMapConverter::EnableFillMode ... 1957
EPointCloudToZMapConverter::EPointCloudToZMapConverter ... 1958
EPointCloudToZMapConverter::GetExtension ... 1958
EPointCloudToZMapConverter::SetExtension ... 1958
EPointCloudToZMapConverter::GetFillUndefinedPixelsDirection ... 1958
EPointCloudToZMapConverter::GetFillUndefinedPixelsMethod ... 1959
EPointCloudToZMapConverter::IsFillModeEnabled ... 1959
EPointCloudToZMapConverter::Load ... 1959
EPointCloudToZMapConverter::GetMapHeight ... 1960
EPointCloudToZMapConverter::GetMapWidth ... 1960
EPointCloudToZMapConverter::GetMapXResolution ... 1960
EPointCloudToZMapConverter::GetMapYResolution ... 1960
EPointCloudToZMapConverter::GetMapZResolution ... 1961
EPointCloudToZMapConverter::SetMapZResolution ... 1961
EPointCloudToZMapConverter::operator= ... 1961
EPointCloudToZMapConverter::operator== ... 1961
EPointCloudToZMapConverter::GetOrientationVector ... 1962
EPointCloudToZMapConverter::SetOrientationVector ... 1962
EPointCloudToZMapConverter::GetOrientationVectorMode ... 1962
EPointCloudToZMapConverter::SetOrientationVectorMode ... 1962
EPointCloudToZMapConverter::GetOrigin ... 1963
EPointCloudToZMapConverter::SetOrigin ... 1963
EPointCloudToZMapConverter::GetReferencePlane ... 1963
EPointCloudToZMapConverter::SetReferencePlane ... 1963
EPointCloudToZMapConverter::GetReferencePlaneMode ... 1963
EPointCloudToZMapConverter::SetReferencePlaneMode ... 1963
EPointCloudToZMapConverter::Save ... 1964
EPointCloudToZMapConverter::SetFillMode ... 1964
EPointCloudToZMapConverter::SetFillModeMedian ... 1965
EPointCloudToZMapConverter::SetMapSize ... 1965
EPointCloudToZMapConverter::SetMapXYResolution ... 1965
EPointCloudToZMapConverter::UnsetMapSize ... 1966
EPointCloudToZMapConverter::UnsetMapXYResolution ... 1966
EPointCloudToZMapConverter::UnsetMapZResolution ... 1967
EPointCloudToZMapConverter::UnsetOrigin ... 1967
EPointCloudToZMapConverter::UnsetWorldToZMapTransform ... 1967
EPointCloudToZMapConverter::GetWorldToZMapTransform ... 1968
EPointCloudToZMapConverter::SetWorldToZMapTransform ... 1968
EPointCloudToZMapConverter::GetZMaptoWorldTransform ... 1968
EPointCloudToZMapConverter::SetZMaptoWorldTransform ... 1968
4.193. EPointGauge Class ... 1969

1971
EPointGauge::SetActive ... 1971
1971
EPointGauge::SetCenter ... 1971
EPointGauge::CopyTo ... 1971
EPointGauge::Drag ... 1972
EPointGauge::Draw ... 1972
EPointGauge::DrawWithCurrentPen ... 1973
EPointGauge::EPointGauge ... 1974
EPointGauge::GetMeasuredPeak ... 1974
EPointGauge::GetMeasuredPoint ... 1975
EPointGauge::HitTest ... 1975
EPointGauge::GetHVConstraint ... 1976
EPointGauge::SetHVConstraint ... 1976
EPointGauge::Measure ... 1976
EPointGauge::GetMinAmplitude ... 1977
EPointGauge::SetMinAmplitude ... 1977
EPointGauge::GetMinArea ... 1977
EPointGauge::SetMinArea ... 1977
EPointGauge::GetNumMeasuredPoints ... 1978
EPointGauge::operator= ... 1978
EPointGauge::Plot ... 1978
EPointGauge::PlotWithCurrentPen ... 1979
EPointGauge::Process ... 1980
EPointGauge::GetRectangularSamplingArea ... 1981
EPointGauge::SetRectangularSamplingArea ... 1981
EPointGauge::SetCenterXY ... 1981
EPointGauge::SetTolerances ... 1982
EPointGauge::GetSmoothing ... 1982
EPointGauge::SetSmoothing ... 1982
EPointGauge::GetThickness ... 1983
EPointGauge::SetThickness ... 1983
EPointGauge::GetThreshold ... 1983
EPointGauge::SetThreshold ... 1983
EPointGauge::GetTolerance ... 1984
EPointGauge::SetTolerance ... 1984
EPointGauge::GetToleranceAngle ... 1984
EPointGauge::SetToleranceAngle ... 1984
EPointGauge::GetTransitionChoice ... 1984
EPointGauge::SetTransitionChoice ... 1984
EPointGauge::GetTransitionIndex ... 1985
EPointGauge::SetTransitionIndex ... 1985
EPointGauge::GetTransitionType ... 1985
EPointGauge::SetTransitionType ... 1985
EPointGauge::GetType ... 1986
EPointGauge::GetValid ... 1986
4.194. EPointShape Class ... 1986
EPointShape::GetCenter ... 1987
EPointShape::SetCenter ... 1987
EPointShape::GetCenterX ... 1987
EPointShape::GetCenterY ... 1987
EPointShape::Closest ... 1988
EPointShape::CopyTo ... 1988
EPointShape::Drag ... 1989
EPointShape::Draw ... 1989
EPointShape::DrawWithCurrentPen ... 1990
EPointShape::HitTest ... 1990
EPointShape::operator!= ... 1991
EPointShape::operator= ... 1991

EPointShape::operator== 1991
EPointShape::SetCenterXY 1992
EPointShape::GetType 1992
4.195. EPolygon Class 1992
EPolygon::AppendVertex 1993
EPolygon::GetArea 1994
EPolygon::CopyTo 1994
EPolygon::EPolygon 1994
EPolygon::GetVertex 1995
EPolygon::InsertVertex 1995
EPolygon::GetIsClosed 1996
EPolygon::SetIsClosed 1996
EPolygon::IsValid 1996
EPolygon::GetLength 1996
EPolygon::Load 1996
EPolygon::GetNumEdges 1997
EPolygon::GetNumVertices 1997
EPolygon::operator!= 1997
EPolygon::operator= 1998
EPolygon::operator== 1998
EPolygon::RemoveVertex 1998
EPolygon::Save 1999
EPolygon::SetVertex 1999
EPolygon::GetVertices 1999
4.196. EPolygonGauge Class 2000
2002
EPolygonGauge::SetActive 2002
EPolygonGauge::AddSkipRange 2002
EPolygonGauge::GetAverageDistance 2003
EPolygonGauge::CopyTo 2003
EPolygonGauge::Drag 2004
EPolygonGauge::Draw 2004
EPolygonGauge::DrawWithCurrentPen 2005
EPolygonGauge::GetEnableScaling 2006
EPolygonGauge::SetEnableScaling 2006
EPolygonGauge::EPolygonGauge 2006
EPolygonGauge::GetFilteringThreshold 2007
EPolygonGauge::SetFilteringThreshold 2007
EPolygonGauge::GetSkipRange 2007
EPolygonGauge::HitTest 2008
EPolygonGauge::GetHVConstraint 2008
EPolygonGauge::SetHVConstraint 2008
EPolygonGauge::Measure 2008
EPolygonGauge::GetMeasuredPolygon 2009
EPolygonGauge::GetMeasurementMode 2009
EPolygonGauge::SetMeasurementMode 2009
EPolygonGauge::MeasureSample 2010
EPolygonGauge::GetMinAmplitude 2010
EPolygonGauge::SetMinAmplitude 2010
EPolygonGauge::GetMinArea 2011
EPolygonGauge::SetMinArea 2011
EPolygonGauge::GetMinNumFitSamples 2011
EPolygonGauge::SetMinNumFitSamples 2011
EPolygonGauge::GetNumFilteringPasses 2012
EPolygonGauge::SetNumFilteringPasses 2012
EPolygonGauge::GetNumSamples 2012
EPolygonGauge::GetNumSkipRanges 2012
EPolygonGauge::GetNumValidSamples 2013
EPolygonGauge::operator= 2013

EPolygonGauge::Process ... 2013
EPolygonGauge::RemoveAllSkipRanges ... 2014
EPolygonGauge::RemoveSkipRange ... 2014
EPolygonGauge::GetSamplingStep ... 2014
EPolygonGauge::SetSamplingStep ... 2014
EPolygonGauge::GetSmoothing ... 2015
EPolygonGauge::SetSmoothing ... 2015
EPolygonGauge::GetThickness ... 2015
EPolygonGauge::SetThickness ... 2015
EPolygonGauge::GetThreshold ... 2016
EPolygonGauge::SetThreshold ... 2016
EPolygonGauge::GetTolerance ... 2016
EPolygonGauge::SetTolerance ... 2016
EPolygonGauge::GetTransitionChoice ... 2017
EPolygonGauge::SetTransitionChoice ... 2017
EPolygonGauge::GetTransitionIndex ... 2017
EPolygonGauge::SetTransitionIndex ... 2017
EPolygonGauge::GetTransitionType ... 2017
EPolygonGauge::SetTransitionType ... 2017
EPolygonGauge::GetType ... 2018
4.197. EPolygonRegion Class ... 2018
EPolygonRegion::AddPoint ... 2019
EPolygonRegion::Drag ... 2019
EPolygonRegion::EPolygonRegion ... 2020
EPolygonRegion::HitTest ... 2020
EPolygonRegion::InsertPoint ... 2021
EPolygonRegion::Load ... 2022
EPolygonRegion::GetNumPoints ... 2022
EPolygonRegion::operator!= ... 2023
EPolygonRegion::operator= ... 2023
EPolygonRegion::operator== ... 2023
EPolygonRegion::GetPoints ... 2024
EPolygonRegion::SetPoints ... 2024
EPolygonRegion::RemovePoint ... 2024
EPolygonRegion::Rotate ... 2025
EPolygonRegion::Save ... 2025
EPolygonRegion::Scale ... 2025
EPolygonRegion::Translate ... 2026
4.198. EPolygonShape Class ... 2026
EPolygonShape::AddVertexAtDisplayPosition ... 2028
EPolygonShape::GetAngle ... 2028
EPolygonShape::SetAngle ... 2028
EPolygonShape::AppendVertex ... 2028
EPolygonShape::GetCenter ... 2029
EPolygonShape::SetCenter ... 2029
EPolygonShape::GetCenterX ... 2029
EPolygonShape::GetCenterY ... 2029
EPolygonShape::Closest ... 2030
EPolygonShape::CopyTo ... 2030
EPolygonShape::DisplayToLocalPosition ... 2030
EPolygonShape::Drag ... 2031
EPolygonShape::Draw ... 2031
EPolygonShape::DrawWithCurrentPen ... 2032
EPolygonShape::EPolygonShape ... 2033
EPolygonShape::GetFormat ... 2033
EPolygonShape::GetVertex ... 2033
EPolygonShape::HitTest ... 2034
EPolygonShape::GetIsClosed ... 2034
EPolygonShape::SetIsClosed ... 2034

EPolygonShape::GetLength 2034
EPolygonShape::LocalToDisplayPosition 2035
EPolygonShape::GetNumEdges 2035
EPolygonShape::GetNumVertices 2035
EPolygonShape::operator= 2035
EPolygonShape::GetPolygon 2036
EPolygonShape::SetPolygon 2036
EPolygonShape::RemoveVertexAtDisplayPosition 2036
EPolygonShape::GetScale 2037
EPolygonShape::SetScale 2037
EPolygonShape::SerializeData 2037
EPolygonShape::SetCenterXY 2037
EPolygonShape::SetVertex 2038
EPolygonShape::GetType 2038
4.199. EPrincipalAxisExtractor Class 2038
EPrincipalAxisExtractor::EPrincipalAxisExtractor 2039
EPrincipalAxisExtractor::Extract 2040
EPrincipalAxisExtractor::HasReferenceTransformSet 2040
EPrincipalAxisExtractor::Load 2040
EPrincipalAxisExtractor::operator= 2041
EPrincipalAxisExtractor::GetReferenceTransform 2041
EPrincipalAxisExtractor::SetReferenceTransform 2041
EPrincipalAxisExtractor::Save 2042
EPrincipalAxisExtractor::UnsetReferenceTransform 2042
4.200. EPseudoColorLookup Class 2042
EPseudoColorLookup::EPseudoColorLookup 2043
EPseudoColorLookup::SetShading 2043
4.201. EQRCode Class 2044
EQRCode::GetDecodedStream 2044
EQRCode::Draw 2045
EQRCode::DrawErrors 2046
EQRCode::DrawErrorsWithCurrentPen 2046
EQRCode::DrawWithCurrentPen 2047
EQRCode::EQRCode 2048
EQRCode::GetErrors 2048
EQRCode::GetGeometry 2048
EQRCode::GetCellPosition 2049
EQRCode::GetDecodedString 2049
EQRCode::GetIsDecodingReliable 2050
EQRCode::IsGS1 2050
EQRCode::GetIso15415GradingParameters 2050
EQRCode::GetIso29158GradingParameters 2051
EQRCode::GetLevel 2051
EQRCode::GetModel 2051
EQRCode::operator= 2051
EQRCode::GetUnusedErrorCorrection 2052
EQRCode::GetVersion 2052
4.202. EQRCodeDecodedStream Class 2052
EQRCodeDecodedStream::GetApplicationIndicator 2053
EQRCodeDecodedStream::GetCodingMode 2053
EQRCodeDecodedStream::GetDecodedStreamParts 2053
EQRCodeDecodedStream::EQRCodeDecodedStream 2054
EQRCodeDecodedStream::operator= 2054
EQRCodeDecodedStream::GetRawBitstream 2054
4.203. EQRCodeDecodedStreamPart Class 2055
EQRCodeDecodedStreamPart::GetDecodedData 2055
EQRCodeDecodedStreamPart::GetECITableIndicator 2055
EQRCodeDecodedStreamPart::GetEncoding 2056

EQRCodeDecodedStreamPart::EQRCodeDecodedStreamPart 2056
EQRCodeDecodedStreamPart::GetDecodedString 2056
EQRCodeDecodedStreamPart::operator= 2057
4.204. EQRCodeGeometry Class 2057
EQRCodeGeometry::Draw 2058
EQRCodeGeometry::DrawWithCurrentPen 2059
EQRCodeGeometry::EQRCodeGeometry 2059
EQRCodeGeometry::GetFinderPatternCenters 2060
EQRCodeGeometry::operator= 2060
EQRCodeGeometry::GetPosition 2061
4.205. EQRCodeGrid Class 2061
2061
EQRCodeGrid::SetEnableAll 2061
EQRCodeGrid::EQRCodeGrid 2062
EQRCodeGrid::GetCellEnabled 2062
EQRCodeGrid::GetResults 2063
EQRCodeGrid::GetNumCols 2063
EQRCodeGrid::GetNumRows 2063
EQRCodeGrid::operator= 2064
EQRCodeGrid::SetEnableCell 2064
EQRCodeGrid::SetEnableColumn 2065
EQRCodeGrid::SetEnableRow 2065
4.206. EQRCodeReader Class 2066
EQRCodeReader::GetCellPolarityConfidenceThreshold 2068
EQRCodeReader::SetCellPolarityConfidenceThreshold 2068
EQRCodeReader::GetComputeGrading 2068
EQRCodeReader::SetComputeGrading 2068
EQRCodeReader::GetDetectionMethod 2068
EQRCodeReader::SetDetectionMethod 2068
EQRCodeReader::GetDetectionTradeOff 2069
EQRCodeReader::SetDetectionTradeOff 2069
EQRCodeReader::EQRCodeReader 2069
EQRCodeReader::GetFilterOutUnreliablyDecodedQRCodes 2070
EQRCodeReader::SetFilterOutUnreliablyDecodedQRCodes 2070
EQRCodeReader::GetForegroundDetectionThreshold 2070
EQRCodeReader::SetForegroundDetectionThreshold 2070
EQRCodeReader::Load 2070
EQRCodeReader::GetMaximumVersion 2071
EQRCodeReader::SetMaximumVersion 2071
EQRCodeReader::GetMaxNumCodes 2071
EQRCodeReader::SetMaxNumCodes 2071
EQRCodeReader::GetMinimumIsotropy 2072
EQRCodeReader::SetMinimumIsotropy 2072
EQRCodeReader::GetMinimumScore 2072
EQRCodeReader::SetMinimumScore 2072
EQRCodeReader::GetMinimumVersion 2072
EQRCodeReader::SetMinimumVersion 2072
EQRCodeReader::operator= 2073
EQRCodeReader::Read 2073
EQRCodeReader::Save 2074
EQRCodeReader::GetScanPrecision 2075
EQRCodeReader::SetScanPrecision 2075
EQRCodeReader::GetSearchedModels 2075
EQRCodeReader::SetSearchedModels 2075
2076
EQRCodeReader::SetSearchField 2076
EQRCodeReader::GetTimeOut 2076
EQRCodeReader::SetTimeOut 2076
4.207. EQuadrangle Class 2076

EQuadrangle::GetArea ... 2077
EQuadrangle::GetCorners ... 2077
EQuadrangle::Draw ... 2078
EQuadrangle::DrawWithCurrentPen ... 2079
EQuadrangle::EQuadrangle ... 2080
EQuadrangle::GetCornerAngle ... 2080
EQuadrangle::GetPoint ... 2080
EQuadrangle::GetSideAngle ... 2081
EQuadrangle::GetSideLength ... 2081
EQuadrangle::GetGravityCenter ... 2081
EQuadrangle::IsPointInside ... 2082
EQuadrangle::IsQuadrangleInside ... 2082
EQuadrangle::operator= ... 2082
EQuadrangle::OverLaps ... 2083
EQuadrangle::GetPerimeter ... 2083
EQuadrangle::SetPoint ... 2083
EQuadrangle::GetUpperLeftCorner ... 2084
4.208. ERandomDecimator Class ... 2084
ERandomDecimator::Decimate ... 2084
ERandomDecimator::ERandomDecimator ... 2085
ERandomDecimator::GetNumberOfPoints ... 2085
ERandomDecimator::SetNumberOfPoints ... 2085
ERandomDecimator::operator!= ... 2086
ERandomDecimator::operator= ... 2086
ERandomDecimator::operator== ... 2086
4.209. ERectangle Class ... 2087
ERectangle::CopyTo ... 2087
ERectangle::ERectangle ... 2088
ERectangle::GetCorners ... 2089
ERectangle::GetEdges ... 2090
ERectangle::GetMidEdges ... 2090
ERectangle::GetPoint ... 2091
ERectangle::operator= ... 2091
ERectangle::SetFromOppositeCorners ... 2091
ERectangle::SetFromOriginMiddleEnd ... 2092
ERectangle::SetFromThreeCorners ... 2092
ERectangle::SetFromTwoPoints ... 2093
ERectangle::SetSize ... 2093
ERectangle::GetSizeX ... 2094
ERectangle::GetSizeY ... 2094
4.210. ERectangleGauge Class ... 2094
2098
ERectangleGauge::SetActive ... 2098
ERectangleGauge::GetActiveEdges ... 2098
ERectangleGauge::SetActiveEdges ... 2098
ERectangleGauge::AddSkipRange ... 2098
ERectangleGauge::GetAverageDistance ... 2099
ERectangleGauge::CopyTo ... 2099
ERectangleGauge::DisableInnerFiltering ... 2100
ERectangleGauge::Drag ... 2100
ERectangleGauge::Draw ... 2101
ERectangleGauge::DrawWithCurrentPen ... 2102
ERectangleGauge::ERectangleGauge ... 2102
ERectangleGauge::GetFilteringThreshold ... 2103
ERectangleGauge::SetFilteringThreshold ... 2103
ERectangleGauge::GetMeasuredPoint ... 2103
ERectangleGauge::GetMinNumFitSamples ... 2104
ERectangleGauge::GetSampleLeftEdge ... 2104
ERectangleGauge::GetSampleLowerEdge ... 2105

ERectangleGauge::GetSampleRightEdge 2106
ERectangleGauge::GetSampleUpperEdge 2107
ERectangleGauge::GetSampleX 2107
ERectangleGauge::GetSampleX 2108
ERectangleGauge::GetSampleY 2109
ERectangleGauge::GetSampleY 2109
ERectangleGauge::GetSkipRange 2110
ERectangleGauge::HitTest 2111
ERectangleGauge::GetHVConstraint 2111
ERectangleGauge::SetHVConstraint 2111
ERectangleGauge::GetInnerFilteringEnabled 2112
ERectangleGauge::GetInnerFilteringThreshold 2112
ERectangleGauge::SetInnerFilteringThreshold 2112
ERectangleGauge::GetKnownAngle 2112
ERectangleGauge::SetKnownAngle 2112
ERectangleGauge::Measure 2113
ERectangleGauge::GetMeasuredRectangle 2113
ERectangleGauge::MeasureSample 2114
ERectangleGauge::MeasureWithoutFitting 2114
ERectangleGauge::GetMinAmplitude 2115
ERectangleGauge::SetMinAmplitude 2115
ERectangleGauge::GetMinArea 2116
ERectangleGauge::SetMinArea 2116
ERectangleGauge::GetNumFilteringPasses 2116
ERectangleGauge::SetNumFilteringPasses 2116
ERectangleGauge::GetNumSamples 2116
ERectangleGauge::GetNumSamplesLeftEdge 2117
ERectangleGauge::GetNumSamplesLowerEdge 2117
ERectangleGauge::GetNumSamplesRightEdge 2117
ERectangleGauge::GetNumSamplesUpperEdge 2117
ERectangleGauge::GetNumSamplesX 2118
ERectangleGauge::GetNumSamplesX 2118
ERectangleGauge::GetNumSamplesY 2118
ERectangleGauge::GetNumSamplesY 2118
ERectangleGauge::GetNumSkipRanges 2119
ERectangleGauge::GetNumValidSamples 2119
ERectangleGauge::operator= 2119
ERectangleGauge::Plot 2120
ERectangleGauge::PlotWithCurrentPen 2121
ERectangleGauge::Process 2122
ERectangleGauge::GetRectangularSamplingArea 2122
ERectangleGauge::SetRectangularSamplingArea 2122
ERectangleGauge::RemoveAllSkipRanges 2123
ERectangleGauge::RemoveSkipRange 2123
ERectangleGauge::GetSamplingStep 2123
ERectangleGauge::SetSamplingStep 2123
ERectangleGauge::SetMinNumFitSamples 2124
ERectangleGauge::GetSmoothing 2125
ERectangleGauge::SetSmoothing 2125
ERectangleGauge::GetThickness 2125
ERectangleGauge::SetThickness 2125
ERectangleGauge::GetThreshold 2125
ERectangleGauge::SetThreshold 2125
ERectangleGauge::GetTolerance 2126
ERectangleGauge::SetTolerance 2126
ERectangleGauge::GetTransitionChoice 2126
ERectangleGauge::SetTransitionChoice 2126
ERectangleGauge::GetTransitionIndex 2127
ERectangleGauge::SetTransitionIndex 2127

ERectangleGauge::GetTransitionType 2127
ERectangleGauge::SetTransitionType 2127
ERectangleGauge::GetType 2128
ERectangleGauge::GetValid 2128
4.211. ERectangleRegion Class 2128
ERectangleRegion::GetAngle 2129
ERectangleRegion::SetAngle 2129
ERectangleRegion::GetCenter 2129
ERectangleRegion::SetCenter 2129
ERectangleRegion::GetCorners 2130
ERectangleRegion::Drag 2130
ERectangleRegion::ERectangleRegion 2131
ERectangleRegion::GetHeight 2132
ERectangleRegion::SetHeight 2132
ERectangleRegion::HitTest 2132
ERectangleRegion::Load 2133
ERectangleRegion::operator!= 2134
ERectangleRegion::operator= 2134
ERectangleRegion::operator== 2134
ERectangleRegion::Rotate 2135
ERectangleRegion::Save 2135
ERectangleRegion::Scale 2135
ERectangleRegion::Translate 2136
ERectangleRegion::GetWidth 2136
ERectangleRegion::SetWidth 2136
4.212. ERectangleShape Class 2137
ERectangleShape::GetAngle 2138
ERectangleShape::SetAngle 2138
ERectangleShape::GetCenter 2138
ERectangleShape::SetCenter 2138
ERectangleShape::GetCenterX 2139
ERectangleShape::GetCenterY 2139
ERectangleShape::Closest 2139
ERectangleShape::CopyTo 2139
ERectangleShape::Drag 2140
ERectangleShape::Draw 2140
ERectangleShape::DrawWithCurrentPen 2141
ERectangleShape::GetCorners 2142
ERectangleShape::GetEdges 2142
ERectangleShape::GetMidEdges 2143
ERectangleShape::GetPoint 2144
ERectangleShape::HitTest 2144
ERectangleShape::operator= 2144
2145
ERectangleShape::SetRectangle 2145
ERectangleShape::GetScale 2145
ERectangleShape::SetScale 2145
ERectangleShape::SetCenterXY 2145
ERectangleShape::SetFromOppositeCorners 2146
ERectangleShape::SetFromOriginMiddleEnd 2146
ERectangleShape::SetFromThreeCorners 2147
ERectangleShape::SetFromTwoPoints 2147
ERectangleShape::SetSize 2148
ERectangleShape::GetSizeX 2148
ERectangleShape::GetSizeY 2149
ERectangleShape::GetType 2149
4.213. ERectangularCropper Class 2149
ERectangularCropper::Crop 2149
ERectangularCropper::ERectangularCropper 2150

ERectangularCropper::operator= 2151
ERectangularCropper::operator== 2151
4.214. EReferenceImageSegmenter Class 2151
2152
EReferenceImageSegmenter::SetBlackLayerEncoded 2152
2153
EReferenceImageSegmenter::SetBlackLayerIndex 2153
EReferenceImageSegmenter::Load 2153
EReferenceImageSegmenter::operator== 2153
EReferenceImageSegmenter::GetReferenceImageBW16 2154
EReferenceImageSegmenter::SetReferenceImageBW16 2154
EReferenceImageSegmenter::GetReferenceImageBW8 2154
EReferenceImageSegmenter::SetReferenceImageBW8 2154
EReferenceImageSegmenter::GetReferenceImageC24 2155
EReferenceImageSegmenter::SetReferenceImageC24 2155
EReferenceImageSegmenter::Save 2155
2155
EReferenceImageSegmenter::SetWhiteLayerEncoded 2155
2156
EReferenceImageSegmenter::SetWhiteLayerIndex 2156
4.215. ERegion Class 2156
ERegion::GetArea 2157
ERegion::GetBoundingBoxHeight 2157
ERegion::GetBoundingBoxOrgX 2158
ERegion::GetBoundingBoxOrgY 2158
ERegion::GetBoundingBoxWidth 2158
ERegion::Contour 2159
ERegion::CropRuns 2159
ERegion::Drag 2160
ERegion::Draw 2160
ERegion::DrawContour 2162
ERegion::DrawContourWithCurrentPen 2163
ERegion::DrawHandles 2164
ERegion::DrawHandlesWithCurrentPen 2165
ERegion::DrawWithCurrentPen 2166
ERegion::GetEditionMode 2167
ERegion::SetEditionMode 2167
ERegion::ERegion 2167
ERegion::Grow 2168
ERegion::HitTest 2168
ERegion::Intersection 2169
ERegion::IsPointInRegion 2169
ERegion::Load 2170
ERegion::operator!= 2170
ERegion::operator= 2171
ERegion::operator== 2171
ERegion::Prepare 2171
ERegion::GetRuns 2173
ERegion::SetRuns 2173
ERegion::Save 2173
ERegion::Shrink 2173
ERegion::Subtraction 2174
ERegion::ToImage 2174
ERegion::TranslateRuns 2175
ERegion::Union 2175
4.216. ERegionFreeHandPainter Class 2176
2176
ERegionFreeHandPainter::SetBrush 2176
ERegionFreeHandPainter::ClearCanvas 2177

ERegionFreeHandPainter::Draw 2177
ERegionFreeHandPainter::DrawContour 2178
ERegionFreeHandPainter::ERegionFreeHandPainter 2179
ERegionFreeHandPainter::Paint 2180
ERegionFreeHandPainter::RetrieveRegion 2180
ERegionFreeHandPainter::SetCanvasSize 2181
4.217. EROIBW1 Class 2181
EROIBW1::EROIBW1 2182
EROIBW1::GetFirstSubROI 2182
EROIBW1::GetBitIndex 2182
EROIBW1::GetNextROI 2183
EROIBW1::GetPixel 2183
EROIBW1::GetNextSiblingROI 2184
EROIBW1::operator= 2184
EROIBW1::GetParent 2184
EROIBW1::SetPixel 2185
EROIBW1::GetTopParent 2185
4.218. EROIBW16 Class 2186
EROIBW16::EROIBW16 2186
EROIBW16::GetFirstSubROI 2186
EROIBW16::GetNextROI 2187
EROIBW16::GetPixel 2187
EROIBW16::GetNextSiblingROI 2188
EROIBW16::operator= 2188
EROIBW16::GetParent 2188
EROIBW16::SetPixel 2188
EROIBW16::GetTopParent 2189
4.219. EROIBW32 Class 2189
EROIBW32::EROIBW32 2190
EROIBW32::GetFirstSubROI 2190
EROIBW32::GetNextROI 2190
EROIBW32::GetPixel 2191
EROIBW32::GetNextSiblingROI 2191
EROIBW32::operator= 2192
EROIBW32::GetParent 2192
EROIBW32::SetPixel 2192
EROIBW32::GetTopParent 2193
4.220. EROIBW32f Class 2193
EROIBW32f::Draw 2194
EROIBW32f::DrawFrame 2196
EROIBW32f::DrawFrameWithCurrentPen 2197
EROIBW32f::EROIBW32f 2199
EROIBW32f::GetFirstSubROI 2199
EROIBW32f::GetNextROI 2199
EROIBW32f::GetPixel 2199
EROIBW32f::GetNextSiblingROI 2200
EROIBW32f::operator= 2200
EROIBW32f::GetParent 2201
EROIBW32f::Save 2201
EROIBW32f::SaveJpeg 2201
EROIBW32f::SaveJpeg2K 2202
EROIBW32f::SavePng 2202
EROIBW32f::SetPixel 2203
EROIBW32f::GetTopParent 2203
4.221. EROIBW8 Class 2204
EROIBW8::EROIBW8 2204
EROIBW8::GetFirstSubROI 2204
EROIBW8::GetNextROI 2205

EROIBW8::GetPixel 2205
EROIBW8::GetNextSiblingROI 2206
EROIBW8::operator= 2206
EROIBW8::GetParent 2206
EROIBW8::SetPixel 2206
EROIBW8::GetTopParent 2207
4.222. EROIC15 Class 2207
EROIC15::EROIC15 2208
EROIC15::GetFirstSubROI 2208
EROIC15::GetNextROI 2208
EROIC15::GetPixel 2209
EROIC15::GetNextSiblingROI 2209
EROIC15::operator= 2210
EROIC15::GetParent 2210
EROIC15::SetPixel 2210
EROIC15::GetTopParent 2211
4.223. EROIC16 Class 2211
EROIC16::EROIC16 2212
EROIC16::GetFirstSubROI 2212
EROIC16::GetNextROI 2212
EROIC16::GetPixel 2213
EROIC16::GetNextSiblingROI 2213
EROIC16::operator= 2213
EROIC16::GetParent 2214
EROIC16::SetPixel 2214
EROIC16::GetTopParent 2214
4.224. EROIC24 Class 2215
EROIC24::EROIC24 2215
EROIC24::GetFirstSubROI 2216
EROIC24::GetNextROI 2216
EROIC24::GetPixel 2216
EROIC24::GetNextSiblingROI 2217
EROIC24::operator= 2217
EROIC24::GetParent 2217
EROIC24::SetPixel 2218
EROIC24::GetTopParent 2218
4.225. EROIC24A Class 2219
EROIC24A::EROIC24A 2219
EROIC24A::GetFirstSubROI 2219
EROIC24A::GetNextROI 2220
EROIC24A::GetPixel 2220
EROIC24A::GetNextSiblingROI 2221
EROIC24A::operator= 2221
EROIC24A::GetParent 2221
EROIC24A::SetPixel 2221
EROIC24A::GetTopParent 2222
4.226. EROIC48 Class 2222
EROIC48::EROIC48 2223
EROIC48::GetFirstSubROI 2223
EROIC48::GetNextROI 2223
EROIC48::GetPixel 2224
EROIC48::GetNextSiblingROI 2224
EROIC48::operator= 2225
EROIC48::GetParent 2225
EROIC48::SetPixel 2225
EROIC48::GetTopParent 2226
4.227. ERotatedBoundingBox Class 2226
ERotatedBoundingBox::GetAngle 2227

ERotatedBoundingBox::GetCenter 2227
ERotatedBoundingBox::GetCenterX 2227
ERotatedBoundingBox::GetCenterY 2228
ERotatedBoundingBox::Draw 2228
ERotatedBoundingBox::DrawWithCurrentPen 2229
ERotatedBoundingBox::ERotatedBoundingBox 2230
ERotatedBoundingBox::GetHeight 2231
ERotatedBoundingBox::LocalToGlobalBox 2231
ERotatedBoundingBox::LocalToGlobalPoint 2231
ERotatedBoundingBox::GetMajorAxis 2231
ERotatedBoundingBox::operator= 2232
ERotatedBoundingBox::operator== 2232
ERotatedBoundingBox::GetQuadrangle 2232
ERotatedBoundingBox::Rotate 2233
ERotatedBoundingBox::Scale 2233
ERotatedBoundingBox::Translate 2233
ERotatedBoundingBox::GetUpperLeftCorner 2234
ERotatedBoundingBox::GetWidth 2234
4.228. ESamplePoint Class 2234
ESamplePoint::ESamplePoint 2235
ESamplePoint::GetIsOutlier 2235
ESamplePoint::GetIsValid 2235
ESamplePoint::operator= 2235
ESamplePoint::GetPosition 2236
4.229. EScaleCalibrationModel Class 2236
EScaleCalibrationModel::EScaleCalibrationModel 2237
EScaleCalibrationModel::GetFactorX 2237
EScaleCalibrationModel::GetFactorY 2237
EScaleCalibrationModel::GetFactorZ 2238
EScaleCalibrationModel::Load 2238
EScaleCalibrationModel::operator= 2238
EScaleCalibrationModel::operator== 2239
EScaleCalibrationModel::Save 2239
EScaleCalibrationModel::GetType 2239
4.230. ESearchParamsType Class 2240
ESearchParamsType::AddContrast 2241
ESearchParamsType::AddFamily 2241
ESearchParamsType::AddFlipping 2242
ESearchParamsType::AddLogicalSize 2242
ESearchParamsType::ClearContrast 2242
ESearchParamsType::ClearFamily 2243
ESearchParamsType::ClearFlipping 2243
ESearchParamsType::ClearLogicalSize 2243
ESearchParamsType::GetContrastCount 2244
ESearchParamsType::GetFamilyCount 2244
ESearchParamsType::GetFlippingCount 2244
ESearchParamsType::GetContrast 2244
ESearchParamsType::GetFamily 2245
ESearchParamsType::GetFlipping 2245
ESearchParamsType::GetLogicalSize 2245
ESearchParamsType::GetLogicalSizeCount 2246
ESearchParamsType::RemoveContrast 2246
ESearchParamsType::RemoveFamily 2246
ESearchParamsType::RemoveFlipping 2247
ESearchParamsType::RemoveLogicalSize 2247
4.231. ESerializer Class 2248
ESerializer::Close 2248
ESerializer::CreateFileReader 2248
ESerializer::CreateFileWriter 2249

ESerializer::CreateMemoryReader 2249
ESerializer::CreateMemoryWriter 2250
ESerializer::GetWriting 2251
4.232. EShape Class 2251
EShape::GetActive 2254
EShape::SetActive 2254
2254
EShape::SetActiveRecursive 2254
EShape::GetActualShape 2254
EShape::SetActualShape 2254
2255
EShape::SetActualShapeRecursive 2255
EShape::Attach 2255
EShape::Closest 2255
EShape::GetClosestShape 2256
EShape::Detach 2256
EShape::DetachDaughters 2256
EShape::DisableBehaviorFilter 2256
EShape::DisableTypeFilter 2257
EShape::Drag 2257
EShape::GetDragable 2257
EShape::SetDragable 2257
2258
EShape::SetDragableRecursive 2258
EShape::Draw 2258
EShape::DrawWithCurrentPen 2259
EShape::EnableBehaviorFilter 2260
EShape::EnableTypeFilter 2260
EShape::GetAllocated 2261
EShape::GetDaughter 2261
EShape::GetDraggingMode 2261
EShape::GetFound 2261
EShape::GetProperty 2262
EShape::GetShapeNamed 2262
EShape::HasProperty 2262
EShape::HasValidPolarityProperty 2263
EShape::GetHitHandle 2263
EShape::GetHitShape 2263
EShape::HitTest 2264
EShape::InvalidateWorld 2264
EShape::GetLabeled 2264
EShape::SetLabeled 2264
2264
EShape::SetLabeledRecursive 2264
EShape::Load 2265
EShape::LocalToSensor 2265
EShape::GetMother 2266
EShape::GetName 2266
EShape::SetName 2266
EShape::GetNumDaughters 2266
EShape::GetPanX 2266
EShape::GetPanY 2267
EShape::RemoveProperty 2267
EShape::GetResizable 2267
EShape::SetResizable 2267
2267
EShape::SetResizableRecursive 2267
EShape::GetRotatable 2268
EShape::SetRotatable 2268

2268
EShape::SetRotatableRecursive 2268
EShape::Save 2268
EShape::GetSelectable 2269
EShape::SetSelectable 2269
2269
EShape::SetSelectableRecursive 2269
EShape::GetSelected 2269
EShape::SetSelected 2269
2270
EShape::SetSelectedRecursive 2270
EShape::SensorToLocal 2270
EShape::SetAllocated 2270
EShape::SetCursor 2271
EShape::SetDraggingMode 2271
EShape::SetPan 2271
EShape::SetProperty 2272
EShape::SetPropertyRecursive 2272
EShape::SetZoom 2273
EShape::GetType 2273
EShape::GetVisible 2274
EShape::SetVisible 2274
2274
EShape::SetVisibleRecursive 2274
EShape::GetWorldShape 2274
EShape::GetZoomX 2274
EShape::GetZoomY 2274
4.233. ESimpleCropper Class 2275
ESimpleCropper::Crop 2275
ESimpleCropper::ESimpleCropper 2276
ESimpleCropper::operator= 2276
ESimpleCropper::operator== 2277
2277
ESimpleCropper::SetXRange 2277
2277
ESimpleCropper::SetYRange 2277
2278
ESimpleCropper::SetZRange 2278
4.234. ESphereFitter Class 2278
ESphereFitter::ESphereFitter 2278
ESphereFitter::Fit 2279
ESphereFitter::Load 2279
ESphereFitter::operator= 2279
ESphereFitter::Save 2280
4.235. ESphericalCropper Class 2280
ESphericalCropper::Crop 2281
ESphericalCropper::ESphericalCropper 2281
ESphericalCropper::operator= 2282
ESphericalCropper::operator== 2282
4.236. ESpot Class 2282
ESpot::GetArea 2283
ESpot::GetBox 2283
ESpot::GetCenter 2284
ESpot::GetDLLabel 2284
ESpot::GetDLProbability 2284
ESpot::Draw 2284
ESpot::ESpot 2285
ESpot::GetHeight 2286
ESpot::Load 2286

ESpot::operator!= 2287
ESpot::operator= 2287
ESpot::operator== 2287
ESpot::GetPolarity 2288
ESpot::GetRegion 2288
ESpot::GetROI 2288
ESpot::Save 2288
ESpot::GetStrength 2289
ESpot::GetType 2289
ESpot::GetWidth 2289
4.237. ESpotDetector Class 2289
ESpotDetector::AddDetectedSpotsToDLProject 2292
ESpotDetector::GetAlignmentArea 2293
ESpotDetector::SetAlignmentArea 2293
ESpotDetector::GetAlignmentTolerance 2293
ESpotDetector::SetAlignmentTolerance 2293
ESpotDetector::Detect 2293
ESpotDetector::GetDLClassifierPath 2294
ESpotDetector::SetDLClassifierPath 2294
ESpotDetector::GetDLExecutionSettings 2294
ESpotDetector::SetDLExecutionSettings 2294
ESpotDetector::Draw 2294
ESpotDetector::GetEnableAlignment 2295
ESpotDetector::SetEnableAlignment 2295
ESpotDetector::ESpotDetector 2296
ESpotDetector::GetDetectionThreshold 2296
ESpotDetector::GetEnableType 2296
ESpotDetector::GetGuardBand 2297
ESpotDetector::GetMinimumArea 2297
ESpotDetector::GetPolarity 2297
ESpotDetector::GetSizes 2298
ESpotDetector::IsDLClassifierSet 2298
ESpotDetector::Load 2299
ESpotDetector::GetNumSpots 2299
ESpotDetector::operator= 2299
ESpotDetector::operator== 2300
ESpotDetector::Save 2300
ESpotDetector::SetDetectionThreshold 2300
ESpotDetector::SetEnableType 2301
ESpotDetector::SetGuardBand 2301
ESpotDetector::SetMinimumArea 2302
ESpotDetector::SetPolarity 2302
ESpotDetector::SetSizes 2303
ESpotDetector::GetSourceROI 2303
ESpotDetector::GetSpots 2303
ESpotDetector::UnsetDLClassifier 2304
4.238. EStatistics Class 2304
EStatistics::ComputeAverageMap 2304
EStatistics::ComputePixelStatistics 2306
EStatistics::ComputeStandardDeviationMap 2310
EStatistics::ComputeStatistics 2312
4.239. EStringPair Class 2317
EStringPair::EStringPair 2317
EStringPair::GetKey 2318
EStringPair::operator= 2318
EStringPair::GetValue 2318
4.240. ESupervisedSegmenter Class 2319
ESupervisedSegmenter::Apply 2321
ESupervisedSegmenter::GetCapacity 2322

ESupervisedSegmenter::SetCapacity 2322
ESupervisedSegmenter::GetClassificationThreshold 2323
ESupervisedSegmenter::SetClassificationThreshold 2323
ESupervisedSegmenter::ESupervisedSegmenter 2323
ESupervisedSegmenter::Evaluate 2323
ESupervisedSegmenter::GetForceGrayscale 2324
ESupervisedSegmenter::SetForceGrayscale 2324
ESupervisedSegmenter::GetNumPatchesForImage 2324
ESupervisedSegmenter::GetTrainingMetrics 2324
ESupervisedSegmenter::GetValidationMetrics 2325
ESupervisedSegmenter::GetInferenceScale 2325
ESupervisedSegmenter::operator= 2325
ESupervisedSegmenter::GetPatchSize 2326
ESupervisedSegmenter::SetPatchSize 2326
ESupervisedSegmenter::GetSamplingDensity 2326
ESupervisedSegmenter::SetSamplingDensity 2326
ESupervisedSegmenter::GetScale 2326
ESupervisedSegmenter::SetScale 2326
ESupervisedSegmenter::GetScaleDisabledAtInference 2327
ESupervisedSegmenter::SetScaleDisabledAtInference 2327
ESupervisedSegmenter::SerializeSettings 2327
ESupervisedSegmenter::GetToolType 2327
4.241. ESupervisedSegmenterBlob Class 2328
ESupervisedSegmenterBlob::GetArea 2328
ESupervisedSegmenterBlob::GetAverageBackgroundProbability 2329
ESupervisedSegmenterBlob::GetAverageProbability 2329
ESupervisedSegmenterBlob::ESupervisedSegmenterBlob 2329
ESupervisedSegmenterBlob::GetLabel 2329
ESupervisedSegmenterBlob::GetMaxBackgroundProbability 2330
ESupervisedSegmenterBlob::GetMaxProbability 2330
ESupervisedSegmenterBlob::GetMinBackgroundProbability 2330
ESupervisedSegmenterBlob::GetMinProbability 2330
ESupervisedSegmenterBlob::operator= 2330
ESupervisedSegmenterBlob::GetRegion 2331
4.242. ESupervisedSegmenterMetrics Class 2332
ESupervisedSegmenterMetrics::GetBalancedError 2335
ESupervisedSegmenterMetrics::GetBalancedIntersectionOverUnion 2335
ESupervisedSegmenterMetrics::GetBalancedPixelAccuracy 2335
ESupervisedSegmenterMetrics::GetBlobDetectionAveragePrecision 2336
ESupervisedSegmenterMetrics::GetBlobDetectionBestFScore 2336
ESupervisedSegmenterMetrics::GetBlobDetectionBestFScoreThreshold 2336
ESupervisedSegmenterMetrics::GetBlobDetectionFScore 2337
ESupervisedSegmenterMetrics::GetBlobDetectionPrecision 2337
ESupervisedSegmenterMetrics::GetBlobDetectionRecall 2337
ESupervisedSegmenterMetrics::GetError 2337
ESupervisedSegmenterMetrics::ESupervisedSegmenterMetrics 2338
ESupervisedSegmenterMetrics::GetGroundtruthBlobConfusion 2338
ESupervisedSegmenterMetrics::GetIntersectionOverUnion 2339
ESupervisedSegmenterMetrics::GetLabel 2339
ESupervisedSegmenterMetrics::GetLabelError 2340
ESupervisedSegmenterMetrics::GetNormalizedPixelConfusion 2340
ESupervisedSegmenterMetrics::GetPixelConfusion 2341
ESupervisedSegmenterMetrics::GetPixelLabelAccuracy 2341
ESupervisedSegmenterMetrics::GetPredictedBlobConfusion 2341
ESupervisedSegmenterMetrics::IsValid 2342
ESupervisedSegmenterMetrics::Load 2342
ESupervisedSegmenterMetrics::GetNumLabels 2343
ESupervisedSegmenterMetrics::operator= 2343
ESupervisedSegmenterMetrics::operator== 2343

ESupervisedSegmenterMetrics::GetPixelAccuracy 2344
ESupervisedSegmenterMetrics::Save 2344
ESupervisedSegmenterMetrics::GetWeightedError 2345
ESupervisedSegmenterMetrics::GetWeightedIntersectionOverUnion 2345
ESupervisedSegmenterMetrics::GetWeightedPixelAccuracy 2345
4.243. ESupervisedSegmenterResult Class 2345
ESupervisedSegmenterResult::GetClassificationThreshold 2347
ESupervisedSegmenterResult::SetClassificationThreshold 2347
ESupervisedSegmenterResult::GetColorizedSegmentation 2347
ESupervisedSegmenterResult::GetColorizedSegmentationWithTransparency 2347
ESupervisedSegmenterResult::Draw 2348
ESupervisedSegmenterResult::ESupervisedSegmenterResult 2349
ESupervisedSegmenterResult::GetBlobs 2349
ESupervisedSegmenterResult::GetLabel 2349
ESupervisedSegmenterResult::GetLabelColor 2350
ESupervisedSegmenterResult::GetNumBlobs 2350
ESupervisedSegmenterResult::GetProbabilityMap 2351
ESupervisedSegmenterResult::GetRegionForLabel 2351
ESupervisedSegmenterResult::GetGroundtruthSegmentationMap 2352
ESupervisedSegmenterResult::SetGroundtruthSegmentationMap 2352
ESupervisedSegmenterResult::HasForegroundSegments 2352
ESupervisedSegmenterResult::HasGroundtruthSegmentation 2352
ESupervisedSegmenterResult::GetHeight 2352
ESupervisedSegmenterResult::GetImageMetrics 2353
ESupervisedSegmenterResult::IsValid 2353
ESupervisedSegmenterResult::GetNumLabels 2353
ESupervisedSegmenterResult::operator= 2353
ESupervisedSegmenterResult::RemoveGroundtruthSegmentation 2354
ESupervisedSegmenterResult::GetScore 2354
ESupervisedSegmenterResult::GetSegmentationMap 2354
ESupervisedSegmenterResult::GetWidth 2354
4.244. EThreeLayersImageSegmenter Class 2355
EThreeLayersImageSegmenter::GetBlackLayerEncoded 2355
EThreeLayersImageSegmenter::SetBlackLayerEncoded 2355
EThreeLayersImageSegmenter::GetBlackLayerIndex 2356
EThreeLayersImageSegmenter::SetBlackLayerIndex 2356
EThreeLayersImageSegmenter::Load 2356
EThreeLayersImageSegmenter::GetNeutralLayerEncoded 2357
EThreeLayersImageSegmenter::SetNeutralLayerEncoded 2357
EThreeLayersImageSegmenter::GetNeutralLayerIndex 2357
EThreeLayersImageSegmenter::SetNeutralLayerIndex 2357
EThreeLayersImageSegmenter::operator== 2357
EThreeLayersImageSegmenter::Save 2358
EThreeLayersImageSegmenter::GetWhiteLayerEncoded 2358
EThreeLayersImageSegmenter::SetWhiteLayerEncoded 2358
EThreeLayersImageSegmenter::GetWhiteLayerIndex 2358
EThreeLayersImageSegmenter::SetWhiteLayerIndex 2358
4.245. ETwoLayersImageSegmenter Class 2359
ETwoLayersImageSegmenter::GetBlackLayerEncoded 2360
ETwoLayersImageSegmenter::SetBlackLayerEncoded 2360
ETwoLayersImageSegmenter::GetBlackLayerIndex 2360
ETwoLayersImageSegmenter::SetBlackLayerIndex 2360
ETwoLayersImageSegmenter::Load 2360
ETwoLayersImageSegmenter::operator== 2361
ETwoLayersImageSegmenter::Save 2361
ETwoLayersImageSegmenter::GetWhiteLayerEncoded 2362
ETwoLayersImageSegmenter::SetWhiteLayerEncoded 2362
ETwoLayersImageSegmenter::GetWhiteLayerIndex 2362
ETwoLayersImageSegmenter::SetWhiteLayerIndex 2362

4.246. EUnsupervisedSegmenter Class 2362
EUnsupervisedSegmenter::Apply 2365
EUnsupervisedSegmenter::GetCapacity 2366
EUnsupervisedSegmenter::SetCapacity 2366
EUnsupervisedSegmenter::GetClassificationThreshold 2366
EUnsupervisedSegmenter::SetClassificationThreshold 2366
EUnsupervisedSegmenter::EUnsupervisedSegmenter 2367
EUnsupervisedSegmenter::GetForceGrayscale 2367
EUnsupervisedSegmenter::SetForceGrayscale 2367
EUnsupervisedSegmenter::GetNumPatchesForImage 2368
EUnsupervisedSegmenter::GetTrainingMetrics 2368
EUnsupervisedSegmenter::GetValidationMetrics 2368
EUnsupervisedSegmenter::GetGoodLabel 2369
EUnsupervisedSegmenter::SetGoodLabel 2369
EUnsupervisedSegmenter::GetInferenceScale 2369
EUnsupervisedSegmenter::operator= 2369
EUnsupervisedSegmenter::GetPatchSize 2370
EUnsupervisedSegmenter::SetPatchSize 2370
EUnsupervisedSegmenter::GetSamplingDensity 2370
EUnsupervisedSegmenter::SetSamplingDensity 2370
EUnsupervisedSegmenter::GetScale 2371
EUnsupervisedSegmenter::SetScale 2371
EUnsupervisedSegmenter::GetScaleDisabledAtInference 2371
EUnsupervisedSegmenter::SetScaleDisabledAtInference 2371
EUnsupervisedSegmenter::SerializeSettings 2371
EUnsupervisedSegmenter::GetToolType 2372
4.247. EUnsupervisedSegmenterMetrics Class 2372
EUnsupervisedSegmenterMetrics::AddErrorResult 2373
EUnsupervisedSegmenterMetrics::AddMetrics 2374
EUnsupervisedSegmenterMetrics::AddResult 2374
EUnsupervisedSegmenterMetrics::GetAverageScoreOnDefectiveImages 2374
EUnsupervisedSegmenterMetrics::GetAverageScoreOnGoodImages 2375
EUnsupervisedSegmenterMetrics::GetError 2375
EUnsupervisedSegmenterMetrics::EUnsupervisedSegmenterMetrics 2375
EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracy 2376
EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracyClassificationThreshold 2376
EUnsupervisedSegmenterMetrics::IsTotallyUnsupervised 2377
EUnsupervisedSegmenterMetrics::IsValid 2377
EUnsupervisedSegmenterMetrics::Load 2377
EUnsupervisedSegmenterMetrics::operator= 2378
EUnsupervisedSegmenterMetrics::RemoveErrorResult 2378
EUnsupervisedSegmenterMetrics::RemoveResult 2378
EUnsupervisedSegmenterMetrics::Save 2379
4.248. EUnsupervisedSegmenterResult Class 2379
EUnsupervisedSegmenterResult::GetClassificationScore 2380
EUnsupervisedSegmenterResult::Draw 2381
EUnsupervisedSegmenterResult::GetError 2381
EUnsupervisedSegmenterResult::EUnsupervisedSegmenterResult 2382
EUnsupervisedSegmenterResult::IsComplete 2382
EUnsupervisedSegmenterResult::IsDefective 2382
EUnsupervisedSegmenterResult::IsGood 2383
EUnsupervisedSegmenterResult::IsValid 2383
EUnsupervisedSegmenterResult::operator= 2383
EUnsupervisedSegmenterResult::GetRegion 2383
EUnsupervisedSegmenterResult::GetSegmentationMap 2384
4.249. EUnwarpingLut Class 2384
EUnwarpingLut::EUnwarpingLut 2384
4.250. EUtils Class 2384

EUtils::Copy 2385
4.251. EVector Class 2387
EVector::Empty 2388
EVector::Load 2388
EVector::GetNumElements 2389
EVector::SetNumElements 2389
EVector::RemoveElement 2389
EVector::Save 2389
4.252. EVectorModel Class 2390
EVectorModel::GetCenter 2391
EVectorModel::SetCenter 2391
EVectorModel::DisableDrawArea 2391
EVectorModel::Draw 2391
EVectorModel::DrawWithCurrentPen 2392
EVectorModel::EnableDrawArea 2392
EVectorModel::EVectorModel 2393
EVectorModel::GetVectorModelExtremas 2393
EVectorModel::Load 2394
EVectorModel::LoadDXF 2394
EVectorModel::operator= 2395
EVectorModel::GetRoot 2395
EVectorModel::Save 2395
EVectorModel::GetScale 2396
EVectorModel::SetScale 2396
4.253. EWedge Class 2396
EWedge::GetAmplitude 2398
EWedge::SetAmplitude 2398
EWedge::GetApexAngle 2398
EWedge::GetBreadth 2399
EWedge::CopyTo 2399
EWedge::GetDirect 2399
EWedge::SetDirect 2399
EWedge::GetEndAngle 2400
EWedge::EWedge 2400
EWedge::GetFullBreadth 2402
EWedge::GetFullCircle 2402
EWedge::GetCorners 2402
EWedge::GetEdges 2403
EWedge::GetInnerPoint 2403
EWedge::GetMidEdges 2404
EWedge::GetOuterPoint 2404
EWedge::GetPoint 2405
EWedge::GetInnerApex 2405
EWedge::GetInnerArcLength 2405
EWedge::GetInnerDiameter 2405
EWedge::GetInnerEnd 2406
EWedge::GetInnerOrg 2406
EWedge::GetInnerRadius 2406
EWedge::operator= 2406
EWedge::GetOrgAngle 2407
EWedge::GetOuterApex 2407
EWedge::GetOuterArcLength 2407
EWedge::GetOuterDiameter 2408
EWedge::GetOuterEnd 2408
EWedge::GetOuterOrg 2408
EWedge::GetOuterRadius 2408
EWedge::SetDiameters 2409
EWedge::SetFromCenterAndOrigin 2409
EWedge::SetFromOriginMiddleEnd 2410

EWedge::SetFromTwoPoints 2411
EWedge::SetRadii 2411
4.254. EWedgeGauge Class 2412
2415
EWedgeGauge::SetActive 2415
EWedgeGauge::GetActiveEdges 2416
EWedgeGauge::SetActiveEdges 2416
EWedgeGauge::AddSkipRange 2416
EWedgeGauge::GetAverageDistance 2417
EWedgeGauge::CopyTo 2417
EWedgeGauge::Drag 2418
EWedgeGauge::Draw 2418
EWedgeGauge::DrawWithCurrentPen 2419
EWedgeGauge::EWedgeGauge 2420
EWedgeGauge::GetFilteringThreshold 2420
EWedgeGauge::SetFilteringThreshold 2420
EWedgeGauge::GetMeasuredPoint 2421
EWedgeGauge::GetMinNumFitSamples 2421
EWedgeGauge::GetSampleA 2422
EWedgeGauge::GetSampleA 2423
EWedgeGauge::GetSampleInnerEdge 2423
EWedgeGauge::GetSampleLeftEdge 2424
EWedgeGauge::GetSampleOuterEdge 2425
EWedgeGauge::GetSampleR 2426
EWedgeGauge::GetSampleR 2426
EWedgeGauge::GetSampleRightEdge 2427
EWedgeGauge::GetSkipRange 2428
EWedgeGauge::HitTest 2428
EWedgeGauge::GetHVConstraint 2429
EWedgeGauge::SetHVConstraint 2429
EWedgeGauge::Measure 2429
EWedgeGauge::GetMeasuredWedge 2430
EWedgeGauge::MeasureSample 2430
EWedgeGauge::MeasureWithoutFitting 2431
EWedgeGauge::GetMinAmplitude 2431
EWedgeGauge::SetMinAmplitude 2431
EWedgeGauge::GetMinArea 2432
EWedgeGauge::SetMinArea 2432
EWedgeGauge::GetNumFilteringPasses 2432
EWedgeGauge::SetNumFilteringPasses 2432
EWedgeGauge::GetNumSamples 2433
EWedgeGauge::GetNumSamplesA 2433
EWedgeGauge::GetNumSamplesA 2433
EWedgeGauge::GetNumSamplesInnerEdge 2434
EWedgeGauge::GetNumSamplesLeftEdge 2434
EWedgeGauge::GetNumSamplesOuterEdge 2434
EWedgeGauge::GetNumSamplesR 2434
EWedgeGauge::GetNumSamplesR 2435
EWedgeGauge::GetNumSamplesRightEdge 2435
EWedgeGauge::GetNumSkipRanges 2435
EWedgeGauge::GetNumValidSamples 2435
EWedgeGauge::operator= 2436
EWedgeGauge::Plot 2436
EWedgeGauge::PlotWithCurrentPen 2437
EWedgeGauge::Process 2438
EWedgeGauge::GetRectangularSamplingArea 2439
EWedgeGauge::SetRectangularSamplingArea 2439
EWedgeGauge::RemoveAllSkipRanges 2439
EWedgeGauge::RemoveSkipRange 2439

EWedgeGauge::GetSamplingStep ... 2440
EWedgeGauge::SetSamplingStep ... 2440
EWedgeGauge::SetDiameters ... 2440
EWedgeGauge::SetFromOriginMiddleEnd ... 2441
EWedgeGauge::SetFromTwoPoints ... 2441
EWedgeGauge::SetMinNumFitSamples ... 2442
EWedgeGauge::SetRadii ... 2443
EWedgeGauge::GetSmoothing ... 2443
EWedgeGauge::SetSmoothing ... 2443
EWedgeGauge::GetThickness ... 2444
EWedgeGauge::SetThickness ... 2444
EWedgeGauge::GetThreshold ... 2444
EWedgeGauge::SetThreshold ... 2444
EWedgeGauge::GetTolerance ... 2444
EWedgeGauge::SetTolerance ... 2444
EWedgeGauge::GetTransitionChoice ... 2445
EWedgeGauge::SetTransitionChoice ... 2445
EWedgeGauge::GetTransitionIndex ... 2445
EWedgeGauge::SetTransitionIndex ... 2445
EWedgeGauge::GetTransitionType ... 2446
EWedgeGauge::SetTransitionType ... 2446
EWedgeGauge::GetType ... 2446
EWedgeGauge::GetValid ... 2446
... 2447
EWedgeGauge::SetWedge ... 2447
4.255. EWedgeShape Class ... 2447
EWedgeShape::GetAmplitude ... 2449
EWedgeShape::SetAmplitude ... 2449
EWedgeShape::GetAngle ... 2449
EWedgeShape::SetAngle ... 2449
EWedgeShape::GetApexAngle ... 2450
EWedgeShape::GetBreadth ... 2450
EWedgeShape::GetCenter ... 2450
EWedgeShape::SetCenter ... 2450
EWedgeShape::GetCenterX ... 2451
EWedgeShape::GetCenterY ... 2451
EWedgeShape::Closest ... 2451
EWedgeShape::CopyTo ... 2451
EWedgeShape::GetDirect ... 2452
EWedgeShape::SetDirect ... 2452
EWedgeShape::Drag ... 2452
EWedgeShape::Draw ... 2453
EWedgeShape::DrawWithCurrentPen ... 2454
EWedgeShape::GetEndAngle ... 2454
EWedgeShape::GetFullBreadth ... 2455
EWedgeShape::GetFullCircle ... 2455
EWedgeShape::GetCorners ... 2455
EWedgeShape::GetEdges ... 2456
EWedgeShape::GetInnerPoint ... 2456
EWedgeShape::GetMidEdges ... 2457
EWedgeShape::GetOuterPoint ... 2457
EWedgeShape::GetPoint ... 2458
EWedgeShape::HitTest ... 2458
EWedgeShape::GetInnerApex ... 2458
EWedgeShape::GetInnerArcLength ... 2459
EWedgeShape::GetInnerDiameter ... 2459
EWedgeShape::GetInnerEnd ... 2459
EWedgeShape::GetInnerOrg ... 2459
EWedgeShape::GetInnerRadius ... 2459

EWedgeShape::operator= 2460
EWedgeShape::GetOrgAngle 2460
EWedgeShape::GetOuterApex 2460
EWedgeShape::GetOuterArcLength 2461
EWedgeShape::GetOuterDiameter 2461
EWedgeShape::GetOuterEnd 2461
EWedgeShape::GetOuterOrg 2462
EWedgeShape::GetOuterRadius 2462
EWedgeShape::GetScale 2462
EWedgeShape::SetScale 2462
EWedgeShape::SetCenterXY 2463
EWedgeShape::SetDiameters 2463
EWedgeShape::SetFromCenterAndOrigin 2464
EWedgeShape::SetFromOriginMiddleEnd 2464
EWedgeShape::SetFromTwoPoints 2465
EWedgeShape::SetRadii 2466
EWedgeShape::GetType 2466
2466
EWedgeShape::SetWedge 2466
4.256. EWindowsDrawAdapter Class 2467
EWindowsDrawAdapter::Arc 2468
EWindowsDrawAdapter::BackedText 2468
2469
EWindowsDrawAdapter::SetBrush 2469
EWindowsDrawAdapter::Close 2469
EWindowsDrawAdapter::DrawPoint 2470
EWindowsDrawAdapter::Ellipse 2470
EWindowsDrawAdapter::EWindowsDrawAdapter 2471
EWindowsDrawAdapter::FilledEllipse 2471
EWindowsDrawAdapter::FilledPolygon 2472
EWindowsDrawAdapter::FilledRectangle 2472
2473
EWindowsDrawAdapter::SetFont 2473
EWindowsDrawAdapter::GetTextSize 2473
EWindowsDrawAdapter::GetHDC 2474
EWindowsDrawAdapter::Image 2474
EWindowsDrawAdapter::Line 2475
2475
EWindowsDrawAdapter::SetPen 2475
EWindowsDrawAdapter::Polygon 2476
EWindowsDrawAdapter::Rectangle 2476
EWindowsDrawAdapter::Text 2477
EWindowsDrawAdapter::UseCurrentBrush 2477
EWindowsDrawAdapter::UseCurrentPen 2478
4.257. EWorldShape Class 2478
EWorldShape::AddLandmark 2482
EWorldShape::AddPoint 2482
EWorldShape::GetAngle 2483
EWorldShape::SetAngle 2483
EWorldShape::AutoCalibrate 2483
EWorldShape::AutoCalibrateDotGrid 2484
EWorldShape::AutoCalibrateLandmarks 2485
EWorldShape::Calibrate 2485
EWorldShape::GetCalibrationModes 2486
EWorldShape::SetCalibrationModes 2486
EWorldShape::CalibrationSucceeded 2486
EWorldShape::GetCenter 2487
EWorldShape::SetCenter 2487
EWorldShape::GetCenterX 2487

EWorldShape::GetCenterY 2487
EWorldShape::Closest 2488
EWorldShape::DisableTypeFilter 2488
2488
EWorldShape::SetDistortion 2488
EWorldShape::GetDistortionStrength 2488
EWorldShape::SetDistortionStrength 2488
EWorldShape::Drag 2489
EWorldShape::DragLandmark 2489
EWorldShape::Draw 2490
EWorldShape::DrawCrossGrid 2491
EWorldShape::DrawCrossGridWithCurrentPen 2492
EWorldShape::DrawGrid 2493
EWorldShape::DrawGridWithCurrentPen 2493
EWorldShape::DrawLandmarks 2494
EWorldShape::DrawWithCurrentPen 2494
EWorldShape::EmptyLandmarks 2495
EWorldShape::EnableTypeFilter 2495
EWorldShape::EWorldShape 2495
EWorldShape::GetFieldHeight 2496
EWorldShape::GetFieldWidth 2496
EWorldShape::GetLandmarkElement 2497
EWorldShape::GetGridPointsMaxVariation 2497
EWorldShape::GetGridPointsMaxVariationThreshold 2498
EWorldShape::SetGridPointsMaxVariationThreshold 2498
EWorldShape::GetGridPointsMeanVariation 2498
EWorldShape::GetGridPointsMeanVariationThreshold 2498
EWorldShape::SetGridPointsMeanVariationThreshold 2498
EWorldShape::GetHitLandmark 2499
EWorldShape::HitLandmarks 2499
EWorldShape::HitTest 2499
EWorldShape::GetNumLandmarkElements 2500
EWorldShape::operator= 2500
EWorldShape::GetPanX 2500
EWorldShape::GetPanY 2500
EWorldShape::GetPerspectiveStrength 2501
EWorldShape::GetRatio 2501
EWorldShape::SetRatio 2501
EWorldShape::RebuildGrid 2501
EWorldShape::RemoveLandmark 2502
EWorldShape::GetScale 2503
EWorldShape::SetScale 2503
EWorldShape::GetSensorHeight 2503
EWorldShape::SensorToWorld 2503
EWorldShape::GetSensorWidth 2504
EWorldShape::SetCenterXY 2504
EWorldShape::SetFieldSize 2504
EWorldShape::SetPan 2505
EWorldShape::SetPerspective 2505
EWorldShape::SetResolution 2506
EWorldShape::SetSensor 2507
EWorldShape::SetSensorSize 2508
EWorldShape::SetSize 2509
EWorldShape::SetupUnwarp 2509
EWorldShape::SetZoom 2510
EWorldShape::GetTiltXAngle 2510
EWorldShape::GetTiltYAngle 2511
EWorldShape::GetType 2511
EWorldShape::Unwarp 2511

EWorldShape::WorldToSensor 2512
EWorldShape::GetXResolution 2513
EWorldShape::GetYResolution 2513
EWorldShape::GetZoomX 2513
EWorldShape::GetZoomY 2513
4.258. EZMap Class 2514
EZMap::AddMetadata 2517
EZMap::Clear 2517
EZMap::Create 2517
EZMap::Draw 2518
EZMap::DrawImage 2521
EZMap::GetBufferPtr 2523
EZMap::GetCheckedBufferPtr 2523
EZMap::GetMetadata 2524
EZMap::GetResolution 2524
EZMap::GetSizeInWorld 2525
EZMap::GetWorldPositionFromMapPosition 2525
EZMap::GetWorldPositionFromPixelPosition 2526
EZMap::GetZMapPositionFromPixelPosition 2526
EZMap::GetHeight 2527
EZMap::SetHeight 2527
EZMap::ImageToWorld 2527
EZMap::ImageToZMap 2528
EZMap::IsVoid 2528
EZMap::Load 2528
EZMap::LoadImage 2529
EZMap::LoadImageAndMetadata 2529
EZMap::LoadMetadata 2530
EZMap::GetMapToWorldMatrix 2530
EZMap::SetMapToWorldMatrix 2530
EZMap::ResetWorldTransformation 2531
EZMap::GetRowPitch 2531
EZMap::Save 2531
EZMap::SaveImage 2532
EZMap::SaveImageAndMetadata 2532
EZMap::SaveMetadata 2533
EZMap::SetBufferPtr 2533
EZMap::SetResolution 2534
EZMap::SetSize 2534
EZMap::GetType 2535
EZMap::GetWidth 2535
EZMap::SetWidth 2535
EZMap::GetWorldShape 2535
EZMap::WorldToImage 2536
EZMap::GetWorldToMapMatrix 2536
EZMap::SetWorldToMapMatrix 2536
EZMap::WorldToZMap 2537
EZMap::GetXResolution 2537
EZMap::SetXResolution 2537
EZMap::GetYResolution 2537
EZMap::SetYResolution 2537
EZMap::ZMapToImage 2538
EZMap::ZMapToWorld 2538
EZMap::GetZResolution 2539
EZMap::SetZResolution 2539
4.259. EZMap16 Class 2539
EZMap16::AddMetadata 2543
EZMap16::AsEImage 2543
EZMap16::Clear 2544

EZMap16::ClearMetadata ... 2544
EZMap16::ConvertCoordinatesMapToPixel ... 2544
EZMap16::ConvertCoordinatesPixelToMap ... 2545
EZMap16::CopyMetadataTo ... 2545
EZMap16::DeleteMetadata ... 2546
EZMap16::Draw ... 2546
EZMap16::DrawImage ... 2549
EZMap16::EZMap16 ... 2551
EZMap16::FillUndefinedPixels ... 2551
EZMap16::FillUndefinedPixelsWithMedian ... 2552
EZMap16::GetBufferPtr ... 2552
EZMap16::GetCheckedBufferPtr ... 2553
EZMap16::GetMetadata ... 2554
EZMap16::GetPixel ... 2554
EZMap16::GetPixelPositionFromWorldPosition ... 2554
EZMap16::GetResolution ... 2555
EZMap16::GetSizeInWorld ... 2555
EZMap16::GetWorldPositionFromMapPosition ... 2556
EZMap16::GetWorldPositionFromPixelPosition ... 2556
EZMap16::GetZMapPositionFromPixelPosition ... 2557
EZMap16::GetZRange ... 2557
EZMap16::GetZValue ... 2558
EZMap16::GetHeight ... 2558
EZMap16::SetHeight ... 2558
EZMap16::ImageToWorld ... 2558
EZMap16::ImageToZMap ... 2559
EZMap16::IsVoid ... 2559
EZMap16::Load ... 2560
EZMap16::LoadImage ... 2560
EZMap16::LoadImageAndMetadata ... 2561
EZMap16::LoadMetadata ... 2561
EZMap16::GetMapToWorldMatrix ... 2562
EZMap16::SetMapToWorldMatrix ... 2562
EZMap16::ModifyMetadata ... 2562
EZMap16::operator= ... 2562
EZMap16::ResetWorldTransformation ... 2563
EZMap16::GetRowPitch ... 2563
EZMap16::Save ... 2563
EZMap16::SaveImage ... 2564
EZMap16::SaveImageAndMetadata ... 2564
EZMap16::SaveMetadata ... 2565
EZMap16::SetBufferPtr ... 2565
EZMap16::SetPixel ... 2566
EZMap16::SetResolution ... 2566
EZMap16::SetSize ... 2567
EZMap16::SetZValue ... 2568
EZMap16::GetType ... 2568
EZMap16::GetUndefinedValue ... 2569
EZMap16::GetWidth ... 2569
EZMap16::SetWidth ... 2569
EZMap16::GetWorldShape ... 2569
EZMap16::WorldToImage ... 2570
EZMap16::GetWorldToMapMatrix ... 2570
EZMap16::SetWorldToMapMatrix ... 2570
EZMap16::WorldToZMap ... 2570
EZMap16::GetXResolution ... 2571
EZMap16::SetXResolution ... 2571
EZMap16::GetYResolution ... 2571
EZMap16::SetYResolution ... 2571

EZMap16::ZMapToImage 2572
EZMap16::ZMapToWorld 2572
EZMap16::GetZResolution 2573
EZMap16::SetZResolution 2573
4.260. EZMap32f Class 2573
EZMap32f::AddMetadata 2577
EZMap32f::AsEImage 2577
EZMap32f::Clear 2578
EZMap32f::ClearMetadata 2578
EZMap32f::ConvertCoordinatesMapToPixel 2578
EZMap32f::ConvertCoordinatesPixelToMap 2579
EZMap32f::CopyMetadataTo 2579
EZMap32f::DeleteMetadata 2580
EZMap32f::Draw 2580
EZMap32f::DrawImage 2583
EZMap32f::EZMap32f 2585
EZMap32f::FillUndefinedPixels 2585
EZMap32f::FillUndefinedPixelsWithMedian 2586
EZMap32f::GetBufferPtr 2586
EZMap32f::GetCheckedBufferPtr 2587
EZMap32f::GetMetadata 2588
EZMap32f::GetPixel 2588
EZMap32f::GetPixelPositionFromWorldPosition 2588
EZMap32f::GetResolution 2589
EZMap32f::GetSizeInWorld 2589
EZMap32f::GetWorldPositionFromMapPosition 2590
EZMap32f::GetWorldPositionFromPixelPosition 2590
EZMap32f::GetZMapPositionFromPixelPosition 2591
EZMap32f::GetZRange 2591
EZMap32f::GetZValue 2592
EZMap32f::GetHeight 2592
EZMap32f::SetHeight 2592
EZMap32f::ImageToWorld 2592
EZMap32f::ImageToZMap 2593
EZMap32f::IsVoid 2593
EZMap32f::Load 2594
EZMap32f::LoadImage 2594
EZMap32f::LoadImageAndMetadata 2595
EZMap32f::LoadMetadata 2595
EZMap32f::GetMapToWorldMatrix 2596
EZMap32f::SetMapToWorldMatrix 2596
EZMap32f::ModifyMetadata 2596
EZMap32f::operator= 2596
EZMap32f::ResetWorldTransformation 2597
EZMap32f::GetRowPitch 2597
EZMap32f::Save 2597
EZMap32f::SaveImage 2598
EZMap32f::SaveImageAndMetadata 2598
EZMap32f::SaveMetadata 2599
EZMap32f::SetBufferPtr 2599
EZMap32f::SetPixel 2600
EZMap32f::SetResolution 2600
EZMap32f::SetSize 2601
EZMap32f::SetZValue 2602
EZMap32f::GetType 2602
EZMap32f::GetUndefinedValue 2603
EZMap32f::GetWidth 2603
EZMap32f::SetWidth 2603
EZMap32f::GetWorldShape 2603

EZMap32f::WorldToImage ... 2604
EZMap32f::GetWorldToMapMatrix ... 2604
EZMap32f::SetWorldToMapMatrix ... 2604
EZMap32f::WorldToZMap ... 2604
EZMap32f::GetXResolution ... 2605
EZMap32f::SetXResolution ... 2605
EZMap32f::GetYResolution ... 2605
EZMap32f::SetYResolution ... 2605
EZMap32f::ZMapToImage ... 2606
EZMap32f::ZMapToWorld ... 2606
EZMap32f::GetZResolution ... 2607
EZMap32f::SetZResolution ... 2607
4.261. EZMap8 Class ... 2607
EZMap8::AddMetadata ... 2611
EZMap8::AsEImage ... 2611
EZMap8::Clear ... 2611
EZMap8::ClearMetadata ... 2612
EZMap8::ConvertCoordinatesMapToPixel ... 2612
EZMap8::ConvertCoordinatesPixelToMap ... 2612
EZMap8::CopyMetadataTo ... 2613
EZMap8::DeleteMetadata ... 2613
EZMap8::Draw ... 2614
EZMap8::DrawImage ... 2617
EZMap8::EZMap8 ... 2619
EZMap8::FillUndefinedPixels ... 2619
EZMap8::FillUndefinedPixelsWithMedian ... 2620
EZMap8::GetBufferPtr ... 2620
EZMap8::GetCheckedBufferPtr ... 2621
EZMap8::GetMetadata ... 2621
EZMap8::GetPixel ... 2622
EZMap8::GetPixelPositionFromWorldPosition ... 2622
EZMap8::GetResolution ... 2623
EZMap8::GetSizeInWorld ... 2623
EZMap8::GetWorldPositionFromMapPosition ... 2624
EZMap8::GetWorldPositionFromPixelPosition ... 2624
EZMap8::GetZMapPositionFromPixelPosition ... 2625
EZMap8::GetZRange ... 2625
EZMap8::GetZValue ... 2626
EZMap8::GetHeight ... 2626
EZMap8::SetHeight ... 2626
EZMap8::ImageToWorld ... 2626
EZMap8::ImageToZMap ... 2627
EZMap8::IsVoid ... 2627
EZMap8::Load ... 2628
EZMap8::LoadImage ... 2628
EZMap8::LoadImageAndMetadata ... 2629
EZMap8::LoadMetadata ... 2629
EZMap8::GetMapToWorldMatrix ... 2629
EZMap8::SetMapToWorldMatrix ... 2629
EZMap8::ModifyMetadata ... 2630
EZMap8::operator= ... 2630
EZMap8::ResetWorldTransformation ... 2630
EZMap8::GetRowPitch ... 2631
EZMap8::Save ... 2631
EZMap8::SaveImage ... 2631
EZMap8::SaveImageAndMetadata ... 2632
EZMap8::SaveMetadata ... 2632
EZMap8::SetBufferPtr ... 2633
EZMap8::SetPixel ... 2633

EZMap8::SetResolution ... 2634
EZMap8::SetSize ... 2634
EZMap8::SetZValue ... 2635
EZMap8::GetType ... 2635
EZMap8::GetUndefinedValue ... 2636
EZMap8::GetWidth ... 2636
EZMap8::SetWidth ... 2636
EZMap8::GetWorldShape ... 2636
EZMap8::WorldToImage ... 2637
EZMap8::GetWorldToMapMatrix ... 2637
EZMap8::SetWorldToMapMatrix ... 2637
EZMap8::WorldToZMap ... 2637
EZMap8::GetXResolution ... 2638
EZMap8::SetXResolution ... 2638
EZMap8::GetYResolution ... 2638
EZMap8::SetYResolution ... 2638
EZMap8::ZMapToImage ... 2639
EZMap8::ZMapToWorld ... 2639
EZMap8::GetZResolution ... 2640
EZMap8::SetZResolution ... 2640
4.262. EZMapToMeshConverter Class ... 2640
EZMapToMeshConverter::Convert ... 2641
EZMapToMeshConverter::EZMapToMeshConverter ... 2641
EZMapToMeshConverter::Load ... 2642
EZMapToMeshConverter::GetMaxEdgeLength ... 2642
EZMapToMeshConverter::SetMaxEdgeLength ... 2642
EZMapToMeshConverter::operator= ... 2642
EZMapToMeshConverter::Save ... 2643
4.263. EZMapToPointCloudConverter Class ... 2643
EZMapToPointCloudConverter::Convert ... 2644
EZMapToPointCloudConverter::EZMapToPointCloudConverter ... 2645
4.264. TextLabel Class ... 2646
TextLabel::GetAlignment ... 2646
TextLabel::SetAlignment ... 2646
TextLabel::GetAnchor ... 2647
TextLabel::SetAnchor ... 2647
TextLabel::GetBackgroundColor ... 2647
TextLabel::SetBackgroundColor ... 2647
TextLabel::GetColor ... 2647
TextLabel::SetColor ... 2647
TextLabel::GetDisplayAnchor ... 2648
TextLabel::SetDisplayAnchor ... 2648
TextLabel::GetFixed ... 2648
TextLabel::SetFixed ... 2648
TextLabel::operator= ... 2648
TextLabel::GetPosX ... 2648
TextLabel::GetPosY ... 2649
2649
TextLabel::SetPoxX ... 2649
2649
TextLabel::SetPoxY ... 2649
TextLabel::GetSize ... 2649
TextLabel::SetSize ... 2649
TextLabel::GetText ... 2650
TextLabel::SetText ... 2650
TextLabel::TextLabel ... 2650
5. Structures ... 2652
5.1. E3DPoint Struct ... 2652

E3DPoint::DistanceTo 2653
E3DPoint::DistanceToSegment 2653
E3DPoint::E3DPoint 2654
E3DPoint::Load 2654
E3DPoint::operator!= 2655
E3DPoint::operator== 2655
E3DPoint::Save 2656
E3DPoint::SquareDistanceTo 2656
E3DPoint::X 2656
E3DPoint::Y 2657
E3DPoint::Z 2657
5.2. EBarCodeGradingParameters Struct 2657
EBarCodeGradingParameters::GetAdditionalRequirementName 2658
EBarCodeGradingParameters::AdditionalRequirementsGrade 2658
EBarCodeGradingParameters::ConvertToAlphabeticGrade 2659
EBarCodeGradingParameters::DecodabilityGrade 2659
EBarCodeGradingParameters::DecodeGrade 2659
EBarCodeGradingParameters::DefectsGrade 2660
EBarCodeGradingParameters::GlobalGrade 2660
EBarCodeGradingParameters::MinimumEdgeContrastGrade 2660
EBarCodeGradingParameters::MinimumReflectanceGrade 2660
EBarCodeGradingParameters::ModulationGrade 2660
EBarCodeGradingParameters::SymbolContrastGrade 2661
5.3. EBrush Struct 2661
EBrush::Color 2661
EBrush::EBrush 2662
EBrush::IsValid 2662
EBrush::Load 2662
EBrush::Opacity 2663
EBrush::operator!= 2663
EBrush::operator== 2663
EBrush::Save 2664
EBrush::Serialize 2664
5.4. EBW1 Struct 2664
EBW1::EBW1 2665
EBW1::GetSize 2665
EBW1::Value 2665
5.5. EBW16 Struct 2666
EBW16::EBW16 2666
EBW16::GetSize 2667
EBW16::Value 2667
5.6. EBW16Path Struct 2667
EBW16Path::Pixel 2667
EBW16Path::X 2667
EBW16Path::Y 2668
5.7. EBW32 Struct 2668
EBW32::EBW32 2668
EBW32::GetSize 2669
EBW32::Value 2669
5.8. EBW32f Struct 2669
EBW32f::EBW32f 2669
EBW32f::GetSize 2670
EBW32f::Value 2670
5.9. EBW8 Struct 2670
EBW8::EBW8 2671
EBW8::GetSize 2671
EBW8::Value 2671
5.10. EBW8Path Struct 2671

EBW8Path::Pixel 2672
EBW8Path::X 2672
EBW8Path::Y 2672
5.11. EC15 Struct 2672
EC15::C0 2673
EC15::C1 2673
EC15::C2 2673
EC15::EC15 2674
EC15::GetSize 2674
5.12. EC16 Struct 2674
EC16::C0 2675
EC16::C1 2675
EC16::C2 2675
EC16::EC16 2675
EC16::GetSize 2676
5.13. EC24 Struct 2676
EC24::C0 2677
EC24::C1 2677
EC24::C2 2677
EC24::EC24 2677
EC24::GetSize 2678
5.14. EC24A Struct 2678
EC24A::A 2679
EC24A::C0 2679
EC24A::C1 2679
EC24A::C2 2679
EC24A::EC24A 2680
EC24A::GetSize 2680
5.15. EC24Path Struct 2680
EC24Path::Pixel 2681
EC24Path::X 2681
EC24Path::Y 2681
5.16. EC48 Struct 2681
EC48::C0 2682
EC48::C1 2682
EC48::C2 2682
EC48::EC48 2682
EC48::GetSize 2683
5.17. EColor Struct 2683
EColor::C0 2683
EColor::C1 2684
EColor::C2 2684
EColor::EColor 2684
5.18. EDepth16 Struct 2685
EDepth16::EDepth16 2685
EDepth16::GetSize 2685
EDepth16::Value 2685
5.19. EDepth32f Struct 2686
EDepth32f::EDepth32f 2686
EDepth32f::GetSize 2686
EDepth32f::Value 2687
5.20. EDepth8 Struct 2687
EDepth8::EDepth8 2687
EDepth8::GetSize 2688
EDepth8::Value 2688
5.21. EFeatureData Struct 2688
EFeatureData::FeatDataSize 2688
EFeatureData::FeatDataType 2689

EFeatureData::FeatNum ... 2689
EFeatureData::Size ... 2689
5.22. EISH Struct ... 2689
EISH::H ... 2690
EISH::I ... 2690
EISH::S ... 2690
5.23. ELAB Struct ... 2690
ELAB::A ... 2691
ELAB::B ... 2691
ELAB::L ... 2691
5.24. ELCH Struct ... 2691
ELCH::C ... 2692
ELCH::H ... 2692
ELCH::L ... 2692
5.25. ELSH Struct ... 2692
ELSH::H ... 2692
ELSH::L ... 2693
ELSH::S ... 2693
5.26. ELUV Struct ... 2693
ELUV::L ... 2693
ELUV::U ... 2694
ELUV::V ... 2694
5.27. EMatchPosition Struct ... 2694
EMatchPosition::Angle ... 2695
EMatchPosition::AreaRatio ... 2695
EMatchPosition::CenterX ... 2695
EMatchPosition::CenterY ... 2696
EMatchPosition::Interpolated ... 2696
EMatchPosition::Scale ... 2696
EMatchPosition::ScaleX ... 2696
EMatchPosition::ScaleY ... 2697
EMatchPosition::Score ... 2697
EMatchPosition::ToRegion ... 2697
5.28. EMatrixCodeIso15415GradingParameters Struct ... 2698
EMatrixCodeIso15415GradingParameters::AxialNonUniformity ... 2698
EMatrixCodeIso15415GradingParameters::AxialNonUniformityGrade ... 2699
EMatrixCodeIso15415GradingParameters::DecodingGrade ... 2699
EMatrixCodeIso15415GradingParameters::EMatrixCodeIso15415GradingParameters ... 2699
EMatrixCodeIso15415GradingParameters::FixedPatternDamageGrade ... 2699
EMatrixCodeIso15415GradingParameters::GridNonUniformity ... 2700
EMatrixCodeIso15415GradingParameters::GridNonUniformityGrade ... 2700
EMatrixCodeIso15415GradingParameters::HorizontalPrintGrowth ... 2700
EMatrixCodeIso15415GradingParameters::ModulationGrade ... 2700
EMatrixCodeIso15415GradingParameters::OverallSymbolGrade ... 2700
EMatrixCodeIso15415GradingParameters::ReflectanceMarginGrade ... 2701
EMatrixCodeIso15415GradingParameters::ScanGrade ... 2701
EMatrixCodeIso15415GradingParameters::SymbolContrast ... 2701
EMatrixCodeIso15415GradingParameters::SymbolContrastGrade ... 2701
EMatrixCodeIso15415GradingParameters::UnusedErrorCorrection ... 2701
EMatrixCodeIso15415GradingParameters::UnusedErrorCorrectionGrade ... 2702
EMatrixCodeIso15415GradingParameters::VerticalPrintGrowth ... 2702
5.29. EMatrixCodeIso29158CalibrationParameters Struct ... 2702
EMatrixCodeIso29158CalibrationParameters::EMatrixCodeIso29158CalibrationParameters ... 2703
EMatrixCodeIso29158CalibrationParameters::MLcal ... 2703
EMatrixCodeIso29158CalibrationParameters::Rcal ... 2703
EMatrixCodeIso29158CalibrationParameters::SRcal ... 2703
EMatrixCodeIso29158CalibrationParameters::SRtarget ... 2704
5.30. EMatrixCodeIso29158GradingParameters Struct ... 2704

EMatrixCodeIso29158GradingParameters::CellContrastGrade 2704
EMatrixCodeIso29158GradingParameters::CellModulationGrade 2705
EMatrixCodeIso29158GradingParameters::EMatrixCodeIso29158GradingParameters 2705
EMatrixCodeIso29158GradingParameters::FixedPatternDamageGrade 2705
EMatrixCodeIso29158GradingParameters::IsMeanLightInRequiredBounds 2705
EMatrixCodeIso29158GradingParameters::MeanLight 2706
EMatrixCodeIso29158GradingParameters::MinimumReflectanceGrade 2706
EMatrixCodeIso29158GradingParameters::OverallSymbolGrade 2706
EMatrixCodeIso29158GradingParameters::ScanGrade 2706
5.31. EMatrixCodeSemiT10GradingParameters Struct 2706
EMatrixCodeSemiT10GradingParameters::CellDefects 2707
EMatrixCodeSemiT10GradingParameters::DataMatrixCellHeight 2707
EMatrixCodeSemiT10GradingParameters::DataMatrixCellWidth 2708
EMatrixCodeSemiT10GradingParameters::EMatrixCodeSemiT10GradingParameters 2708
EMatrixCodeSemiT10GradingParameters::FinderPatternDefects 2708
EMatrixCodeSemiT10GradingParameters::HorizontalMarkGrowth 2708
EMatrixCodeSemiT10GradingParameters::HorizontalMarkMisplacement 2708
EMatrixCodeSemiT10GradingParameters::SymbolContrast 2709
EMatrixCodeSemiT10GradingParameters::SymbolContrastSNR 2709
EMatrixCodeSemiT10GradingParameters::UnusedErrorCorrection 2709
EMatrixCodeSemiT10GradingParameters::VerticalMarkGrowth 2709
EMatrixCodeSemiT10GradingParameters::VerticalMarkMisplacement 2709
5.32. EMatrixPosition Struct 2710
EMatrixPosition::EMatrixPosition 2710
EMatrixPosition::operator!= 2711
EMatrixPosition::operator== 2711
EMatrixPosition::X 2711
EMatrixPosition::Y 2711
5.33. EObjectData Struct 2712
EObjectData::Class 2712
EObjectData::IsHole 2713
EObjectData::IsSelected 2713
EObjectData::ObjNbHole 2713
EObjectData::ObjNbRun 2713
EObjectData::ObjNum 2714
5.34. EOCR2CharacterCandidate Struct 2714
EOCR2CharacterCandidate::Code 2714
EOCR2CharacterCandidate::EOCR2CharacterCandidate 2715
EOCR2CharacterCandidate::Score 2715
5.35. EPath Struct 2715
EPath::X 2715
EPath::Y 2716
5.36. EPeak Struct 2716
EPeak::Amplitude 2716
EPeak::Area 2716
EPeak::Center 2717
EPeak::Length 2717
EPeak::Start 2717
5.37. EPen Struct 2717
EPen::Brush 2718
EPen::EPen 2718
EPen::IsValid 2719
EPen::Load 2719
EPen::operator!= 2720
EPen::operator== 2720
EPen::Save 2720
EPen::Serialize 2721
EPen::Style 2721

EPen::Width 2721
5.38. EQRCodeAdditionalParametersGrades Struct 2722
EQRCodeAdditionalParametersGrades::EQRCodeAdditionalParametersGrades 2722
EQRCodeAdditionalParametersGrades::FormatInformationGrade 2722
EQRCodeAdditionalParametersGrades::VersionInformationGrade 2723
5.39. EQRCodeIso15415GradingParameters Struct 2723
EQRCodeIso15415GradingParameters::AdditionalParametersGrades 2724
EQRCodeIso15415GradingParameters::AxialNonUniformity 2724
EQRCodeIso15415GradingParameters::AxialNonUniformityGrade 2724
EQRCodeIso15415GradingParameters::DecodingGrade 2724
EQRCodeIso15415GradingParameters::EQRCodeIso15415GradingParameters 2725
EQRCodeIso15415GradingParameters::FixedPatternDamageGrade 2725
EQRCodeIso15415GradingParameters::GridNonUniformity 2725
EQRCodeIso15415GradingParameters::GridNonUniformityGrade 2725
EQRCodeIso15415GradingParameters::HorizontalPrintGrowth 2725
EQRCodeIso15415GradingParameters::ModulationGrade 2726
EQRCodeIso15415GradingParameters::OverallSymbolGrade 2726
EQRCodeIso15415GradingParameters::ReflectanceMarginGrade 2726
EQRCodeIso15415GradingParameters::ScanGrade 2726
EQRCodeIso15415GradingParameters::SymbolContrast 2727
EQRCodeIso15415GradingParameters::SymbolContrastGrade 2727
EQRCodeIso15415GradingParameters::UnusedErrorCorrection 2727
EQRCodeIso15415GradingParameters::UnusedErrorCorrectionGrade 2727
EQRCodeIso15415GradingParameters::VerticalPrintGrowth 2727
5.40. EQRCodeIso29158CalibrationParameters Struct 2728
EQRCodeIso29158CalibrationParameters::EQRCodeIso29158CalibrationParameters 2728
EQRCodeIso29158CalibrationParameters::MLcal 2728
EQRCodeIso29158CalibrationParameters::Rcal 2729
EQRCodeIso29158CalibrationParameters::SRcal 2729
EQRCodeIso29158CalibrationParameters::SRtarget 2729
5.41. EQRCodeIso29158GradingParameters Struct 2729
EQRCodeIso29158GradingParameters::CellContrastGrade 2730
EQRCodeIso29158GradingParameters::CellModulationGrade 2730
EQRCodeIso29158GradingParameters::EQRCodeIso29158GradingParameters 2730
EQRCodeIso29158GradingParameters::FixedPatternDamageGrade 2731
EQRCodeIso29158GradingParameters::IsMeanLightInRequiredBounds 2731
EQRCodeIso29158GradingParameters::MeanLight 2731
EQRCodeIso29158GradingParameters::MinimumReflectanceGrade 2731
EQRCodeIso29158GradingParameters::OverallSymbolGrade 2731
EQRCodeIso29158GradingParameters::ScanGrade 2732
5.42. ERenderStyle Struct 2732
ERenderStyle::ERenderStyle 2732
ERenderStyle::fillRGB 2733
ERenderStyle::hasFill 2733
ERenderStyle::hasLine 2733
ERenderStyle::lineRGB 2733
ERenderStyle::pointRGB 2734
5.43. ERGB Struct 2734
ERGB::B 2734
ERGB::G 2734
ERGB::R 2735
5.44. ERGBColor Struct 2735
ERGBColor::Blue 2735
ERGBColor::ERGBColor 2735
ERGBColor::Green 2736
ERGBColor::Red 2736
5.45. EROCPoint Struct 2736
EROCPoint::EROCPoint 2737

EROCPoint::FP 2738
EROCPoint::FPR 2738
EROCPoint::Load 2738
EROCPoint::N 2739
EROCPoint::P 2739
EROCPoint::Save 2739
EROCPoint::Threshold 2740
EROCPoint::TP 2740
EROCPoint::TPR 2740
5.46. ERun Struct 2740
ERun::ERun 2741
ERun::Length 2741
ERun::operator!= 2741
ERun::operator== 2742
ERun::OrgX 2742
ERun::Y 2742
5.47. ERunData Struct 2743
ERunData::Class 2743
ERunData::Len 2743
ERunData::ObjNum 2743
ERunData::OrgX 2744
ERunData::OrgY 2744
5.48. ESize Struct 2744
ESize::ESize 2745
ESize::Height 2745
ESize::operator!= 2745
ESize::operator== 2746
ESize::Width 2746
5.49. EVSH Struct 2746
EVSH::H 2746
EVSH::S 2747
EVSH::V 2747
5.50. EXYZ Struct 2747
EXYZ::X 2747
EXYZ::Y 2748
EXYZ::Z 2748
5.51. EYIQ Struct 2748
EYIQ::I 2748
EYIQ::Q 2748
EYIQ::Y 2749
5.52. EYSH Struct 2749
EYSH::H 2749
EYSH::S 2749
EYSH::Y 2750
5.53. EYUV Struct 2750
EYUV::U 2750
EYUV::V 2750
EYUV::Y 2751
6. Enumerations 2752
6.1. E3DAttribute Enum 2752
6.2. E3DObjectFeature Enum 2753
6.3. EAdaptiveThresholdMethod Enum 2754
6.4. EAlignmentPolarity Enum 2754
6.5. EAngleUnit Enum 2755
6.6. EArithmeticLogicOperation Enum 2755
6.7. EasyOCR2CharacterFilter Enum 2757
6.8. EasyOCR2CharSpacingBias Enum 2757

6.9. EasyOCR2CharWidthBias Enum 2758
6.10. EasyOCR2DrawDetectionStyle Enum 2758
6.11. EasyOCR2DrawRecognitionStyle Enum 2758
6.12. EasyOCR2DrawSegmentationStyle Enum 2759
6.13. EasyOCR2TextPolarity Enum 2759
6.14. EAttributeType Enum 2760
6.15. EAxisOriginMode Enum 2760
6.16. EAxisSystemType Enum 2761
6.17. EBarCodeSymbologies Enum 2761
6.18. EBayerConfiguration Enum 2763
6.19. EByteInterpretationMode Enum 2763
6.20. ECalibrationMode Enum 2764
6.21. ECalibrationType Enum 2764
6.22. ECannyThresholdingMode Enum 2765
6.23. ECC000Family Enum 2765
6.24. ECellColor Enum 2765
6.25. EClassifierCapacity Enum 2766
6.26. EClippingMode Enum 2766
6.27. ECodeType Enum 2766
6.28. EColorQuantization Enum 2767
6.29. EColorRampMode Enum 2767
6.30. EColorSystem Enum 2768
6.31. EComparisonDistanceMode Enum 2769
6.32. EConfusionMatrixElement Enum 2770
6.33. EConnexity Enum 2770
6.34. EContourMode Enum 2770
6.35. EContourThreshold Enum 2771
6.36. ECorrelationMode Enum 2771
6.37. EDatasetType Enum 2772
6.38. EDataSize Enum 2772
6.39. EDataType Enum 2772
6.40. EDeepLearningDeviceType Enum 2773
6.41. EDeepLearningInferencePrecision Enum 2773
6.42. EDeepLearningToolType Enum 2773
6.43. EDLDataAugmentationType Enum 2774
6.44. EDongleType Enum 2774
6.45. EDoubleThresholdMode Enum 2775
6.46. EDraggingMode Enum 2775
6.47. EDragHandle Enum 2775
6.48. EDrawableFeature Enum 2777
6.49. EDrawingMode Enum 2777
6.50. EEditionMode Enum 2778
6.51. EEncodingConnexity Enum 2778
6.52. EError Enum 2779
6.53. EFamily Enum 2810
6.54. EFeature Enum 2810
6.55. EFiducialMatchingMode Enum 2813
6.56. EFillUndefinedPixelsDirection Enum 2813
6.57. EFillUndefinedPixelsMethod Enum 2814
6.58. EFilteringMode Enum 2814
6.59. EFindContrastMode Enum 2814
6.60. EFlipAxis Enum 2815
6.61. EFlipping Enum 2815
6.62. EFontStyle Enum 2816
6.63. EFramePosition Enum 2816

6.64. EFrequentialDomainFormat Enum 2816
6.65. EGrayscaleSingleThreshold Enum 2817
6.66. EHarrisThresholdingMode Enum 2817
6.67. EHeatmapColormap Enum 2817
6.68. EHistogramFeature Enum 2818
6.69. EHitAndMissValue Enum 2818
6.70. EImageAnnotationFormat Enum 2819
6.71. EImageFileType Enum 2819
6.72. EImageType Enum 2820
6.73. EKernelRectifier Enum 2820
6.74. EKernelRotation Enum 2821
6.75. EKernelType Enum 2821
6.76. ELearnParam Enum 2822
6.77. ELegacyFeature Enum 2823
6.78. ELineSpacingMode Enum 2825
6.79. ELocalSearchMode Enum 2826
6.80. ELocatorCapacity Enum 2826
6.81. ELocatorFeature Enum 2827
6.82. ELogicalSize Enum 2827
6.83. EMailBarcodeOrientation Enum 2829
6.84. EMailBarcodeSymbologies Enum 2829
6.85. EMapConversionMode Enum 2830
6.86. EMapConversionMode Enum 2830
6.87. EMatchContrastMode Enum 2830
6.88. EMatchingMode Enum 2831
6.89. EMatrixCodeContrastMode Enum 2831
6.90. EMaximumAnalysisMode Enum 2831
6.91. ENoiseRemovalMethod Enum 2832
6.92. ENormalizationMode Enum 2832
6.93. EObjectBasedCalibrationPrecisionVsSpeedTradeOff Enum 2832
6.94. EObjectBasedCalibrationType Enum 2833
6.95. EOCR2Classifier Enum 2833
6.96. EOCR2DetectionMethod Enum 2834
6.97. EOCR2SegmentationMethod Enum 2834
6.98. EOCRClass Enum 2834
6.99. EOCRColor Enum 2836
6.100. EPathVectorDrawOption Enum 2836
6.101. EPatternStyle Enum 2836
6.102. EPatternType Enum 2837
6.103. EPenStyle Enum 2837
6.104. EPhotometricStereoContrast Enum 2837
6.105. EPickingMode Enum 2838
6.106. EPlaneCropperType Enum 2838
6.107. EPlotItem Enum 2838
6.108. EPointCloudFilteringMethod Enum 2839
6.109. EPolygonMeasurementMode Enum 2839
6.110. EProjectionType Enum 2839
6.111. EQRCodeCodingMode Enum 2840
6.112. EQRCodeEncoding Enum 2840
6.113. EQRCodeLevel Enum 2841
6.114. EQRCodeModel Enum 2841
6.115. EQRCodeScanPrecision Enum 2841
6.116. EQRDetectionMethod Enum 2842
6.117. EQRDetectionTradeOff Enum 2842
6.118. EReadingOrientation Enum 2843

6.119. EReadMode Enum 2843
6.120. ERectangleMode Enum 2844
6.121. EReductionMode Enum 2844
6.122. EReferenceNoise Enum 2845
6.123. ERgbStandard Enum 2845
6.124. ERoiHit Enum 2845
6.125. ERotationRightAngles Enum 2846
6.126. ESegmentationMethod Enum 2846
6.127. ESegmentationMode Enum 2847
6.128. ESelectByPosition Enum 2847
6.129. ESelectionFlag Enum 2848
6.130. ESelectOption Enum 2848
6.131. ESerializerFileWriterMode Enum 2849
6.132. EShapeBehavior Enum 2849
6.133. EShapeType Enum 2850
6.134. EShiftingMode Enum 2850
6.135. ESingleThresholdMode Enum 2851
6.136. ESortDirection Enum 2851
6.137. ESortOption Enum 2851
6.138. ESourceColorMode Enum 2852
6.139. ESpotPolarity Enum 2852
6.140. ESpotType Enum 2852
6.141. EStockMeasurementUnit Enum 2853
6.142. ESupervisedSegmenterCapacity Enum 2853
6.143. ESymbologies Enum 2854
6.144. ESymbolPolarity Enum 2856
6.145. ETextLabelAlignment Enum 2856
6.146. EThinStructureMode Enum 2856
6.147. EThresholdMode Enum 2857
6.148. ETrainingMode Enum 2857
6.149. ETransitionChoice Enum 2857
6.150. ETransitionType Enum 2858
6.151. EUIAPI Enum 2858
6.152. EUnsupervisedScore Enum 2858
6.153. EUnsupervisedSegmenterCapacity Enum 2859
6.154. EVerbosity Enum 2859
6.155. EViewDirection Enum 2860
6.156. EZMapGeneratorResolutionXYMode Enum 2860
6.157. EZMapOrientationVectorMode Enum 2860
6.158. EZMapReferencePlaneMode Enum 2861
6.159. Features Enum 2861

1. Pixel Accessors

1.1. EBW8PixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EBW8PixelAccessor	-
GetPixel	-
SetPixel	-

EBW8PixelAccessor::EBW8PixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EBW8PixelAccessor(
 EROIBW8& roi
 )
```

Parameters

roi

-

EBW8PixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
OEV_UINT8 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EBW8PixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 OEV_UINT8 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

1.2. EBW16PixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EBW16PixelAccessor	-
GetPixel	-
SetPixel	-

EBW16PixelAccessor::EBW16PixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EBW16PixelAccessor(
   EROIBW16& roi
)
```

Parameters

roi

-

EBW16PixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
OEV_UINT16 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EBW16PixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 OEV_UINT16 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

1.3. EBW32PixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EBW32PixelAccessor	-
GetPixel	-
SetPixel	-

EBW32PixelAccessor::EBW32PixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EBW32PixelAccessor(
 EROIBW32& roi
 )
```

Parameters

roi

-

EBW32PixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
OEV_UINT32 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EBW32PixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 OEV_UINT32 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

1.4. EC15PixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EC15PixelAccessor	-
GetPixel	-
SetPixel	-

EC15PixelAccessor::EC15PixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EC15PixelAccessor(
 EROIC15& roi
 )
```

Parameters

roi

-

EC15PixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
EC15 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EC15PixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 EC15 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

1.5. EC16PixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EC16PixelAccessor	-
GetPixel	-
SetPixel	-

EC16PixelAccessor::EC16PixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EC16PixelAccessor(
 EROIC16& roi
 )
```

Parameters

roi

-

EC16PixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
EC16 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EC16PixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 EC16 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

1.6. EC24APixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EC24APixelAccessor	-
GetPixel	-
SetPixel	-

EC24APixelAccessor::EC24APixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EC24APixelAccessor(
 EROIC24A& roi
 )
```

Parameters

roi

-

EC24APixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
EC24A GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EC24APixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 EC24A value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

1.7. EC24PixelAccessor Class

-

Namespace: Euresys::Open_eVision_1_2

Methods

EC24PixelAccessor	-
GetPixel	-
SetPixel	-

EC24PixelAccessor::EC24PixelAccessor

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void EC24PixelAccessor(
 EROIC24& roi
 )
```

Parameters

roi

-

EC24PixelAccessor::GetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
EC24 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EC24PixelAccessor::SetPixel

-

Namespace: Euresys::Open_eVision_1_2

```
[C++]
void SetPixel(
 EC24 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

-

x

-

y

-

2. Common

2.1. Easy Class

Classes

"Easy Class" on page 324

2.2. Image and ROI Classes

Image Classes

"EImageBW1 Class" on page 1406
"EImageBW8 Class" on page 1416
"EImageBW16 Class" on page 1409
"EImageBW32 Class" on page 1412

"EImageC15 Class" on page 1419
"EImageC16 Class" on page 1421
"EImageC24 Class" on page 1423
"EImageC24A Class" on page 1426
"EImageC48 Class" on page 1428

ROI Classes

"EROIBW1 Class" on page 2181
"EROIBW8 Class" on page 2204
"EROIBW16 Class" on page 2186
"EROIBW32 Class" on page 2189

"EROIC15 Class" on page 2207
"EROIC16 Class" on page 2211
"EROIC24 Class" on page 2215
"EROIC24A Class" on page 2219
"EROIC48 Class" on page 2222

2.3. Region Classes

Classes

"ERegion Class" on page 2156
"ERectangleRegion Class" on page 2128
"EPolygonRegion Class" on page 2018
"ECircleRegion Class" on page 773
"EEllipseRegion Class" on page 1274

3. Libraries

3.1. Easy3D Library

Classes

EAffineTransformer
"E3DAxisDisplay Class" on page 152
E3DAxisSystem
E3DBox
E3DViewer
ECalibrationGenerator
"ECalibrationModel Class" on page 701
EColorRamp
EConverter
EDecimator
"EDepthMapToMeshConverter Class" on page 1249
"EDepthMapToPointCloudConverter Class" on page 1252
"EErrorStatistics Class" on page 1283
"EFeaturesAligner Class" on page 1312
"EFilters Class" on page 1317
"EMesh Class" on page 1662
"EMeshToZMapConverter Class" on page 1668
E3DPlane
"EPlaneCropper Class" on page 1877
"EPlaneFinder Class" on page 1880
"EPlaneFitter Class" on page 1892
"EPointCloud Class" on page 1906
"EPointCloudFactory Class" on page 1935
"EPointCloudStatistics Class" on page 1947
"EPointCloudToZMapConverter Class" on page 1953
"EPrincipalAxisExtractor Class" on page 2038
"ERandomDecimator Class" on page 2084
"ERectangularCropper Class" on page 2149
"EScaleCalibrationModel Class" on page 2236
"ESimpleCropper Class" on page 2275
"ESphericalCropper Class" on page 2280
"EStatistics Class" on page 2304
EUtils
"EZMapToPointCloudConverter Class" on page 2643

Structs

E3DPoint
EDepth8
EDepth16
EDepth32f
"ERenderStyle Struct" on page 2732

Enumerations

"E3DAttribute Enum" on page 2752
EAlignmentPolarity
"EAttributeType Enum" on page 2760
"EAxisOriginMode Enum" on page 2760
"EColorRampMode Enum" on page 2767
EMaximumAnalysisMode
ENoiseRemovalMethod
EObjectBasedCalibrationPrecisionVsSpeedTradeOff
EObjectBasedCalibrationType
EPlaneCropperType
"EProjectionType Enum" on page 2839
EZMapOrientationVectorMode
EZMapReferencePlaneMode

3.2. Easy3DLaserLine Library

Classes

"EExplicitGeometricCalibrationModel Class" on page 1289
"EObjectBasedCalibrationGenerator Class" on page 1689
"EObjectBasedCalibrationModel Class" on page 1699
"ELaserLineExtractor Class" on page 1469

3.3. Easy3DObject Library

Classes

"E3DObject Class" on page 205
"E3DObjectExtractor Class" on page 215

3.4. Easy3DMatch Library

Classes

"E3DAligner Class" on page 138
"E3DAlignment Class" on page 148
"E3DAnomaly Class" on page 150
"E3DComparer Class" on page 173
"E3DMatch Class" on page 188
"E3DMatcher Class" on page 189
"EPointCloudMerger Class" on page 1942

Enumerations

"EComparisonDistanceMode Enum" on page 2769

3.5. EasyImage Library

Classes

EasyImage
EKernel
EMovingAverage

Enumerations

EArithmeticLogicOperation
EContourMode
EContourThreshold
EHistogramFeature
EKernelRectifier
EKernelRotation
EKernelType
EReferenceNoise
EThresholdMode

3.6. EasyColor Library

Classes

EasyColor
EColorLookup
EPseudoColorLookup

Enumerations

EColorQuantization
EColorSystem
ERgbStandard

3.7. EasyObject Library

Classes

EasyObject
ECodedImage2
ECodedElement
EObject
EHole
EObjectSelection
EImageEncoder
EImageSegmenter
ETwoLayersImageSegmenter
EThreeLayersImageSegmenter
EBinaryImageSegmenter
EGrayscaleSingleThresholdSegmenter
EGrayscaleDoubleThresholdSegmenter
EColorSingleThresholdSegmenter
EColorRangeThresholdSegmenter
EImageRangeSegmenter
EReferenceImageSegmenter
ELabeledImageSegmenter
EObjectRunsIterator
EObjectTemplateMatcher

Enumerations

EEncodingConnexity
ESegmentationMethod
ESingleThresholdMode
EDoubleThresholdMode
EFeature

3.8. EasyMatch Library

Classes

EMatcher

Structs

EMatchPosition

Enumerations

ECorrelationMode
EMatchContrastMode
EFilteringMode

3.9. EasyFind Library

Classes

EFoundPattern
EPatternFinder
"EFindFeaturePoint Class" on page 1322

Enumerations

EFindContrastMode
ELocalSearchMode
EPatternType
EReductionMode
EThinStructureMode

3.10. EasyGauge Library

Classes

ECircleGauge
ELineGauge
EPointGauge
ERectangleGauge
EWedgeGauge
EFrameShape
EWorldShape

Enumerations

EClippingMode
EPlotItem
ETransitionChoice
ETransitionType

3.11. EasyOCR Library

Classes

EOCR

Enumerations

EMatchingMode
EShiftingMode
EOCRClass
EOCRColor
ESegmentationMode

3.12. EasyOCR2 Library

Classes

EOCR2
EOCR2Text
EOCR2Line
EOCR2Word
EOCR2Char

Structs

EOCR2CharacterCandidate

Enumerations

EasyOCR2CharacterFilter
EasyOCR2CharSpacingBias
EasyOCR2CharWidthBias
EasyOCR2DrawDetectionStyle
EasyOCR2DrawRecognitionStyle
EasyOCR2DrawSegmentationStyle
EasyOCR2TextPolarity

3.13. EasyBarCode Library

Classes

EBarCode
EMailBarcode

Enumerations

EMailBarcodeSymbologies
EMailBarcodeOrientation

3.14. EasyBarCode2 Library

Classes

EBarCode2
EBarCodeReader2

Enumerations

EBarCodeSymbologies

3.15. EasyMatrixCode Library

Classes

EMatrixCode
EMatrixCodeReader
ESearchParamsType

Enumerations

EFamily
EFlipping
ELearnParam
ELogicalSize
EMatrixCodeContrastMode

3.16. EasyMatrixCode2 Library

Classes

EMatrixCode2
EMatrixCode2Reader

Enumerations

"EReadMode Enum" on page 2843

3.17. EasyQRCode Library

Classes

EQRCode
EQRCodeDecodedStream
EQRCodeDecodedStreamPart
EQRCodeGeometry
EQRCodeReader
EQuadrangle

Enumerations

EQRCodeCodingMode
EQRCodeEncoding
EQRCodeLevel
EQRCodeModel
EQRCodeScanPrecision

3.18. EasyClassify Library

Classes

"EClassificationDataset Class" on page 795
"EClassificationMetrics Class" on page 857
"EClassificationResult Class" on page 865
"EClassifier Class" on page 873
"EDataAugmentation Class" on page 1035
"EDatasetSplit Class" on page 1055
"EDeepLearningTool Class" on page 1130

Enumerations

"EDatasetType Enum" on page 2772

3.19. EasySegment Library

Classes

"EClassificationDataset Class" on page 795
"EDataAugmentation Class" on page 1035
"EDatasetSplit Class" on page 1055
"EUnsupervisedSegmenterMetrics Class" on page 2372
"EUnsupervisedSegmenterResult Class" on page 2379
"EUnsupervisedSegmenter Class" on page 2362
"ESupervisedSegmenter Class" on page 2319
"ESupervisedSegmenterMetrics Class" on page 2332
"ESupervisedSegmenterResult Class" on page 2345
"EDeepLearningTool Class" on page 1130
"EDeepLearningDefectDetectionMetrics Class" on page 1080

Structs

"EROCPoint Struct" on page 2736

Enumerations

"EUnsupervisedSegmenterCapacity Enum" on page 2859
"ESupervisedSegmenterCapacity Enum" on page 2853
"EConfusionMatrixElement Enum" on page 2770
"EDatasetType Enum" on page 2772

3.20. EasyLocate Library

Classes

"EClassificationDataset Class" on page 795
"EDataAugmentation Class" on page 1035
"EDatasetSplit Class" on page 1055

EInterestPointLocator

"ELocator Class" on page 1517
ELocatorBase
"ELocatorResult Class" on page 1559
"ELocatorMetrics Class" on page 1531
"ELocatorObject Class" on page 1548
"ELocatorPredictedObject Class" on page 1556

Enumerations

"ELocatorCapacity Enum" on page 2826
"EDatasetType Enum" on page 2772

3.21. Legacy

EasyObject Library (Legacy)

Classes

ECodedImage

Enumerations

EConnexity
ELegacyFeature
ESelectByPosition
ESelectOption
ESortOption

Functions

EasyObject::ContourArea
EasyObject::ContourGravityCenter
EasyObject::ContourInertia

4. Classes

4.1. E3DAligner Class

Aligns an EPointCloud or EZMap on a reference EMesh, EPointCloud or EZMap.

Derived Class(es):E3DMatcher
Namespace: Euresys::Open_eVision::Easy3D

Methods

Align	Aligns an EZMap or EPointCloud on the reference model.
ClearReprojectionPlane	Clears re-projection plane set with E3DAligner::ScanReprojectionPlane or E3DAligner::SetFlatScan.
E3DAligner	Constructs a 3D Aligner.
GetScanReprojectionPlane	Sets/Gets the plane on which the model lays. This is used to re-project the scans before trying to find the position of the object within them. The E3DPlane should lay slightly below/above (depending on the orientation of the normal) the real plane, so that all points of the real plane are reprojected in the new ZMap.
IsReprojectionPlaneSet	Returns whether a re-projection plane was set with E3DAligner::ScanReprojectionPlane or E3DAligner::SetFlatScan or not.
Load	Loads the E3DAligner. The given ESerializer must have been created for reading.
operator=	Assignment operator.
RetrieveReferencePoses	Retrieves the reference poses of the E3DAligner in the form of a vector of E3DPlane. The corresponding reference patterns vector can be retrieved with E3DAligner::RetrieveReferencePosesProjections.
RetrieveReferencePosesProjections	Retrieves the reference patterns of the E3DAligner in the form of a vector of EZMap8. The corresponding reference poses vector can be retrieved with E3DAligner::RetrieveReferencePoses.
Save	Saves the E3DAligner. The given ESerializer must have been created for writing.
SetFlatScan	Uses a flat scan of the setup to compute the plane on which the object lays. This helps computing alignment when the sensor does not lay on top of the objects.
SetReference	Sets the 3D reference model that is used to create reference patterns. It may be a CAD of the object as an EMesh with reference plane(s) on which they must be projected to roughly correspond to the face visible in the scans. In that case, when a scan re-projection plane is set (E3DAligner::ScanReprojectionPlane, E3DAligner::SetFlatScan), the reference plane(s) set by this method should correspond to the scan's re-projection plane. It may also be a golden scan represented by an EZMap or an EPointCloud with reference plane indicating on which direction it must be projected.
SetScanReprojectionPlane	Sets/Gets the plane on which the model lays. This is used to re-project the scans before trying to find the position of the object within them. The E3DPlane should lay slightly below/above (depending on the orientation of the normal) the real plane, so that all points of the real plane are reprojected in the new ZMap.

E3DAligner::Align

Aligns an EZMap or EPointCloud on the reference model.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DAlignment Align(
 const EZMap& zmap
 )

E3DAlignment Align(
 const EPointCloud& cloud,
 const E3DPlane& projectionPlane
 )

E3DAlignment Align(
 const EPointCloud& cloud,
 float azimuth,
 float elevation
 )
```

Parameters

zmap

The EZMap to align

cloud

The EPointCloud to align on the reference

projectionPlane

The E3DPlane on which the cloud is orthographically projected.

azimuth

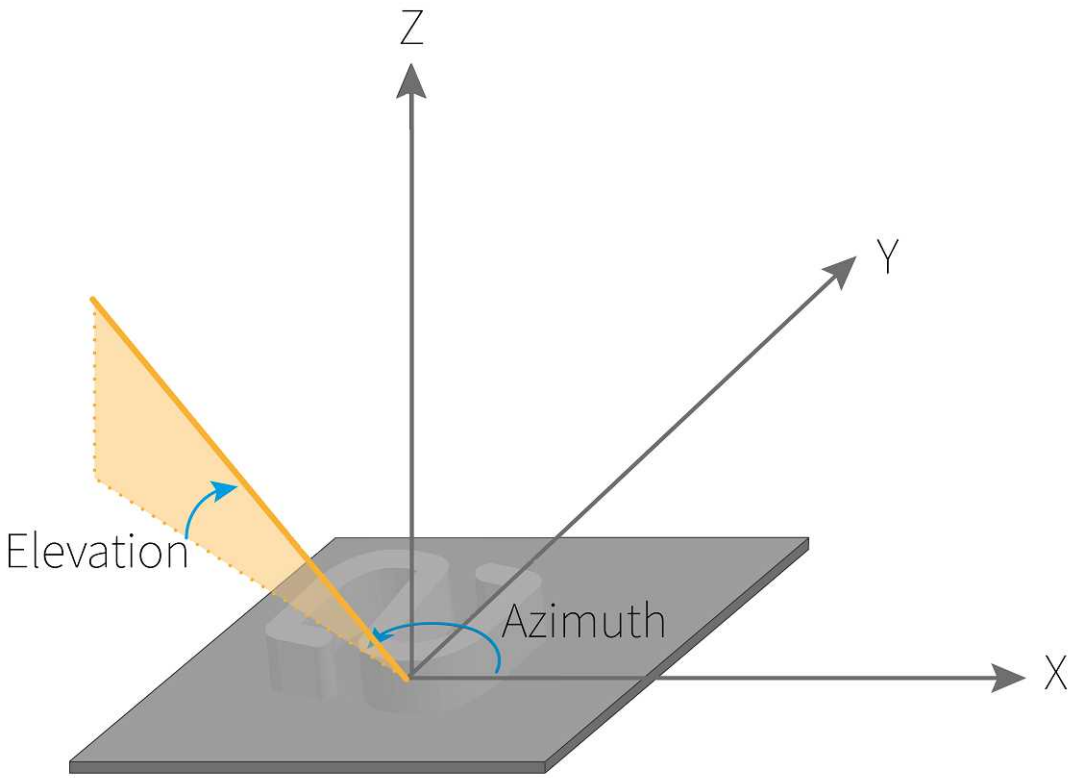
The azimuth angle of the normal of the projection plane in Easy::AngleUnit. Azimuth angles are oriented trigonometrically around the z axis. The x axis corresponds to an azimuth of 0 degrees.

elevation

The elevation angle of the normal of the projection plane in Easy::AngleUnit. Elevation angles represent how close the normal is to the z = 0 plane.

Remarks

When specifying azimuth and elevation, only the normal of the projection plane is specified. The distance from origin of the E3DPlane will be computed from the points cloud, so that all points of the cloud are visible. This might not be wanted if there are points in the cloud that are useless for the process (e.g. if we see other objects far below the model). This is also a bit slower as the plane's distance is recomputed on each scan.



E3DAligner::ClearReprojectionPlane

Clears re-projection plane set with E3DAligner::ScanReprojectionPlane or E3DAligner::SetFlatScan.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ClearReprojectionPlane(
 )
```

E3DAligner::E3DAligner

Constructs a 3D Aligner.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DAligner(
 )
void E3DAligner(
 const E3DAligner& other
 )
```

Parameters

other

Another E3DAligner object to be copied in the new E3DAligner object.

E3DAligner::IsReprojectionPlaneSet

Returns whether a re-projection plane was set with E3DAligner::ScanReprojectionPlane or E3DAligner::SetFlatScan or not.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsReprojectionPlaneSet(
 )
```

E3DAligner::Load

Loads the E3DAligner. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DAligner::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DAligner& operator=(
 const E3DAligner& other
 )
```

Parameters

other

The E3DAligner object that should be assigned.

E3DAligner::RetrieveReferencePoses

Retrieves the reference poses of the E3DAligner in the form of a vector of E3DPlane. The corresponding reference patterns vector can be retrieved with E3DAligner::RetrieveReferencePosesProjections.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void RetrieveReferencePoses(
 std::vector<Euresys::Open_eVision::Easy3D::E3DPlane>& referencePoses
 )
```

Parameters

referencePoses

The vector that will be filled with copies of the reference poses

E3DAligner::RetrieveReferencePosesProjections

Retrieves the reference patterns of the E3DAligner in the form of a vector of EZMap8. The corresponding reference poses vector can be retrieved with E3DAligner::RetrieveReferencePoses.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void RetrieveReferencePosesProjections(
 std::vector<Euresys::Open_eVision::Easy3D::EZMap8>& referencePosesProjections
 )
```

Parameters

referencePosesProjections

The vector that will be filled with copies of the reference poses projections

E3DAligner::Save

Saves the E3DAligner. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DAligner::GetScanReprojectionPlane

E3DAligner::SetScanReprojectionPlane

Sets/Gets the plane on which the model lays. This is used to re-project the scans before trying to find the position of the object within them. The E3DPlane should lay slightly below/above (depending on the orientation of the normal) the real plane, so that all points of the real plane are reprojected in the new ZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane GetScanReprojectionPlane() const
void SetScanReprojectionPlane(const E3DPlane& plane)
```

Remarks

When the sensor is not directly on top of the object, setting the re-projection plane will improve results, you can also give a flat scan on which the normal of the plane is identified automatically, see E3DAligner::SetFlatScan. When the reference is a EMesh, the corresponding reference plane should match the re-projection plane.

E3DAligner::SetFlatScan

Uses a flat scan of the setup to compute the plane on which the object lays. This helps computing alignment when the sensor does not lay on top of the objects.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFlatScan(
 const EZMap& scan
 )

void SetFlatScan(
 const EZMap& scan,
 bool objectAbovePlane
 )

void SetFlatScan(
 const EPointCloud& cloud
 )

void SetFlatScan(
 const EPointCloud& cloud,
 bool objectAbovePlane
 )
```

Parameters

scan

The EZMap representing a scan of the setup when no object lays on it.

objectAbovePlane

Whether the object to align lays above or below the plane of the scan. To project the object, we must not only know the plane but also if we must project the points above or below it. The object is said to lay above the plane, if, for a given (x, y), z(object) > z (plane). Default: true

cloud

The EPointCloud representing a scan of the setup when no object lays on it.

Remarks

When the sensor is not directly on top of the object, setting the plane improves the results, you can also give the plane coordinates directly, see E3DAligner::ScanReprojectionPlane. When the reference is a EMesh, the corresponding reference plane should match the re-projection plane.

E3DAligner::SetReference

Sets the 3D reference model that is used to create reference patterns. It may be a CAD of the object as an EMesh with reference plane(s) on which they must be projected to roughly correspond to the face visible in the scans. In that case, when a scan re-projection plane is set (E3DAligner::ScanReprojectionPlane, E3DAligner::SetFlatScan), the reference plane(s) set by this method should correspond to the scan's re-projection plane. It may also be a golden scan represented by an EZMap or an EPointCloud with reference plane indicating on which direction it must be projected.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetReference(
 const EMesh& mesh,
 const E3DPlane& plane
 )

void SetReference(
 const EMesh& mesh,
 float azimuth,
 float elevation
 )

void SetReference(
 const EMesh& mesh,
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPlane>& planes
 )

void SetReference(
 const EMesh& mesh,
 const std::vector<float>& azimuths,
 const std::vector<float>& elevations
 )

void SetReference(
 const EZMap& zmap
 )

void SetReference(
 const EPointCloud& cloud,
 const E3DPlane& plane
 )

void SetReference(
 const EPointCloud& cloud,
 float azimuth,
 float elevation
 )
```

Parameters

mesh

The EMesh object that represents the 3D reference model.

plane

The plane onto which the model is projected orthographically (all points below the plane are discarded).

azimuth

The azimuth angle of the normal of the reference plane in Easy::AngleUnit. Azimuth angles are oriented trigonometrically around the z axis. The x axis corresponds to an azimuth of 0 degrees.

elevation

The elevation angle of the normal of the reference plane in Easy::AngleUnit. Elevation angles represent how close the normal is to the z = 0 plane.

planes

vector of planes

azimuths

vector of azimuths

elevations

vector of elevations

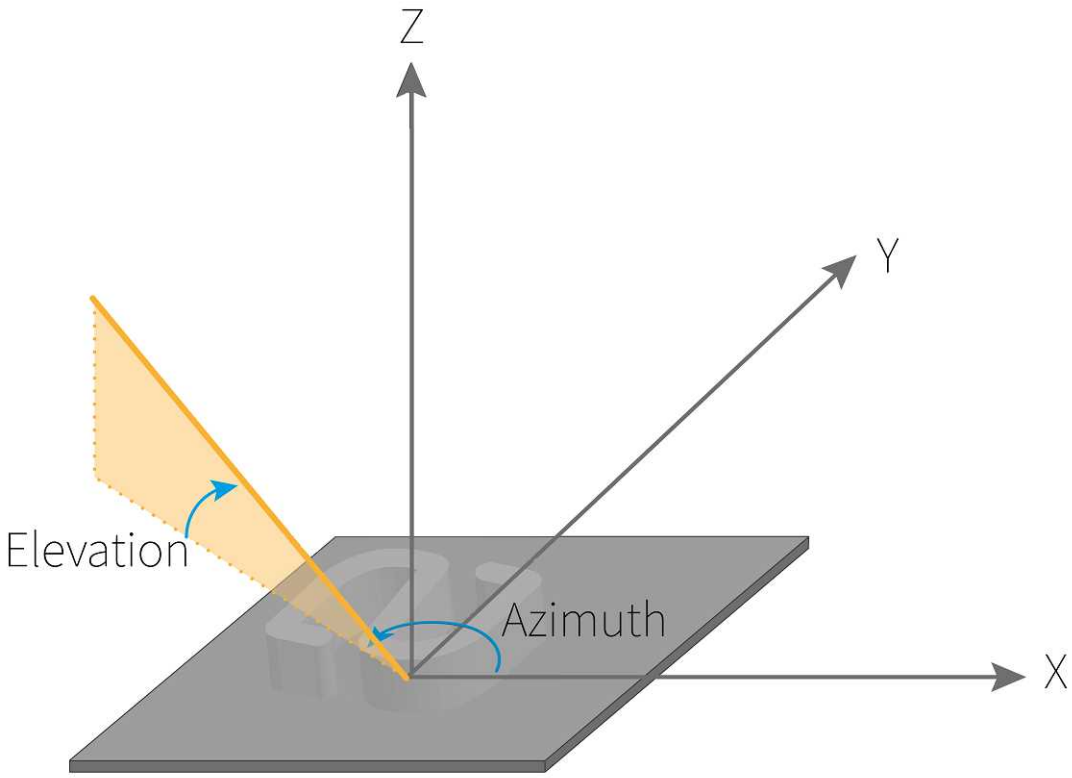
zmap

The EZMap object that represents a reference golden scan. The scan should only contain the object to align (the plane on which the points lay should be removed).

cloud

The EPointCloud object that represents a reference golden scan. The scan should only contain the object to align (the plane on which the points lay should be removed).

Remarks



4.2. E3DAlignment Class

Represents a 3D Alignment returned by E3DAligner.

Derived Class(es):E3DMatch

Namespace: Euresys::Open_eVision::Easy3D

Methods

E3DAlignment	Constructs an E3DAlignment.
GetError	Gets the E3DAlignment error. The lower the error, the better the alignment. The error represents the average euclidean distance between the points of the reference and the scan without taking outliers into account.
GetPose	Gets the pose of the E3DAlignment. The pose is an E3DTransformMatrix to apply to the scan to align it on the reference.
GetRefPoseMatchedIndex	Gets the reference pose index to which the scan was matched. These reference poses can be obtained by using E3DAligner::RetrieveReferencePosesProjections.
operator=	Assignment operator.

E3DAlignment::E3DAlignment

Constructs an E3DAlignment.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DAlignment(
 )
void E3DAlignment(
 const E3DAlignment& other
 )
```

Parameters

other

Another E3DAlignment object to be copied in the new E3DAlignment object.

E3DAlignment::GetError

Gets the E3DAlignment error. The lower the error, the better the alignment. The error represents the average euclidean distance between the points of the reference and the scan without taking outliers into account.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetError() const
```

E3DAlignment::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DAlignment& operator=(
 const E3DAlignment& other
 )
```

Parameters

other

The E3DAlignment object that should be copied.

E3DAlignment::GetPose

Gets the pose of the E3DAlignment. The pose is an E3DTransformMatrix to apply to the scan to align it on the reference.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetPose() const
```

E3DAlignment::GetRefPoseMatchedIndex

Gets the reference pose index to which the scan was matched. These reference poses can be obtained by using E3DAligner::RetrieveReferencePosesProjections.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetRefPoseMatchedIndex() const
```

4.3. E3DAnomaly Class

Represents a detected 3D anomaly.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Create3DObject	Creates an E3DObject from the E3DAnomaly to render it in an E3DViewer. This method is deprecated, you can now retrieve the bounding box with E3DAnomaly::BoundingBox and display it directly in the E3DViewer.
E3DAnomaly	Constructs an E3DAnomaly.
GetArea	Gets the area of the E3DAnomaly.
GetBoundingBox	Gets the bounding box of the E3DAnomaly.
GetCenterOfGravity	Gets the center of gravity of the E3DAnomaly.
GetCloud	Gets the EPointCloud containing the points belonging to the E3DAnomaly as well as their distance to the scan.
operator=	Assignment operator.

E3DAnomaly::GetArea

Gets the area of the E3DAnomaly.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetArea() const
```

E3DAnomaly::GetBoundingBox

Gets the bounding box of the E3DAnomaly.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DBox GetBoundingBox() const
```

E3DAnomaly::GetCenterOfGravity

Gets the center of gravity of the E3DAnomaly.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetCenterOfGravity() const
```

E3DAnomaly::GetCloud

Gets the EPointCloud containing the points belonging to the E3DAnomaly as well as their distance to the scan.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloud GetCloud() const
```

E3DAnomaly::Create3DObject

This method is deprecated.

Creates an E3DObject from the E3DAnomaly to render it in an E3DViewer. This method is deprecated, you can now retrieve the bounding box with E3DAnomaly::BoundingBox and display it directly in the E3DViewer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DObject Create3DObject(
 )
```

E3DAnomaly::E3DAnomaly

Constructs an E3DAnomaly.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DAnomaly(
 )
void E3DAnomaly(
 const E3DAnomaly& other
 )
```

Parameters

other

Another E3DAnomaly object to be copied in the new E3DAnomaly object.

E3DAnomaly::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DAnomaly& operator=(
 const E3DAnomaly& other
 )
```

Parameters

other

The E3DAnomaly object that should be copied.

4.4. E3DAxisDisplay Class

Represents the axis and the grid to display in the E3DViewer.

Namespace: Euresys::Open_eVision::Easy3D

Methods

E3DAxisDisplay	Creates an E3DAxisDisplay object.
GetAxisGraduationColor	Sets/Gets the axis graduation color.
GetAxisOrigin	Set and Get the axis origin mode.
GetAxisOriginUserDefined	Set and Get the axis origin.
GetAxisSize	Sets/Gets the size of each axis.
GetAxisXColor	Sets/Gets the color of the X axis.
GetAxisYColor	Sets/Gets the color of the Y axis.
GetAxisZColor	Sets/Gets the color of the Z axis.
GetGridColor	Sets/Gets the grid color.
GetRenderAxis	Enables or disables the display of the axis.
GetRenderGrid	Enables or disables the display of Grid.
GetRenderGridStep	Sets/Gets the grid step of each axis.
operator=	Assignment operator.
operator==	Comparison operator.
SetAxisGraduationColor	Sets/Gets the axis graduation color.
SetAxisOrigin	Set and Get the axis origin mode.
SetAxisOriginUserDefined	Set and Get the axis origin.
SetAxisSize	Sets/Gets the size of each axis.
SetAxisXColor	Sets/Gets the color of the X axis.
SetAxisYColor	Sets/Gets the color of the Y axis.
SetAxisZColor	Sets/Gets the color of the Z axis.
SetGridColor	Sets/Gets the grid color.
SetRenderAxis	Enables or disables the display of the axis.
SetRenderGrid	Enables or disables the display of Grid.
SetRenderGridStep	Sets/Gets the grid step of each axis.

E3DAxisDisplay::GetAxisGraduationColor

E3DAxisDisplay::SetAxisGraduationColor

Sets/Gets the axis graduation color.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ERGBColor GetAxisGraduationColor() const
void SetAxisGraduationColor(ERGBColor color)
```

Remarks

The default color is white.

E3DAxisDisplay::GetAxisOrigin

E3DAxisDisplay::SetAxisOrigin

Set and Get the axis origin mode.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::EAxisOriginMode GetAxisOrigin() const
void SetAxisOrigin(Euresys::Open_eVision::Easy3D::EAxisOriginMode mode)
```

Remarks

The default mode is EAxisOriginMode.

E3DAxisDisplay::GetAxisOriginUserDefined

E3DAxisDisplay::SetAxisOriginUserDefined

Set and Get the axis origin.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetAxisOriginUserDefined() const
void SetAxisOriginUserDefined(const E3DPoint& origin)
```

Remarks

This function also sets the axis origin mode to EAxisOriginMode.

E3DAxisDisplay::GetAxisSize

E3DAxisDisplay::SetAxisSize

Sets/Gets the size of each axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetAxisSize() const
void SetAxisSize(const E3DPoint& size)
```

Remarks

The unit is the same as the one from the EPointCloud. Default value is the size of EPointCloud rounded up.

E3DAxisDisplay::GetAxisXColor

E3DAxisDisplay::SetAxisXColor

Sets/Gets the color of the X axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ERGBColor GetAxisXColor() const
void SetAxisXColor(ERGBColor color)
```

Remarks

The default color is red.

E3DAxisDisplay::GetAxisYColor

E3DAxisDisplay::SetAxisYColor

Sets/Gets the color of the Y axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ERGBColor GetAxisYColor() const
void SetAxisYColor(ERGBColor color)
```

Remarks

The default color is green.

E3DAxisDisplay::GetAxisZColor

E3DAxisDisplay::SetAxisZColor

Sets/Gets the color of the Z axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ERGBColor GetAxisZColor() const
void SetAxisZColor(ERGBColor color)
```

Remarks

The default color is blue.

E3DAxisDisplay::E3DAxisDisplay

Creates an E3DAxisDisplay object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DAxisDisplay(
 )
void E3DAxisDisplay(
 const E3DAxisDisplay& other
 )
```

Parameters

other

Reference to the E3DAxisDisplay used for the initialization.

E3DAxisDisplay::GetGridColor

E3DAxisDisplay::SetGridColor

Sets/Gets the grid color.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ERGBColor GetGridColor() const
void SetGridColor(ERGBColor color)
```

Remarks

The default color is grey.

E3DAxisDisplay::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DAxisDisplay& operator=(
 const E3DAxisDisplay& other
 )
```

Parameters

other

The E3DAxisDisplay object that should be copied.

E3DAxisDisplay::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const E3DAxisDisplay& other
 )
```

Parameters

other

The E3DAxisDisplay object to compare with.

E3DAxisDisplay::GetRenderAxis

E3DAxisDisplay::SetRenderAxis

Enables or disables the display of the axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetRenderAxis() const

void SetRenderAxis(bool state)
```

Remarks

The default state is true.

E3DAxisDisplay::GetRenderGrid

E3DAxisDisplay::SetRenderGrid

Enables or disables the display of Grid.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetRenderGrid() const
void SetRenderGrid(bool state)
```

Remarks

Display Grid with true (true by default).

E3DAxisDisplay::GetRenderGridStep

E3DAxisDisplay::SetRenderGridStep

Sets/Gets the grid step of each axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetRenderGridStep() const
void SetRenderGridStep(const E3DPoint& value)
```

Remarks

The unit of the grid step value is the same as the one from the EPointCloud. If value is equal to 0, then the step is auto computed and if the value is smaller than 0, then there is no step on the axis. Default value is the size of the EPointCloud rounded up and divided by ten.

4.5. E3DAxisSystem Class

Represent a 3D base axis system.

Derived Class(es):E3DOrthonormalAxisSystemE3DRightOrthonormalAxisSystem

Namespace: Euresys::Open_eVision::Easy3D

Methods

CheckIsNormal	check if axis system is Normed
CheckIsOrthogonal	check if axis system is Orthogonal
CheckIsRightHanded	check if axis system is Right
E3DAxisSystem	Constructs an E3DAxisSystem.
GetAxisX	Gets the X axis.
GetAxisY	Gets the Y axis.
GetAxisZ	Gets the Z axis.
GetNormX	Gets the norm of the X axis.
GetNormY	Gets the norm of the Y axis.
GetNormZ	Gets the norm of the Z axis.
GetOrigin	Gets the origin.
IsNormal	return true if this base axis system is Normed
IsOrthogonal	return true if this base axis system is Orthogonal
IsRightHanded	return true if this base axis system is Right Handed
Load	Load the E3DAxisSystem configuration. The given ESerializer must have been created for reading.
operator!=	-
operator=	Assignment operator.
operator==	operator comparison
Save	Save the E3DAxisSystem configuration. The given ESerializer must have been created for writing.

E3DAxisSystem::GetAxisX

Gets the X axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetAxisX() const
```

E3DAxisSystem::GetAxisY

Gets the Y axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetAxisY() const
```

E3DAxisSystem::GetAxisZ

Gets the Z axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetAxisZ() const
```

E3DAxisSystem::CheckIsNormal

check if axis system is Normed

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool CheckIsNormal(
 const E3DPoint& axisX,
 const E3DPoint& axisY,
 const E3DPoint& axisZ
 )
```

Parameters

axisX

-

axisY

-

axisZ

-

E3DAxisSystem::CheckIsOrthogonal

check if axis system is Orthogonal

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool CheckIsOrthogonal(
 const E3DPoint& axisX,
 const E3DPoint& axisY,
 const E3DPoint& axisZ
 )
```

Parameters

axisX

-

axisY

-

axisZ

-

E3DAxisSystem::CheckIsRightHanded

check if axis system is Right

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool CheckIsRightHanded(
 const E3DPoint& axisX,
 const E3DPoint& axisY,
 const E3DPoint& axisZ
 )
```

Parameters

axisX

-

axisY

-

axisZ

-

E3DAxisSystem::E3DAxisSystem

Constructs an E3DAxisSystem.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DAxisSystem(
 )
```

```
void E3DAxisSystem(
 const E3DPoint& Origin,
 const E3DPoint& axisX,
 const E3DPoint& axisY,
 const E3DPoint& axisZ
 )
void E3DAxisSystem(
 const E3DAxisSystem& other
 )
```

Parameters

Origin
The origin of the axis system

axisX
The X axis

axisY
The Y axis

axisZ
The Z axis

other
Reference to another E3DAxisSystem used for the initialization.

E3DAxisSystem::IsNormal

return true if this base axis system is Normed

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsNormal(
 )
```

E3DAxisSystem::IsOrthogonal

return true if this base axis system is Orthogonal

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsOrthogonal(
 )
```

E3DAxisSystem::IsRightHanded

return true if this base axis system is Right Handed

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsRightHanded(
 )
```

E3DAxisSystem::Load

Load the E3DAxisSystem configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

E3DAxisSystem::GetNormX

Gets the norm of the X axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetNormX() const
```

E3DAxisSystem::GetNormY

Gets the norm of the Y axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetNormY() const
```

E3DAxisSystem::GetNormZ

Gets the norm of the Z axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetNormZ() const
```

E3DAxisSystem::operator!=

-

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator!=(
 const E3DAxisSystem& other
 )
```

Parameters

other

-

E3DAxisSystem::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DAxisSystem& operator=(
 const E3DAxisSystem& other
 )
```

Parameters

other

The source E3DAxisSystem.

E3DAxisSystem::operator==

operator comparison

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const E3DAxisSystem& other
 )
```

Parameters

other

-

E3DAxisSystem::GetOrigin

Gets the origin.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetOrigin() const
```

E3DAxisSystem::Save

Save the E3DAxisSystem configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.6. E3DBox Class

A 3D box, used as bounding volume for E3DObject class. A box is defined by a center, 3 axis and 3 extent for the 3 axis. A 3D point (x,y,z) is inside the E3DBox if ... By default a E3DBox is an axis aligned unit cube, centered on the origin.

Namespace: Euresys::Open_eVision::Easy3D

Methods

E3DBox	Constructs an E3DBox. By default a E3DBox is an axis aligned unit cube, centered on the origin.
GetAxes	3D orthonormal axis system of the box.
GetCenter	3D box center.
GetXAxis	X axis of the box.
GetXSize	Size of the 3D box along its X axis .
GetXYQuadrangle	2D quadrangle of the E3DBox in the XY plane
GetYAxis	Y axis of the box.
GetYSize	Size of the 3D box along its Y axis.
GetZAxis	Z axis of the box.
GetZSize	Size of the 3D box along its Z axis.
Load	Loads the E3DBox. The given ESerializer must have been created for reading.
operator!=	Checks if two E3DBox are different
operator=	Assignment operator for the E3DBox.
operator==	Checks if two E3DBox are strictly equal
Save	Saves the E3DBox. The given ESerializer must have been created for writing.
SetAxes	3D orthonormal axis system of the box.
SetXSize	Size of the 3D box along its X axis .
SetYSize	Size of the 3D box along its Y axis.
SetZSize	Size of the 3D box along its Z axis.
Transform	Transforms the 3D box with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only).

E3DBox::GetAxes

E3DBox::SetAxes

3D orthonormal axis system of the box.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DOrthonormalAxisSystem GetAxes() const

void SetAxes(const E3DOrthonormalAxisSystem& axes)
```

E3DBox::GetCenter

3D box center.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetCenter() const
```

E3DBox::E3DBox

Constructs an E3DBox. By default a E3DBox is an axis aligned unit cube, centered on the origin.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DBox(
 )

void E3DBox(
 float xSize,
 float ySize,
 float zSize
 )

void E3DBox(
 const E3DOrthonormalAxisSystem& axes,
 float xSize,
 float ySize,
 float zSize
 )
```

```
void E3DBox(
 const E3DBox& other
 )

void E3DBox(
 const E3DPoint& center,
 float xSize,
 float ySize,
 float zSize
 )

void E3DBox(
 const EFloatRange& xBounds,
 const EFloatRange& yBounds,
 const EFloatRange& zBounds
 )

void E3DBox(
 const E3DPoint& center,
 float roll,
 float pitch,
 float yaw,
 float xSize,
 float ySize,
 float zSize
 )
```

Parameters

xSize
The full size of the box along the X axis.

ySize
The full size of the box along the Y axis.

zSize
The full size of the box along the Z axis.

axes
Axis system.

other
Reference to another E3DBox used for the initialization.

center
The 3D coordinate of the box center.

xBounds
The bounds of box along the X axis.

yBounds
The bounds of box along the Y axis.

zBounds
The bounds of box along the Z axis.

roll
Roll (rotation along the X axis) of the box.

pitch
Pitch (rotation along the Y axis) of the box.
yaw
Yaw (rotation along the Z axis) of the box.

Remarks

By convention, roll is applied first and yaw last.

E3DBox::Load

Loads the E3DBox. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.
serializer
The serializer.

E3DBox::operator!=

Checks if two E3DBox are different

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator!=(
 const E3DBox& other
 )
```

Parameters

other
-

E3DBox::operator=

Assignment operator for the E3DBox.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DBox& operator=(
 const E3DBox& other
 )
```

Parameters

other

-

E3DBox::operator==

Checks if two E3DBox are strictly equal

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const E3DBox& other
 )
```

Parameters

other

-

E3DBox::Save

Saves the E3DBox. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DBox::Transform

Transforms the 3D box with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DBox Transform(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix

The 3D transformation matrix.

E3DBox::GetXAxis

X axis of the box.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetXAxis() const
```

E3DBox::GetXSize

E3DBox::SetXSize

Size of the 3D box along its X axis .

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetXSize() const
void SetXSize(float xSize)
```

E3DBox::GetXYQuadrangle

2D quadrangle of the E3DBox in the XY plane

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EQuadrangle GetXYQuadrangle() const
```

E3DBox::GetYAxis

Y axis of the box.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetYAxis() const
```

E3DBox::GetYSize

E3DBox::SetYSize

Size of the 3D box along its Y axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetYSize() const
void SetYSize(float ySize)
```

E3DBox::GetZAxis

Z axis of the box.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetZAxis() const
```

E3DBox::GetZSize

E3DBox::SetZSize

Size of the 3D box along its Z axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetZSize() const
void SetZSize(float zSize)
```

4.7. E3DComparer Class

Represents a 3D comparison context.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Compare	Compares the reference patterns against a scan.
ComputesAnomalies	Computes the anomalies and returns them as a vector of E3DAnomaly.
E3DComparer	Constructs a 3D matching context.
GetAnomalyHysteresis	Gets the hysteresis related to the anomalies detection.
GetAnomalyThresholds	Gets the thresholds related to the anomalies detection.
GetAutomaticCropFactor	Sets/Gets the automatic cropping factor. The factor multiplies the anomaly distance threshold (E3DComparer::SetAnomalyThresholds) to obtain the margin for the cropping. If the value is smaller than 0, then no crop is performed before computing the distance. Default: 1
GetComparisonDistanceMode	Sets/Gets the distance mode for the 3D comparison. All values of the enum are not allowed for E3DComparer. Default: EComparisonDistanceMode_Euclidean.
GetComparisonPointCloud	Gets the EPointCloud that is used for the comparison.
GetDontCare	Sets/Gets a vector of E3DBox that defines the regions that should not be compared. By default, no point of the reference is ignored.
GetEdgeCroppingParameters	Gets the parameters driving the edge cropping.
GetEnableAutomaticDecimation	If True, the given reference is automatically decimated with a resolution depending on the distance threshold to speed-up processing. If False, all the points of the given reference are used to compute the distance. Default: True
GetEnableAutomaticEdgeCropping	If True, edges in the reference will be cropped automatically. This is useful to avoid false positives that could occur near the edges of the object, in particular when using normals (see E3DComparer and EComparisonDistanceMode). To modify the parameters of the cropping, see E3DComparer::SetEdgeCroppingParameters. Default: False

GetNoExtraMaterial	Sets/Gets a vector of E3DBox that defines the regions where there should not be points in the scan far from the reference. By default, the points far from the reference are not considered as anomalies (if there also are some other points closer to the reference).
GetROI	Sets/Gets a vector of E3DBox that defines the regions that should be compared. By default, the entire reference is considered.
Load	Loads the E3DComparer. The given ESerializer must have been created for reading.
operator=	Assignment operator.
PrepareReference	Prepare the reference internal structures. By default, this is done on the first call to E3DComparer::Compare.
Save	Saves the E3DComparer. The given ESerializer must have been created for writing.
SetAnomalyHysteresis	Sets the hysteresis related to the anomalies detection.
SetAnomalyThresholds	Sets the thresholds related to the anomalies detection.
SetAutomaticCropFactor	Sets/Gets the automatic cropping factor. The factor multiplies the anomaly distance threshold (E3DComparer::SetAnomalyThresholds) to obtain the margin for the cropping. If the value is smaller than 0, then no crop is performed before computing the distance. Default: 1
SetComparisonDistanceMode	Sets/Gets the distance mode for the 3D comparison. All values of the enum are not allowed for E3DComparer. Default: EComparisonDistanceMode_Euclidean.
SetDontCare	Sets/Gets a vector of E3DBox that defines the regions that should not be compared. By default, no point of the reference is ignored.
SetEdgeCroppingParameters	Sets the parameters driving the edge cropping.
SetEnableAutomaticDecimation	If True, the given reference is automatically decimated with a resolution depending on the distance threshold to speed-up processing. If False, all the points of the given reference are used to compute the distance. Default: True
SetEnableAutomaticEdgeCropping	If True, edges in the reference will be cropped automatically. This is useful to avoid false positives that could occur near the edges of the object, in particular when using normals (see E3DComparer and EComparisonDistanceMode). To modify the parameters of the cropping, see E3DComparer::SetEdgeCroppingParameters. Default: False
SetMeshReference	Sets the 3D reference model. See also E3DComparer::PointCloudReference.
SetNoExtraMaterial	Sets/Gets a vector of E3DBox that defines the regions where there should not be points in the scan far from the reference. By default, the points far from the reference are not considered as anomalies (if there also are some other points closer to the reference).
SetPointCloudReference	Sets the 3D reference model. See also E3DComparer::MeshReference.

SetROI	Sets/Gets a vector of E3DBox that defines the regions that should be compared. By default, the entire reference is considered.

E3DComparer::GetAutomaticCropFactor

E3DComparer::SetAutomaticCropFactor

Sets/Gets the automatic cropping factor. The factor multiplies the anomaly distance threshold (E3DComparer::SetAnomalyThresholds) to obtain the margin for the cropping. If the value is smaller than 0, then no crop is performed before computing the distance.
Default: 1

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetAutomaticCropFactor() const

void SetAutomaticCropFactor(float factor)
```

Remarks

Note that if the cropping margin is too small, the E3DPoint of the anomalies will not be visible (with the function E3DComparer::GetComparisonPointCloud and the parameter pointFromReference = False). Moreover, if the E3DComparer is set to False, the anomalies will not be detected.

E3DComparer::Compare

Compares the reference patterns against a scan.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Compare(
 const EPointCloud& scan
 )
```

Parameters

scan

An EPointCloud that represents the scan to compare to the reference.

E3DComparer::GetComparisonDistanceMode

E3DComparer::SetComparisonDistanceMode

Sets/Gets the distance mode for the 3D comparison. All values of the enum are not allowed for E3DComparer. Default: Euclidean.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EComparisonDistanceMode GetComparisonDistanceMode()
const

void SetComparisonDistanceMode(Euresys::Open_eVision::Easy3D::EComparisonDistanceMode
mode)
```

E3DComparer::ComputesAnomalies

Computes the anomalies and returns them as a vector of E3DAnomaly.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DAnomaly> ComputesAnomalies(
 )
```

E3DComparer::GetDontCare

E3DComparer::SetDontCare

Sets/Gets a vector of E3DBox that defines the regions that should not be compared. By default, no point of the reference is ignored.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DBox> GetDontCare() const

void SetDontCare(const std::vector<Euresys::Open_eVision::Easy3D::E3DBox>& dontCare)
```

E3DComparer::E3DComparer

Constructs a 3D matching context.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DComparer(
 )
void E3DComparer(
 const E3DComparer& other
 )
```

Parameters

other

Another E3DComparer object to be copied in the new E3DComparer object.

E3DComparer::GetEnableAutomaticDecimation

E3DComparer::SetEnableAutomaticDecimation

If True, the given reference is automatically decimated with a resolution depending on the distance threshold to speed-up processing. If False, all the points of the given reference are used to compute the distance.
Default: True

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableAutomaticDecimation() const
void SetEnableAutomaticDecimation(bool state)
```

E3DComparer::GetEnableAutomaticEdgeCropping

E3DComparer::SetEnableAutomaticEdgeCropping

If True, edges in the reference will be cropped automatically. This is useful to avoid false positives that could occur near the edges of the object, in particular when using normals (see E3DComparer and EComparisonDistanceMode). To modify the parameters of the cropping, see E3DComparer::SetEdgeCroppingParameters. Default: False

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableAutomaticEdgeCropping() const
void SetEnableAutomaticEdgeCropping(bool state)
```

E3DComparer::GetAnomalyHysteresis

Gets the hysteresis related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetAnomalyHysteresis(
 float& distanceHysteresisFactor,
 float& areaHysteresisFactor
 )
```

Parameters

distanceHysteresisFactor

The distance hysteresis factor. Must be greater than or equal to 1. Default: 1

areaHysteresisFactor

The area hysteresis factor. Must be smaller than or equal to 1. Default: 1.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See E3DComparer::SetAnomalyThresholds. In addition, an hysteresis can be specified, such that the cluster will also have to contain a subset of points whose distance is greater than distanceHysteresisFactor * distanceThreshold and area greater than areaHysteresisFactor * areaThreshold. See also E3DComparer::SetAnomalyThresholds.

E3DComparer::GetAnomalyThresholds

Gets the thresholds related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetAnomalyThresholds(
 float& distanceThreshold,
 float& areaThreshold
 )
```

Parameters

distanceThreshold

The distance threshold.

areaThreshold

The area threshold.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See also E3DComparer::SetAnomalyHysteresis to more advanced anomaly detection.

E3DComparer::GetComparisonPointCloud

Gets the EPointCloud that is used for the comparison.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetComparisonPointCloud(
 EPointCloud& cloudOut,
 bool storeDistances,
 bool storeColors,
 bool fullModel
 )
```

Parameters

cloudOut

Where to store the EPointCloud.

storeDistances

If set to True, the distances computed will be stored in the EPointCloud the functions EPointCloud::GetAttribute and EPointCloud::GetAttributeBuffer can be used with E3DAttribute_Distance to retrieve the distances.
Default: True

storeColors

If set to True, the colors related to the distances computed will be stored in the EPointCloud the functions EPointCloud::GetAttribute and EPointCloud::GetAttributeBuffer can be used with E3DAttribute_Color to retrieve the distances. The colors can also be used in the E3DViewer.
Default: True

fullModel

If set to True, the points that are not inside the ROI (see E3DComparer::ROI) will also be in the EPointCloud. Default: False

E3DComparer::GetEdgeCroppingParameters

Gets the parameters driving the edge cropping.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetEdgeCroppingParameters(
 OEV_UINT32& numberNeighbors,
 float& curvatureThreshold
 )
```

Parameters

numberNeighbors

The number of neighbors that we will use to compute the local curvature. The bigger this parameter, the thicker the cropping will be. This number must be bigger than 10. Default: 100

curvatureThreshold

The curvature value above which a point is cropped (must be between 0 and 1). Default: 0.15

Remarks

Edge cropping is disabled by default, enable it using E3DComparer::EnableAutomaticEdgeCropping.

E3DComparer::Load

Loads the E3DComparer. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DComparer::SetMeshReference

Sets the 3D reference model. See also E3DComparer::PointCloudReference.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetMeshReference(const EMesh& mesh)
```

E3DComparer::GetNoExtraMaterial

E3DComparer::SetNoExtraMaterial

Sets/Gets a vector of E3DBox that defines the regions where there should not be points in the scan far from the reference. By default, the points far from the reference are not considered as anomalies (if there also are some other points closer to the reference).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DBox> GetNoExtraMaterial() const

void SetNoExtraMaterial(const std::vector<Euresys::Open_eVision::Easy3D::E3DBox>&
noExtraMaterial)
```

E3DComparer::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DComparer& operator=(
 const E3DComparer& other
 )
```

Parameters

other

The E3DComparer object that should be copied.

E3DComparer::SetPointCloudReference

Sets the 3D reference model. See also E3DComparer::MeshReference.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPointCloudReference(const EPointCloud& pointcloud)
```

E3DComparer::PrepareReference

Prepare the reference internal structures. By default, this is done on the first call to E3DComparer::Compare.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void PrepareReference(
 )
```

E3DComparer::GetROI

E3DComparer::SetROI

Sets/Gets a vector of E3DBox that defines the regions that should be compared. By default, the entire reference is considered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DBox> GetROI() const
void SetROI(const std::vector<Euresys::Open_eVision::Easy3D::E3DBox>& roi)
```

E3DComparer::Save

Saves the E3DComparer. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

E3DComparer::SetAnomalyHysteresis

Sets the hysteresis related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetAnomalyHysteresis(
 float distanceHysteresisFactor,
 float areaHysteresisFactor
 )
```

Parameters

distanceHysteresisFactor

The distance hysteresis factor. Must be greater than or equal to 1. Default: 1

areaHysteresisFactor

The area hysteresis factor. Must be smaller than or equal to 1. Default: 1.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See E3DComparer::SetAnomalyThresholds. In addition, an hysteresis can be specified, such that the cluster will also have to contain a subset of points whose distance is greater than distanceHysteresisFactor * distanceThreshold and area greater than areaHysteresisFactor * areaThreshold. See also E3DComparer::SetAnomalyThresholds.

E3DComparer::SetAnomalyThresholds

Sets the thresholds related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetAnomalyThresholds(
 float distanceThreshold,
 float areaThreshold
 )
```

Parameters

distanceThreshold

The distance threshold.

areaThreshold

The area threshold.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See also E3DComparer::SetAnomalyHysteresis to more advanced anomaly detection.

E3DComparer::SetEdgeCroppingParameters

Sets the parameters driving the edge cropping.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetEdgeCroppingParameters(
 OEV_UINT32 numberNeighbors,
 float curvatureThreshold
 )
```

Parameters

numberNeighbors

The number of neighbors that we will use to compute the local curvature. The bigger this parameter, the thicker the cropping will be. This number must be bigger than 10. Default: 100

curvatureThreshold

The curvature value above which a point is cropped (must be between 0 and 1). Default: 0.15

Remarks

Edge cropping is disabled by default, enable it using E3DComparer::EnableAutomaticEdgeCropping.

4.8. E3DLine Class

Represents a 3D line.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Define	(re)Defines the line parameters: It is possible to use the first and the second points of the line.
E3DLine	Creates an E3DLine object. It is possible to initialize it by specifying its two points.
GetFirstPoint	Gets the first point of the line.
GetSecondPoint	Gets the second point of the line.
Load	Loads a E3DLine. The given ESerializer must have been created for reading.
operator!=	Operator "!=": tests if two E3DLine objects are different.
operator=	Assignment operator
operator==	Operator "==": tests if two E3DLine objects are identical

Save	Saves a E3DLine. The given ESerializer must have been created for writing.

E3DLine::Define

(re)Defines the line parameters:
It is possible to use the first and the second points of the line.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Define(
 const E3DPoint& p1,
 const E3DPoint& p2
 )
```

Parameters

p1

-

p2

-

E3DLine::E3DLine

Creates an E3DLine object.
It is possible to initialize it by specifying its two points.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DLine(
 )

void E3DLine(
 const E3DPoint& p1,
 const E3DPoint& p2
 )

void E3DLine(
 const E3DLine& other
 )
```

Parameters

p1
The first point of the line.
p2
The second point of the line.
other
-

E3DLine::GetFirstPoint

Gets the first point of the line.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetFirstPoint() const
```

E3DLine::Load

Loads a E3DLine. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.
serializer
The ESerializer object that is read from.

E3DLine::operator!=

Operator "!=": tests if two E3DLine objects are different.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator!=(
 const E3DLine& other
 )
```

Parameters

other

-

E3DLine::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DLine& operator=(
 const E3DLine& other
 )
```

Parameters

other

The E3DLine object that should be copied.

E3DLine::operator==

Operator "==": tests if two E3DLine objects are identical

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const E3DLine& other
 )
```

Parameters

other

-

E3DLine::Save

Saves a E3DLine. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

E3DLine::GetSecondPoint

Gets the second point of the line.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetSecondPoint() const
```

4.9. E3DMatch Class

Represents a 3D match returned by E3DMatcher.

Base Class:E3DAlignment

Namespace: Euresys::Open_eVision::Easy3D

Methods

E3DMatch	Constructs an E3DMatch.
GetAnomalies	Gets the list of E3DAnomaly of the E3DMatch. The anomalies represent zones where there are important discrepancies between the sample and the reference.
operator=	Assignment operator.

E3DMatch::GetAnomalies

Gets the list of E3DAnomaly of the E3DMatch. The anomalies represent zones where there are important discrepancies between the sample and the reference.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
std::vector<Euresys::Open_eVision::Easy3D::E3DAnomaly> GetAnomalies() const
```

E3DMatch::E3DMatch

Constructs an E3DMatch.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DMatch(
 )
void E3DMatch(
 const E3DMatch& other
 )
void E3DMatch(
 const E3DAlignment& other
 )
```

Parameters

other

Another E3DMatch object to be copied in the new E3DMatch object.

E3DMatch::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DMatch& operator=(
 const E3DMatch& other
 )
```

Parameters

other

The E3DMatch object that should be copied.

4.10. E3DMatcher Class

Aligns an EPointCloud or EZMap on and compares it with a reference EMesh, EPointCloud or EZMap.

Base Class:E3DAligner

Namespace: Euresys::Open_eVision::Easy3D

Methods

ClearComparisonNoExtraMaterial	Clears the ERegion that should not have extra material in the scan. See also E3DMatcher::AllComparisonNoExtraMaterial and E3DMatcher::SetComparisonNoExtraMaterial.
ClearComparisonROI	Clears the ERegion that should be compared for all of the references. See also E3DMatcher::AllComparisonROI and E3DMatcher::SetComparisonROI.
E3DMatcher	Constructs a 3D matching context.
GetAllComparisonNoExtraMaterial	Sets/Gets the ERegion that should not have extra material in the scan. A pointer to ERegion per reference pose is needed and they should be given in the corresponding order and should be of the same size than their corresponding reference poses projection. See E3DAligner::RetrieveReferencePosesProjections. By default, the points in the scan far from the reference are not considered as anomalies (except if there are missing points in the scan nearby).
GetAllComparisonROI	Sets/Gets the ERegion that should be compared for all of the references. A pointer to ERegion per reference pose is needed and they should be given in the corresponding order and should be of the same size than their corresponding reference poses projections. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.
GetAnomalyHysteresis	Gets the hysteresis related to the anomalies detection.
GetAnomalyThresholds	Gets the thresholds related to the anomalies detection.
GetAutomaticCropFactor	Sets/Gets the automatic cropping factor. The factor multiplies the anomaly distance threshold (E3DMatcher::SetAnomalyThresholds) to obtain the margin for the cropping. If the value is smaller than 0, then no crop is performed before computing the distance. Default: 1
GetComparisonDistanceMode	Sets/Gets the distance mode for the 3D comparison. Default: EComparisonDistanceMode_Euclidean.
GetComparisonNoExtraMaterial	Sets/Gets the ERegion that should not have extra material in the scan. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.
GetComparisonPointCloud	Gets the EPointCloud on which anomalies are detected.
GetComparisonROI	Sets/Gets the ERegion of the reference that should be compared. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.
GetEdgeCroppingParameters	Gets the parameters driving the edge cropping.

GetEnableAutomaticDecimation	If True, the given cloud or ZMap is automatically decimated with a resolution depending on the distance threshold to speed-up processing. If False, all the points of the given cloud or ZMap are used to compute the distance. Default: True
GetEnableAutomaticEdgeCropping	If True, edges in the reference will be cropped automatically. This is useful to avoid false positives that could occur near the edges of the object, in particular when using normals (see E3DMatcher::ComparisonDistanceMode and EComparisonDistanceMode). To modify the parameters of the cropping, see E3DMatcher::SetEdgeCroppingParameters. Default: False
GetEnableMissingPointAsAnomaly	If True, the points that are not present in the scan (but are present in the reference), are considered as anomalies. If False, only points that are present in the scan can be considered as anomalies. Default: True
Load	Loads the E3DMatcher. The given ESerializer must have been created for reading.
Match	Matches the reference patterns against an EZMap or an EPointCloud.
operator=	Assignment operator.
PrepareReference	Prepare the reference internal structures. By default, this is done on the first call to E3DMatcher::Match.
Save	Saves the E3DMatcher. The given ESerializer must have been created for writing.
SetAllComparisonNoExtraMaterial	Sets/Gets the ERegion that should not have extra material in the scan. A pointer to ERegion per reference pose is needed and they should be given in the corresponding order and should be of the same size than their corresponding reference poses projection. See E3DAligner::RetrieveReferencePosesProjections. By default, the points in the scan far from the reference are not considered as anomalies (except if there are missing points in the scan nearby).
SetAllComparisonROI	Sets/Gets the ERegion that should be compared for all of the references. A pointer to ERegion per reference pose is needed and they should be given in the corresponding order and should be of the same size than their corresponding reference poses projections. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.
SetAnomalyHysteresis	Sets the hysteresis related to the anomalies detection.
SetAnomalyThresholds	Sets the thresholds related to the anomalies detection.
SetAutomaticCropFactor	Sets/Gets the automatic cropping factor. The factor multiplies the anomaly distance threshold (E3DMatcher::SetAnomalyThresholds) to obtain the margin for the cropping. If the value is smaller than 0, then no crop is performed before computing the distance. Default: 1
SetComparisonDistanceMode	Sets/Gets the distance mode for the 3D comparison. Default: EComparisonDistanceMode_Euclidean.

SetComparisonNoExtraMaterial	Sets/Gets the ERegion that should not have extra material in the scan. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.
SetComparisonROI	Sets/Gets the ERegion of the reference that should be compared. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.
SetEdgeCroppingParameters	Sets the parameters driving the edge cropping.
SetEnableAutomaticDecimation	If True, the given cloud or ZMap is automatically decimated with a resolution depending on the distance threshold to speed-up processing. If False, all the points of the given cloud or ZMap are used to compute the distance. Default: True
SetEnableAutomaticEdgeCropping	If True, edges in the reference will be cropped automatically. This is useful to avoid false positives that could occur near the edges of the object, in particular when using normals (see E3DMatcher::ComparisonDistanceMode and EComparisonDistanceMode). To modify the parameters of the cropping, see E3DMatcher::SetEdgeCroppingParameters. Default: False
SetEnableMissingPointAsAnomaly	If True, the points that are not present in the scan (but are present in the reference), are considered as anomalies. If False, only points that are present in the scan can be considered as anomalies. Default: True

E3DMatcher::GetAllComparisonNoExtraMaterial

E3DMatcher::SetAllComparisonNoExtraMaterial

Sets/Gets the ERegion that should not have extra material in the scan. A pointer to ERegion per reference pose is needed and they should be given in the corresponding order and should be of the same size than their corresponding reference poses projection. See E3DAligner::RetrieveReferencePosesProjections. By default, the points in the scan far from the reference are not considered as anomalies (except if there are missing points in the scan nearby).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::ERegion> GetAllComparisonNoExtraMaterial() const

void SetAllComparisonNoExtraMaterial(const std::vector<const Euresys::Open_
eVision::ERegion*>& noExtraMaterial)
```

E3DMatcher::GetAllComparisonROI

E3DMatcher::SetAllComparisonROI

Sets/Gets the ERegion that should be compared for all of the references. A pointer to ERegion per reference pose is needed and they should be given in the corresponding order and should be of the same size than their corresponding reference poses projections. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::ERegion> GetAllComparisonROI() const

void SetAllComparisonROI(const std::vector<const Euresys::Open_eVision::ERegion*>&
regionsOfInterest)
```

E3DMatcher::GetAutomaticCropFactor

E3DMatcher::SetAutomaticCropFactor

Sets/Gets the automatic cropping factor. The factor multiplies the anomaly distance threshold (E3DMatcher::SetAnomalyThresholds) to obtain the margin for the cropping. If the value is smaller than 0, then no crop is performed before computing the distance.
Default: 1

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetAutomaticCropFactor() const

void SetAutomaticCropFactor(float factor)
```

Remarks

Note that if the cropping margin is too small, the E3DPoint of the anomalies will not be visible (with the function E3DMatcher::GetComparisonPointCloud and the parameter pointFromReference = False). Moreover, if the E3DMatcher::EnableMissingPointAsAnomaly is set to False, the anomalies will not be detected.

E3DMatcher::ClearComparisonNoExtraMaterial

Clears the ERegion that should not have extra material in the scan. See also E3DMatcher::AllComparisonNoExtraMaterial and E3DMatcher::SetComparisonNoExtraMaterial.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearComparisonNoExtraMaterial(
 )
```

E3DMatcher::ClearComparisonROI

Clears the ERegion that should be compared for all of the references. See also E3DMatcher::AllComparisonROI and E3DMatcher::SetComparisonROI.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearComparisonROI(
 )
```

E3DMatcher::GetComparisonDistanceMode

E3DMatcher::SetComparisonDistanceMode

Sets/Gets the distance mode for the 3D comparison. Default: Euclidean.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::EComparisonDistanceMode GetComparisonDistanceMode()
const

void SetComparisonDistanceMode(Euresys::Open_eVision::Easy3D::EComparisonDistanceMode
mode)
```

E3DMatcher::E3DMatcher

Constructs a 3D matching context.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DMatcher(
 )

void E3DMatcher(
 const E3DMatcher& other
 )
```

Parameters

other

Another E3DMatcher object to be copied in the new E3DMatcher object.

E3DMatcher::GetEnableAutomaticDecimation

E3DMatcher::SetEnableAutomaticDecimation

If True, the given cloud or ZMap is automatically decimated with a resolution depending on the distance threshold to speed-up processing. If False, all the points of the given cloud or ZMap are used to compute the distance.
Default: True

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableAutomaticDecimation() const
void SetEnableAutomaticDecimation(bool state)
```

E3DMatcher::GetEnableAutomaticEdgeCropping

E3DMatcher::SetEnableAutomaticEdgeCropping

If True, edges in the reference will be cropped automatically. This is useful to avoid false positives that could occur near the edges of the object, in particular when using normals (see E3DMatcher::ComparisonDistanceMode and EComparisonDistanceMode).
To modify the parameters of the cropping, see E3DMatcher::SetEdgeCroppingParameters.
Default: False

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableAutomaticEdgeCropping() const
void SetEnableAutomaticEdgeCropping(bool state)
```

E3DMatcher::GetEnableMissingPointAsAnomaly

E3DMatcher::SetEnableMissingPointAsAnomaly

If True, the points that are not present in the scan (but are present in the reference), are considered as anomalies. If False, only points that are present in the scan can be considered as anomalies.
Default: True

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetEnableMissingPointAsAnomaly() const
void SetEnableMissingPointAsAnomaly(bool state)
```

Remarks

Setting the value to False can be interesting when for example there are different shadow effects on the reference and on the scan or if there are illumination issues in the scan.

E3DMatcher::GetAnomalyHysteresis

Gets the hysteresis related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetAnomalyHysteresis(
 float& distanceHysteresisFactor,
 float& areaHysteresisFactor
 )
```

Parameters

distanceHysteresisFactor

The distance hysteresis factor. Must be greater than or equal to 1. Default: 1

areaHysteresisFactor

The area hysteresis factor. Must be smaller than or equal to 1. Default: 1.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See E3DMatcher::SetAnomalyThresholds. In addition, an hysteresis can be specified, such that the cluster will also have to contain a subset of points whose distance is greater than distanceHysteresisFactor * distanceThreshold and area greater than areaHysteresisFactor * areaThreshold. See also E3DMatcher::SetAnomalyThresholds.

E3DMatcher::GetAnomalyThresholds

Gets the thresholds related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetAnomalyThresholds(
 float& distanceThreshold,
 float& areaThreshold
 )
```

Parameters

distanceThreshold

The distance threshold.

areaThreshold

The area threshold.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See also E3DMatcher::SetAnomalyHysteresis to more advanced anomaly detection.

E3DMatcher::GetComparisonNoExtraMaterial

Sets/Gets the ERegion that should not have extra material in the scan. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const ERegion* GetComparisonNoExtraMaterial(
 int offset
 )
```

Parameters

offset

offset of the ERegion to get back.

Remarks

This method is a shortcut of E3DMatcher when you only want to set/get one reference pose.

E3DMatcher::GetComparisonPointCloud

Gets the EPointCloud on which anomalies are detected.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetComparisonPointCloud(
 EPointCloud& comparisonCloud,
 bool storeDistances,
 bool storeColors,
 bool pointFromReference,
 bool fullModel
 )
```

Parameters

comparisonCloud

The EPointCloud in which to copy the data.

storeDistances

If set to True, the distances computed will be stored in the EPointCloud.
Default: True

storeColors

If set to True, the colors related to the distances computed will be stored in the EPointCloud.
Default: True

pointFromReference

If set to True, the points from the reference will be stored in comparisonCloud. Otherwise, it will be the points from the scan.
If E3DMatcher::ComparisonDistanceMode is set to EComparisonDistanceMode_Normals_Advanced, result always contains points from the scan.
Default: False

fullModel

If set to True, the points that are not inside the ROI (see E3DMatcher) will also be in the EPointCloud.
Default: False

E3DMatcher::GetComparisonROI

Sets/Gets the ERegion of the reference that should be compared. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const ERegion* GetComparisonROI(
 int offset
 )
```

Parameters

offset

offset of the ERegion to get back.

Remarks

This method is a shortcut of E3DMatcher when you only want to set/get one reference pose.

E3DMatcher::GetEdgeCroppingParameters

Gets the parameters driving the edge cropping.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetEdgeCroppingParameters(
 OEV_UINT32& numberNeighbors,
 float& curvatureThreshold
 )
```

Parameters

numberNeighbors

The number of neighbors that we will use to compute the local curvature. The bigger this parameter, the thicker the cropping will be. This number must be bigger than 10. Default: 100

curvatureThreshold

The curvature value above which a point is cropped (must be between 0 and 1). Default: 0.15

Remarks

Edge cropping is disabled by default, enable it using E3DMatcher::EnableAutomaticEdgeCropping.

E3DMatcher::Load

Loads the E3DMatcher. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DMatcher::Match

Matches the reference patterns against an EZMap or an EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DMatch Match(
 const EZMap& zmap
 )

E3DMatch Match(
 const EPointCloud& cloud,
 const E3DPlane& projectionPlane
 )

E3DMatch Match(
 const EPointCloud& cloud,
 float azimuth,
 float elevation
 )
```

Parameters

zmap

The EZMap that will be aligned and compared with the reference.

cloud

The EPointCloud that will be aligned and compared with the reference.

projectionPlane

The plane on which the cloud is orthographically projected for alignment.

azimuth

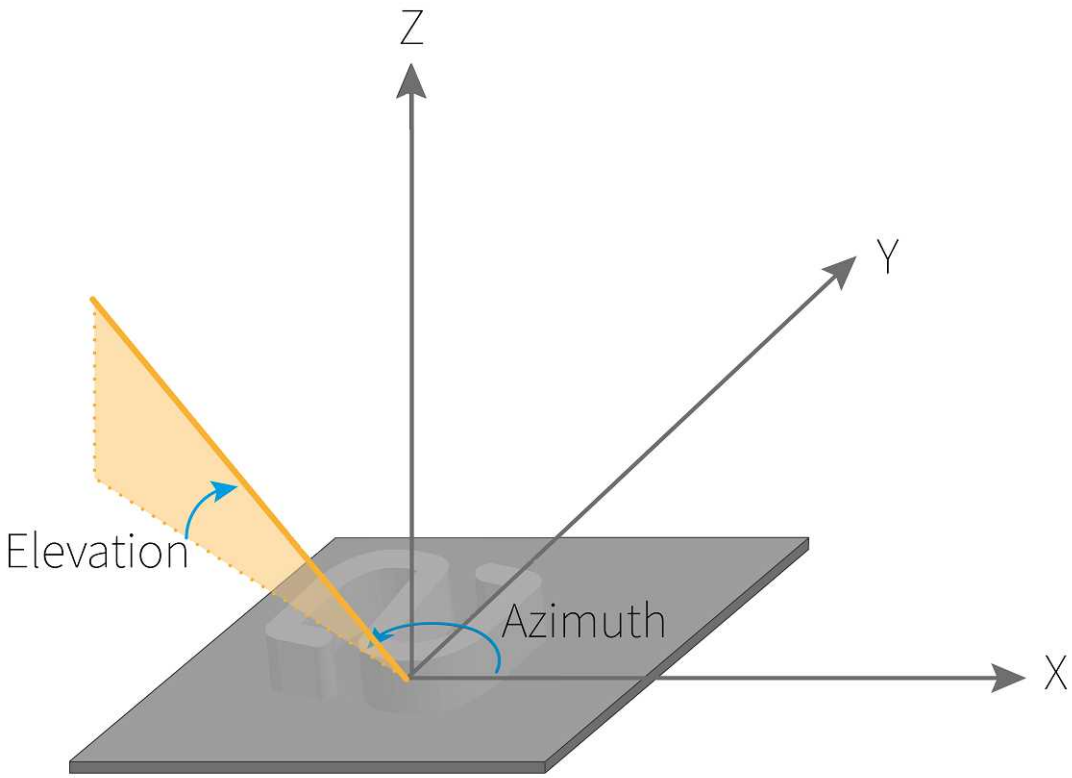
The azimuth angle of the normal of the projection plane in Easy::AngleUnit. Azimuth angles are oriented trigonometrically around the z axis. The x axis corresponds to an azimuth of 0 degrees.

elevation

The elevation angle of the normal of the projection plane in Easy::AngleUnit. Elevation angles represent how close the normal is to the z = 0 plane.

Remarks

When specifying azimuth and elevation, only the normal of the projection plane is specified. The distance from origin of the E3DPlane will be computed from the points cloud, so that all points of the cloud are visible. This might not be wanted if there are points in the cloud that are useless for the process (e.g. if we see other objects far below the model). This is also a bit slower as the plane's distance is recomputed on each scan.



E3DMatcher::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DMatcher& operator=(
 const E3DMatcher& other
 )
```

Parameters

other

The E3DMatcher object that should be copied.

E3DMatcher::PrepareReference

Prepare the reference internal structures. By default, this is done on the first call to E3DMatcher::Match.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void PrepareReference(
 )
```

E3DMatcher::Save

Saves the E3DMatcher. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

E3DMatcher::SetAnomalyHysteresis

Sets the hysteresis related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetAnomalyHysteresis(
 float distanceHysteresisFactor,
 float areaHysteresisFactor
 )
```

Parameters

distanceHysteresisFactor

The distance hysteresis factor. Must be greater than or equal to 1. Default: 1

areaHysteresisFactor

The area hysteresis factor. Must be smaller than or equal to 1. Default: 1.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See E3DMatcher::SetAnomalyThresholds. In addition, an hysteresis can be specified, such that the cluster will also have to contain a subset of points whose distance is greater than distanceHysteresisFactor * distanceThreshold and area greater than areaHysteresisFactor * areaThreshold. See also E3DMatcher::SetAnomalyThresholds.

E3DMatcher::SetAnomalyThresholds

Sets the thresholds related to the anomalies detection.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetAnomalyThresholds(
 float distanceThreshold,
 float areaThreshold
 )
```

Parameters

distanceThreshold

The distance threshold.

areaThreshold

The area threshold.

Remarks

To be considered as an anomaly, a cluster of points with a distance greater than distanceThreshold must have an area greater than areaThreshold. See also E3DMatcher::SetAnomalyHysteresis to more advanced anomaly detection.

E3DMatcher::SetComparisonNoExtraMaterial

Sets/Gets the ERegion that should not have extra material in the scan. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetComparisonNoExtraMaterial(
 const ERegion* noExtraMaterial
 )
```

Parameters

noExtraMaterial

A pointer to ERegion defining the area on which we will check no data is missing in the scan. This param is a shortcut to avoid building a vector of size 1.

Remarks

This method is a shortcut of E3DMatcher when you only want to set/get one reference pose.

E3DMatcher::SetComparisonROI

Sets/Gets the ERegion of the reference that should be compared. See E3DAligner::RetrieveReferencePosesProjections. By default, all the points of the reference are used for the comparison.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetComparisonROI(
 const ERegion* regionOfInterest
 )
```

Parameters

regionOfInterest

A pointer to ERegion defining the area that will be used for comparison with the reference pose. This param is a shortcut to avoid building a vector of size 1.

Remarks

This method is a shortcut of E3DMatcher when you only want to set/get one reference pose.

E3DMatcher::SetEdgeCroppingParameters

Sets the parameters driving the edge cropping.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetEdgeCroppingParameters(
 OEV_UINT32 numberNeighbors,
 float curvatureThreshold
 )
```

Parameters

numberNeighbors

The number of neighbors that we will use to compute the local curvature. The bigger this parameter, the thicker the cropping will be. This number must be bigger than 10. Default: 100

curvatureThreshold

The curvature value above which a point is cropped (must be between 0 and 1). Default: 0.15

Remarks

Edge cropping is disabled by default, enable it using E3DMatcher::EnableAutomaticEdgeCropping.

4.11. E3DObject Class

A E3DObject is a geometric description of a set of 3D points, produced by E3DObjectExtractor. Several 3D features are available. All 3D features are expressed in the ZMap metric coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Draw	Draws the specified feature of the object in the given graphic context
E3DObject	Constructs an E3DObject with default (invalid) values
GetArea	Object area in metric units.
GetAspectRatio	Aspect ratio of the object. The aspect ratio is the smallest dimension divided by the largest dimension, its value is always between 0 and 1. The smaller the ratio, the more elongated the object is. A value of 1 means a perfectly square object (length=width).
GetAveragePosition	3D average position of the object.
GetBasePlane	Base plane. The base plane is calculated using points surrounding the object.
GetBaseTilt	Angle between the base plane and the vertical (Z) axis.
GetBoundingBox	The E3DBox (3D oriented bounding box) of the object. The bounding box is oriented in the XY plane of the ZMap space (rotation over the ZMap Z axis). The bounding box X and Y sizes are the object length and width (see E3DObject and E3DObject). The bounding box Z size is always in the ZMap Z axis direction.
GetIsFeatureComputed	returns true or false depending if the feature was computed for this object
GetLength	Length of the object in metric units. The length is the largest dimension of the object on the XY plane of the ZMap space.

GetLocalHeight	Local height of the object in metric units. The local height of the object is relative to the surroundings. The base plane is used as the reference for the calculation of the local height. If it is not possible to evaluate a base plane, the local height has the same value as the reference height (E3DObject)
GetLocalTilt	Angle between the object plane and the base plane.
GetLocalTopPosition	3D highest position of the object relatively to the object base plane. If it is not possible to evaluate a base plane, the local top position is the reference top position (E3DObject)
GetNumPixels	Number of ZMap pixels composing the object.
GetOrientation	Orientation of the object. The orientation is the angle between the object major (longest) axis and the ZMap X axis.
GetPlane	Plane fitted to the object 3D positions.
GetRectangleRegion	ERectangleRegion enclosing the object ZMap pixels.
GetReferenceHeight	Reference height of the object in metric units. The reference height of the object is relative to the ZMap origin (also known as the reference plane).
GetReferenceTilt	Angle between the object plane and the vertical (Z) axis.
GetReferenceTopPosition	3D top position relative to the ZMap origin (this is the position with the highest Z coordinate)
GetRegion	ERegion composed of the object ZMap pixels.
GetSphere	Sphere fitted to the object
GetVolume	Object volume in metric units.
GetWidth	Width of the object in metric units. The width is the smallest dimension of the object on the XY plane of the ZMap space.
Load	Loads the E3DObject. The given ESerializer must have been created for reading.
operator=	Assignment operator
operator==	Comparison operator
Save	Saves the E3DObject. The given ESerializer must have been created for writing.
Transform	Transforms the 3D object with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only). Image based attributes of the 3D object are unchanged.

E3DObject::GetArea

Object area in metric units.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetArea() const
```

E3DObject::GetAspectRatio

Aspect ratio of the object. The aspect ratio is the smallest dimension divided by the largest dimension, its value is always between 0 and 1. The smaller the ratio, the more elongated the object is. A value of 1 means a perfectly square object (length=width).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetAspectRatio() const
```

E3DObject::GetAveragePosition

3D average position of the object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetAveragePosition() const
```

E3DObject::GetBasePlane

Base plane. The base plane is calculated using points surrounding the object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane GetBasePlane() const
```

E3DObject::GetBaseTilt

Angle between the base plane and the vertical (Z) axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetBaseTilt() const
```

E3DObject::GetBoundingBox

The E3DBox (3D oriented bounding box) of the object. The bounding box is oriented in the XY plane of the ZMap space (rotation over the ZMap Z axis). The bounding box X and Y sizes are the object length and width (see E3DObject and E3DObject). The bounding box Z size is always in the ZMap Z axis direction.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DBox GetBoundingBox() const
```

E3DObject::Draw

Draws the specified feature of the object in the given graphic context

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Draw(
 EDrawAdapter* drawAdapter,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

feature

The feature to draw.

color

The color in which to draw the feature (optional).

zoomX

-

zoomY

-

panX

-

panY

-

graphicContext

Graphic context on which to draw.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

E3DObject::E3DObject

Constructs an E3DObject with default (invalid) values

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DObject(
 )
void E3DObject(
 const E3DObject& other
 )
```

Parameters

other

-

E3DObject::GetIsFeatureComputed

returns true or false depending if the feature was computed for this object

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetIsFeatureComputed(
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

feature

-

E3DObject::GetLength

Length of the object in metric units. The length is the largest dimension of the object on the XY plane of the ZMap space.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetLength() const
```

E3DObject::Load

Loads the E3DObject. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DObject::GetLocalHeight

Local height of the object in metric units. The local height of the object is relative to the surroundings. The base plane is used as the reference for the calculation of the local height. If it is not possible to evaluate a base plane, the local height has the same value as the reference height (E3DObject)

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetLocalHeight() const
```

E3DObject::GetLocalTilt

Angle between the object plane and the base plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetLocalTilt() const
```

E3DObject::GetLocalTopPosition

3D highest position of the object relatively to the object base plane. If it is not possible to evaluate a base plane, the local top position is the reference top position (E3DObject)

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetLocalTopPosition() const
```

E3DObject::GetNumPixels

Number of ZMap pixels composing the object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumPixels() const
```

E3DObject::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DObject& operator=(
 const E3DObject& other
 )
```

Parameters

other

-

E3DObject::operator==

Comparison operator

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const E3DObject& other
 )
```

Parameters

other

The other E3DObject.

E3DObject::GetOrientation

Orientation of the object. The orientation is the angle between the object major (longest) axis and the ZMap X axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetOrientation() const
```

E3DObject::GetPlane

Plane fitted to the object 3D positions.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane GetPlane() const
```

E3DObject::GetRectangleRegion

ERectangleRegion enclosing the object ZMap pixels.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const ERectangleRegion& GetRectangleRegion() const
```

E3DObject::GetReferenceHeight

Reference height of the object in metric units. The reference height of the object is relative to the ZMap origin (also known as the reference plane).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetReferenceHeight() const
```

E3DObject::GetReferenceTilt

Angle between the object plane and the vertical (Z) axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetReferenceTilt() const
```

E3DObject::GetReferenceTopPosition

3D top position relative to the ZMap origin (this is the position with the highest Z coordinate)

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetReferenceTopPosition() const
```

E3DObject::GetRegion

ERegion composed of the object ZMap pixels.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const ERegion& GetRegion() const
```

E3DObject::Save

Saves the E3DObject. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

E3DObject::GetSphere

Sphere fitted to the object

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DSphere& GetSphere() const
```

E3DObject::Transform

Transforms the 3D object with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only). Image based attributes of the 3D object are unchanged.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DObject Transform(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix

The 3D transformation matrix.

E3DObject::GetVolume

Object volume in metric units.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetVolume() const
```

E3DObject::GetWidth

Width of the object in metric units. The width is the smallest dimension of the object on the XY plane of the ZMap space.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetWidth() const
```

4.12. E3DObjectExtractor Class

E3DObjectExtractor is used to extract 3D objects from a ZMap.

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddToMesh	For all extracted objects, adds a 3D representation (in the form of a list of triangles) of the given feature to the given mesh. If the feature is not defined for the E3DObject, this method has no effect.
Draw	Draws the specified feature of all extracted objects in the given graphic context. If the feature is not defined for the E3DObject, this method has no effect.
E3DObjectExtractor	Constructs an E3DObjectExtractor with default (invalid) values
Extract	Processes the ZMap and extracts a list of 3D objects matching the criteria. Returns the number of extracted objects.
GetAreaRange	The allowed area range for the objects. Area is expressed in the metric coordinate system. E.g, if the ZMap space is in mm, the area is in mm2.

GetAspectRatioRange	The extraction 2D aspect ratio range. The aspect ratio is the smallest dimension divided by the largest dimension, its value is always between 0 and 1. The smaller the ratio, the more elongated the object is. A value of 1 means a perfectly square object (length=width).
GetBackgroundMask	Gets the ERegion composed of background ZMap pixels (where there is not object).
GetComputeFeature	Returns true or false depending if the feature should be computed by the extractor.
GetContourReinforce	When contour reinforcement is enable, a filter is applied to the input image to detect and enhance the frontiers between objects. That option is interesting when objects to extract are in contact. The default state is false (OFF).
GetExtractionSensitivity	Set or Get the extraction sensitivity. The sensitivity ranges from 0 (minimum sensitivity) to 1 (maximum sensitivity). With high sensitivity, the library tries to extract object that are mixed with their surrounding. Low sensitivity values tend to ignore faint objects. The default sensitivity value is 0.6.
GetLengthRange	The extraction object length range in metric units. Length is the largest dimension of the object, expressed the ZMap coordinate system.
GetLocalHeightRange	The extraction object local height range in metric units. Local height is the dimension of the object along the normal of the base plane. For a height based on the ZMap origin, use E3DObjectExtractor.
GetLocalTiltRange	The allowed angle range of the object local tilt. This is the angle between the base plane and the object plane. A value of 0 means that the object top surface is parallel to its base. Valid values are between 0 and +90 degrees (or 0 and Pi if angle unit is radians).
GetObjects	Returns the list of extracted objects. The objects are sorted from smallest area to largest area.
GetObjectsMask	Gets the ERegion composed of all the objects ZMap pixels.
GetOrientationRange	The allowed angle range of the oriented 2D rectangle region. This is the angle of the longest axis (the length of the object), in counter clockwise direction, from the horizontal axis. Valid values are between -90 and +90 degrees (or -Pi/2 and Pi/2 if angle unit is radians).
GetOverlappedAreaRatio	Set or Get the object area ratio applicable when the overlapped mode is enabled. The area ratio defines the minimum area difference between 2 extracted objects that overlap. E.g with a value of 4, the top object must have an area at least 4 times smaller than the bottom object (or the bottom object must have an area at least 4 times bigger than the top object).
GetOverlappedHeightDifference	Set or Get the object height difference applicable when the overlapped mode is enabled. That value defines the minimum height difference between 2 overlapped objects.
GetOverlappedObject	Enable or disable the extraction of overlapped objects.
GetReferenceHeightRange	The extraction object reference height range in metric units. Reference height is the dimension of the object along the ZMap Z axis from ZMap origin. For a height based on the object base plane, use E3DObjectExtractor.

GetReferenceTiltRange	The allowed angle range of the object reference tilt. This is the angle between the object plane and the ZMap Z Axis. A value of 0 means that the object top surface is parallel to the ZMap XY plane. Valid values are between 0 and +90 degrees (or 0 and Pi if angle unit is radians).
GetVolumeRange	The allowed volume range for the objects. Volume is expressed in the metric coordinate system. E.g, if the ZMap space is in mm, the volume is in mm3.
GetWidthRange	The extraction object width range in metric units. Width is the smallest dimension of the object, expressed the ZMap coordinate system.
Load	Load the E3DObjectExtractor. The given ESerializer must have been created for reading.
operator=	Assignment operator
operator==	Comparison operator
Save	Save the E3DObjectExtractor. The given ESerializer must have been created for writing.
SetAllComputeFeatures	Enables all features to be computed by the extractor.
SetAreaRange	The allowed area range for the objects. Area is expressed in the metric coordinate system. E.g, if the ZMap space is in mm, the area is in mm2.
SetAspectRatioRange	The extraction 2D aspect ratio range. The aspect ratio is the smallest dimension divided by the largest dimension, its value is always between 0 and 1. The smaller the ratio, the more elongated the object is. A value of 1 means a perfectly square object (length=width).
SetComputeFeature	Sets true if the feature should computed by the extractor, false otherwise.
SetContourReinforce	When contour reinforcement is enable, a filter is applied to the input image to detect and enhance the frontiers between objects. That option is interesting when objects to extract are in contact. The default state is false (OFF).
SetExtractionSensitivity	Set or Get the extraction sensitivity. The sensitivity ranges from 0 (minimum sensitivity) to 1 (maximum sensitivity). With high sensitivity, the library tries to extract object that are mixed with their surrounding. Low sensitivity values tend to ignore faint objects. The default sensitivity value is 0.6.
SetLengthRange	The extraction object length range in metric units. Length is the largest dimension of the object, expressed the ZMap coordinate system.
SetLocalHeightRange	The extraction object local height range in metric units. Local height is the dimension of the object along the normal of the base plane. For a height based on the ZMap origin, use E3DObjectExtractor.
SetLocalTiltRange	The allowed angle range of the object local tilt. This is the angle between the base plane and the object plane. A value of 0 means that the object top surface is parallel to its base. Valid values are between 0 and +90 degrees (or 0 and Pi if angle unit is radians).
SetOrientationRange	The allowed angle range of the oriented 2D rectangle region. This is the angle of the longest axis (the length of the object), in counter clockwise direction, from the horizontal axis. Valid values are between -90 and +90 degrees (or -Pi/2 and Pi/2 if angle unit is radians).

SetOverlappedAreaRatio	Set or Get the object area ratio applicable when the overlapped mode is enabled. The area ratio defines the minimum area difference between 2 extracted objects that overlap. E.g with a value of 4, the top object must have an area at least 4 times smaller than the bottom object (or the bottom object must have an area at least 4 times bigger than the top object).
SetOverlappedHeightDifference	Set or Get the object height difference applicable when the overlapped mode is enabled. That value defines the minimum height difference between 2 overlapped objects.
SetOverlappedObject	Enable or disable the extraction of overlapped objects.
SetReferenceHeightRange	The extraction object reference height range in metric units. Reference height is the dimension of the object along the ZMap Z axis from ZMap origin. For a height based on the object base plane, use E3DObjectExtractor.
SetReferenceTiltRange	The allowed angle range of the object reference tilt. This is the angle between the object plane and the ZMap Z Axis. A value of 0 means that the object top surface is parallel to the ZMap XY plane. Valid values are between 0 and +90 degrees (or 0 and Pi if angle unit is radians).
SetVolumeRange	The allowed volume range for the objects. Volume is expressed in the metric coordinate system. E.g, if the ZMap space is in mm, the volume is in mm3.
SetWidthRange	The extraction object width range in metric units. Width is the smallest dimension of the object, expressed the ZMap coordinate system.
UnsetAllComputeFeatures	Disables all features to be computed by the extractor.

E3DObjectExtractor::AddToMesh

For all extracted objects, adds a 3D representation (in the form of a list of triangles) of the given feature to the given mesh. If the feature is not defined for the E3DObject, this method has no effect.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void AddToMesh(
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature,
 EMesh& mesh
 )
```

Parameters

feature

The feature to draw, only 3D features are supported.

mesh

The mesh to add the graphics for.

E3DObjectExtractor::GetAreaRange

E3DObjectExtractor::SetAreaRange

The allowed area range for the objects. Area is expressed in the metric coordinate system. E.g, if the ZMap space is in mm, the area is in mm2.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetAreaRange() const
void SetAreaRange(const EFloatRange& areaRange)
```

E3DObjectExtractor::GetAspectRatioRange

E3DObjectExtractor::SetAspectRatioRange

The extraction 2D aspect ratio range. The aspect ratio is the smallest dimension divided by the largest dimension, its value is always between 0 and 1. The smaller the ratio, the more elongated the object is. A value of 1 means a perfectly square object (length=width).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetAspectRatioRange() const
void SetAspectRatioRange(const EFloatRange& arRange)
```

E3DObjectExtractor::GetBackgroundMask

Gets the ERegion composed of background ZMap pixels (where there is not object).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ERegion GetBackgroundMask() const
```

E3DObjectExtractor::GetContourReinforce

E3DObjectExtractor::SetContourReinforce

When contour reinforcement is enable, a filter is applied to the input image to detect and enhance the frontiers between objects. That option is interesting when objects to extract are in contact. The default state is false (OFF).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetContourReinforce() const

void SetContourReinforce(bool state)
```

E3DObjectExtractor::Draw

Draws the specified feature of all extracted objects in the given graphic context. If the feature is not defined for the E3DObject, this method has no effect.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext
Graphic context on which to draw.

feature
The feature to draw.

color
The color in which to draw the feature (optional).

zoomX
Horizontal zooming factor. By default, true scale is used.

zoomY
Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX
Horizontal panning value expressed in pixels. By default, no panning occurs.

panY
Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

E3DObjectExtractor::E3DObjectExtractor

Constructs an E3DObjectExtractor with default (invalid) values

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DObjectExtractor(
 )
void E3DObjectExtractor(
 const E3DObjectExtractor& other
 )
```

Parameters

other
-

E3DObjectExtractor::Extract

Processes the ZMap and extracts a list of 3D objects matching the criteria. Returns the number of extracted objects.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int Extract(
 const EZMap8& zMap
 )

int Extract(
 const EZMap8& zMap,
 ERegion& region
 )

int Extract(
 const EZMap16& zMap
 )

int Extract(
 const EZMap16& zMap,
 ERegion& region
 )

int Extract(
 const EZMap32f& zMap
 )

int Extract(
 const EZMap32f& zMap,
 ERegion& region
 )

int Extract(
 const EZMap* zMap
 )

int Extract(
 const EZMap* zMap,
 ERegion& region
 )
```

Parameters

zMap

The source ZMap.

region

The region of interest, only pixels inside the given region are considered for object extraction.

E3DObjectExtractor::GetExtractionSensitivity

E3DObjectExtractor::SetExtractionSensitivity

Set or Get the extraction sensitivity. The sensitivity ranges from 0 (minimum sensitivity) to 1 (maximum sensitivity). With high sensitivity, the library tries to extract object that are mixed with their surrounding. Low sensitivity values tend to ignore faint objects. The default sensitivity value is 0.6.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetExtractionSensitivity() const
void SetExtractionSensitivity(float sensitivity)
```

E3DObjectExtractor::GetComputeFeature

Returns true or false depending if the feature should be computed by the extractor.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetComputeFeature(
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

feature

-

E3DObjectExtractor::GetLengthRange

E3DObjectExtractor::SetLengthRange

The extraction object length range in metric units. Length is the largest dimension of the object, expressed the ZMap coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EFloatRange& GetLengthRange() const
void SetLengthRange(const EFloatRange& lengthRange)
```

E3DObjectExtractor::Load

Load the E3DObjectExtractor. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

E3DObjectExtractor::GetLocalHeightRange

E3DObjectExtractor::SetLocalHeightRange

The extraction object local height range in metric units. Local height is the dimension of the object along the normal of the base plane. For a height based on the ZMap origin, use E3DObjectExtractor.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetLocalHeightRange() const
void SetLocalHeightRange(const EFloatRange& localHeightRange)
```

E3DObjectExtractor::GetLocalTiltRange

E3DObjectExtractor::SetLocalTiltRange

The allowed angle range of the object local tilt. This is the angle between the base plane and the object plane. A value of 0 means that the object top surface is parallel to its base. Valid values are between 0 and +90 degrees (or 0 and Pi if angle unit is radians).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetLocalTiltRange() const
void SetLocalTiltRange(const EFloatRange& tiltRange)
```

E3DObjectExtractor::GetObjects

Returns the list of extracted objects. The objects are sorted from smallest area to largest area.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DObject> GetObjects() const
```

E3DObjectExtractor::GetObjectsMask

Gets the ERegion composed of all the objects ZMap pixels.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ERegion GetObjectsMask() const
```

E3DObjectExtractor::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DObjectExtractor& operator=(
 const E3DObjectExtractor& other
 )
```

Parameters

other

-

E3DObjectExtractor::operator==

Comparison operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator==(
 const E3DObjectExtractor& other
 )
```

Parameters

other

The other object.

E3DObjectExtractor::GetOrientationRange

E3DObjectExtractor::SetOrientationRange

The allowed angle range of the oriented 2D rectangle region. This is the angle of the longest axis (the length of the object), in counter clockwise direction, from the horizontal axis. Valid values are between -90 and +90 degrees (or -Pi/2 and Pi/2 if angle unit is radians).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetOrientationRange() const
void SetOrientationRange(const EFloatRange& orientationRange)
```

E3DObjectExtractor::GetOverlappedAreaRatio

E3DObjectExtractor::SetOverlappedAreaRatio

Set or Get the object area ratio applicable when the overlapped mode is enabled. The area ratio defines the minimum area difference between 2 extracted objects that overlap. E.g with a value of 4, the top object must have an area at least 4 times smaller than the bottom object (or the bottom object must have an area at least 4 times bigger than the top object).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetOverlappedAreaRatio() const
void SetOverlappedAreaRatio(float ratio)
```

Remarks

See E3DObjectExtractor::OverlappedObject

E3DObjectExtractor::GetOverlappedHeightDifference

E3DObjectExtractor::SetOverlappedHeightDifference

Set or Get the object height difference applicable when the overlapped mode is enabled. That value defines the minimum height difference between 2 overlapped objects.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetOverlappedHeightDifference() const
void SetOverlappedHeightDifference(float difference)
```

Remarks

See E3DObjectExtractor::OverlappedObject

E3DObjectExtractor::GetOverlappedObject

E3DObjectExtractor::SetOverlappedObject

Enable or disable the extraction of overlapped objects.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetOverlappedObject() const
void SetOverlappedObject(bool state)
```

Remarks

See E3DObjectExtractor::OverlappedAreaRatio and E3DObjectExtractor::OverlappedHeightDifference

E3DObjectExtractor::GetReferenceHeightRange

E3DObjectExtractor::SetReferenceHeightRange

The extraction object reference height range in metric units. Reference height is the dimension of the object along the ZMap Z axis from ZMap origin. For a height based on the object base plane, use E3DObjectExtractor.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetReferenceHeightRange() const
void SetReferenceHeightRange(const EFloatRange& referenceHeightRange)
```

E3DObjectExtractor::GetReferenceTiltRange

E3DObjectExtractor::SetReferenceTiltRange

The allowed angle range of the object reference tilt. This is the angle between the object plane and the ZMap Z Axis. A value of 0 means that the object top surface is parallel to the ZMap XY plane. Valid values are between 0 and +90 degrees (or 0 and Pi if angle unit is radians).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetReferenceTiltRange() const
void SetReferenceTiltRange(const EFloatRange& tiltRange)
```

E3DObjectExtractor::Save

Save the E3DObjectExtractor. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

E3DObjectExtractor::SetAllComputeFeatures

Enables all features to be computed by the extractor.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetAllComputeFeatures(
 )
```

E3DObjectExtractor::SetComputeFeature

Sets true if the feature should computed by the extractor, false otherwise.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetComputeFeature(
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature,
 bool enable
 )
```

Parameters

feature

A feature from E3DObjectFeature.

enable

A bool to choose whether to enable or disable a feature.

E3DObjectExtractor::UnsetAllComputeFeatures

Disables all features to be computed by the extractor.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetAllComputeFeatures(
 )
```

E3DObjectExtractor::GetVolumeRange

E3DObjectExtractor::SetVolumeRange

The allowed volume range for the objects. Volume is expressed in the metric coordinate system. E.g, if the ZMap space is in mm, the volume is in mm3.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EFloatRange& GetVolumeRange() const

void SetVolumeRange(const EFloatRange& volumeRange)
```

E3DObjectExtractor::GetWidthRange

E3DObjectExtractor::SetWidthRange

The extraction object width range in metric units. Width is the smallest dimension of the object, expressed the ZMap coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EFloatRange& GetWidthRange() const

void SetWidthRange(const EFloatRange& widthRange)
```

4.13. E3DOrthonormalAxisSystem Class

E3DOrthonormalAxisSystem is a subassembly of E3DAxisSystem with properties Orthogonal and Normed

Base Class:E3DAxisSystem

Namespace: Euresys::Open_eVision::Easy3D

Methods

E3DOrthonormalAxisSystem	Constructs an E3DOrthonormalAxisSystem.
operator=	Assignment operator.

E3DOrthonormalAxisSystem::E3DOrthonormalAxisSystem

Constructs an E3DOrthonormalAxisSystem.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DOrthonormalAxisSystem(
 )

void E3DOrthonormalAxisSystem(
 const E3DAxisSystem& other
 )

void E3DOrthonormalAxisSystem(
 const E3DOrthonormalAxisSystem& other
 )
```

```
void E3DOrthonormalAxisSystem(
 const E3DPoint& Origin,
 const E3DPoint& axisX,
 const E3DPoint& axisY,
 const E3DPoint& axisZ
 )
```

Parameters

other
Reference to another E3DOrthonormalAxisSystem used for the initialization.
Origin
The origin of the axis system
axisX
The X axis
axisY
The Y axis
axisZ
The Z axis

Remarks

throws an exception if the axis are not orthogonal and normed

E3DOrthonormalAxisSystem::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DOrthonormalAxisSystem& operator=(
 const E3DOrthonormalAxisSystem& other
 )
```

Parameters

other
The source E3DOrthonormalAxisSystem.

4.14. E3DPlane Class

Represents a 3D plane.
The equation of the plane is "n_vect . (x,y,z) = signedDistance" where "n_vect" is the normal vector and "signedDistance" is the signed distance from the origin to the plane.
The signed distance is positive when the vector binding the origin to the closest point on the plane has the same direction as "n_vect" and is negative when this vector has the opposite direction as "n_vect".

Namespace: Euresys::Open_eVision::Easy3D

Methods

AngleWithPlane	Returns the angle (in the first quadrant) between the normals of this plane and the one passed in argument.
Define	(re)Defines the plane parameters: It is possible to use the normal and the signed distance of the plane. In that case, it exits with an exception if the normal vector is the null vector. It is also possible to use 3 points of the plane. In that case, it exits with an exception if the 3 points are aligned.
DistanceTo	Returns the signed distance between the plane and a given point. A positive distance means that the vector connecting the plane to the point has the same direction as the normal while a negative distance means that it has the opposite direction.
E3DPlane	Creates an E3DPlane object. It is possible to initialize it by specifying its normal and signed distance from the origin. In that case, it exits with an exception if the norm of the normal is null. It is also possible to initialize it by specifying 3 points of the plane. In that case, it exits with an exception if the 3 points are aligned.
GetNormal	Gets/Sets the normal vector of the plane.
GetSignedDistanceFromOrigin	Gets/Sets the signed distance between the origin and the plane.
GetTransformationTo	Return the transformation that moves the plane to the given destination plane.
IntersectionWithTwoPlanes	Compute the intersection between this plane and the two given as argument. If the intersection exists, true is returned and intersection is filled with the intersection.
Load	Loads a E3DPlane. The given ESerializer must have been created for reading.
operator-	Operator "-": returns a new E3DPlane translated in the inverse of the direction of the normal to the plane.
operator+	Operator "+": returns a new E3DPlane translated in the direction of the normal to the plane.
operator=	Assignment operator
ProjectPoint	Returns the position of the given point projected on the plane.
Save	Saves a E3DPlane. The given ESerializer must have been created for writing.
SetNormal	Gets/Sets the normal vector of the plane.
SetSignedDistanceFromOrigin	Gets/Sets the signed distance between the origin and the plane.

Transform	Transforms the 3D plane with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only).
XPlane	The X=0 plane.
YPlane	The Y=0 plane.
ZPlane	The Z=0 plane.

E3DPlane::AngleWithPlane

Returns the angle (in the first quadrant) between the normals of this plane and the one passed in argument.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float AngleWithPlane(
 const E3DPlane& other
 )
```

Parameters

other

The plane with respect to which we compute our angle

Remarks

This function does not take the orientation of the normals into account, i.e. (0, 0, 1) and (0, 0, -1) will give the same angle

E3DPlane::Define

(re)Defines the plane parameters:
It is possible to use the normal and the signed distance of the plane.
In that case, it exits with an exception if the normal vector is the null vector.
It is also possible to use 3 points of the plane.
In that case, it exits with an exception if the 3 points are aligned.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Define(
 const E3DPoint& normal,
 float signedDistance
 )

void Define(
 const E3DPoint& point1,
 const E3DPoint& point2,
 const E3DPoint& point3
 )
```

Parameters

normal

The normal vector, represented by an E3DPoint.

signedDistance

The signed distance between the origin and the plane.

point1

First point.

point2

Second point.

point3

Third point.

Remarks

When we define a plane by specifying 3 points, the normal vector always points toward the positive Z.

E3DPlane::DistanceTo

Returns the signed distance between the plane and a given point.
A positive distance means that the vector connecting the plane to the point has the same direction as the normal while a negative distance means that it has the opposite direction.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float DistanceTo(
 const E3DPoint& point
 )
```

Parameters

point

The 3D point to measure the distance to.

E3DPlane::E3DPlane

Creates an E3DPlane object.
It is possible to initialize it by specifying its normal and signed distance from the origin.
In that case, it exits with an exception if the norm of the normal is null.
It is also possible to initialize it by specifying 3 points of the plane.
In that case, it exits with an exception if the 3 points are aligned.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DPlane(
 )
```

```
void E3DPlane(
 const E3DPoint& normal,
 float signedDistance
 )

void E3DPlane(
 const E3DPoint& point1,
 const E3DPoint& point2,
 const E3DPoint& point3
 )

void E3DPlane(
 const E3DPlane& other
 )
```

Parameters

normal

The normal vector. May not be null.

signedDistance

The signed distance from the origin to the plane.

point1

First point

point2

Second point

point3

Third point

other

Reference to the plane used for the initialization.

Remarks

When we define a plane by specifying 3 points, the normal vector always points toward the positive Z.

E3DPlane::GetTransformationTo

Return the transformation that moves the plane to the given destination plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetTransformationTo(
 const E3DPlane& destination
 )
```

Parameters

destination

The destination plane.

E3DPlane::IntersectionWithTwoPlanes

Compute the intersection between this plane and the two given as argument. If the intersection exists, true is returned and intersection is filled with the intersection.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IntersectionWithTwoPlanes(
 const E3DPlane& plane1,
 const E3DPlane& plane2,
 E3DPoint& intersection
 )
```

Parameters

plane1
The first plane with which we compute the intersection.

plane2
The second plane with which we compute the intersection.

intersection
The point that will contain the intersection between the three planes.

E3DPlane::Load

Loads a E3DPlane. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

E3DPlane::GetNormal

E3DPlane::SetNormal

Gets/Sets the normal vector of the plane.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetNormal() const
void SetNormal(const E3DPoint& vector)
```

Remarks

Normal values will be stored normalized.

E3DPlane::operator-

Operator "-": returns a new E3DPlane translated in the inverse of the direction of the normal to the plane.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane operator-(
 float offset
 )
```

Parameters

offset

offset value

E3DPlane::operator+

Operator "+": returns a new E3DPlane translated in the direction of the normal to the plane.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane operator+(
 float offset
 )
```

Parameters

offset

offset value

E3DPlane::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane& operator=(
 const E3DPlane& other
 )
```

Parameters

other

The E3DPlane object that should be copied.

E3DPlane::ProjectPoint

Returns the position of the given point projected on the plane.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint ProjectPoint(
 const E3DPoint& point
 )
```

Parameters

point

The 3D point to project on plane.

E3DPlane::Save

Saves a E3DPlane. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

E3DPlane::GetSignedDistanceFromOrigin

E3DPlane::SetSignedDistanceFromOrigin

Gets/Sets the signed distance between the origin and the plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetSignedDistanceFromOrigin() const
void SetSignedDistanceFromOrigin(float signedDistance)
```

E3DPlane::Transform

Transforms the 3D plane with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane Transform(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix
The 3D transformation matrix.

E3DPlane::XPlane

The X=0 plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane XPlane(
 )
```

E3DPlane::YPlane

The Y=0 plane.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane YPlane(
 )
```

E3DPlane::ZPlane

The Z=0 plane.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane ZPlane(
 )
```

4.15. E3DRightOrthonormalAxisSystem Class

E3DRightOrthonormalAxisSystem is a subassembly of E3DAxisSystem with properties Orthogonal and Normed and Right

Base Class:E3DAxisSystem

Namespace: Euresys::Open_eVision::Easy3D

Methods

E3DRightOrthonormalAxisSystem	Constructs an E3DRightOrthonormalAxisSystem.
operator=	Assignment operator.

E3DRightOrthonormalAxisSystem::E3DRightOrthonormalAxisSystem

Constructs an E3DRightOrthonormalAxisSystem.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void E3DRightOrthonormalAxisSystem(
 )
```

```
void E3DRightOrthonormalAxisSystem(
 const E3DAxisSystem& other
 )

void E3DRightOrthonormalAxisSystem(
 const E3DRightOrthonormalAxisSystem& other
 )

void E3DRightOrthonormalAxisSystem(
 const E3DPoint& Origin,
 const E3DPoint& axisX,
 const E3DPoint& axisY,
 const E3DPoint& axisZ
 )
```

Parameters

other

Reference to another E3DRightOrthonormalAxisSystem used for the initialization.

Origin

The origin of the axis system

axisX

The X axis

axisY

The Y axis

axisZ

The Z axis

Remarks

throws an exception if the axis are not orthogonal and normed and right

E3DRightOrthonormalAxisSystem::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DRightOrthonormalAxisSystem& operator=(
 const E3DRightOrthonormalAxisSystem& other
 )
```

Parameters

other

The source E3DRightOrthonormalAxisSystem.

4.16. E3DSphere Class

Represents a 3D sphere.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Define	(re)Defines the sphere parameters: It is possible to use the center and the radius of the sphere.
DistanceTo	Computes the distance between a E3DPoint and the sphere, defined as the distance to the nearest point of the sphere.
E3DSphere	Creates an E3DSphere object. It is possible to initialize it by specifying its center and radius.
GenerateMesh	-
GetCenter	Gets the center point of the sphere.
GetRadius	Gets the radius of the sphere.
Load	Loads a E3DSphere. The given ESerializer must have been created for reading.
operator=	Assignment operator
Save	Saves a E3DSphere. The given ESerializer must have been created for writing.
Transform	Transforms the 3D sphere with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only).

E3DSphere::GetCenter

Gets the center point of the sphere.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetCenter() const
```

E3DSphere::Define

(re)Defines the sphere parameters:
It is possible to use the center and the radius of the sphere.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Define(
 const E3DPoint& center,
 float radius
 )
```

Parameters

center

The center of the sphere.

radius

The radius of the sphere.

E3DSphere::DistanceTo

Computes the distance between a E3DPoint and the sphere, defined as the distance to the nearest point of the sphere.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float DistanceTo(
 const E3DPoint& point
 )
```

Parameters

point

The 3D point to measure the distance to.

E3DSphere::E3DSphere

Creates an E3DSphere object.
It is possible to initialize it by specifying its center and radius.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DSphere(
 )

void E3DSphere(
 const E3DPoint& center,
 float radius
 )

void E3DSphere(
 const E3DSphere& other
 )
```

Parameters

center
The center of the sphere.
radius
The radius of the sphere.
other
-

E3DSphere::GenerateMesh

-

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EMesh GenerateMesh(
 int subdivision,
 float& distanceToSphere
 )
```

Parameters

subdivision
-
distanceToSphere
-

E3DSphere::Load

Loads a E3DSphere. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.
serializer
The serializer.

E3DSphere::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DSphere& operator=(
 const E3DSphere& other
 )
```

Parameters

other

The E3DSphere object that should be copied.

E3DSphere::GetRadius

Gets the radius of the sphere.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetRadius() const
```

E3DSphere::Save

Saves a E3DSphere. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

E3DSphere::Transform

Transforms the 3D sphere with the given transformation matrix. The transformation matrix must contain a rigid transformation (translation and rotation only).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DSphere Transform(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix

The 3D transformation matrix.

4.17. E3DTransformMatrix Class

Represents a 3D transformation [4x4] matrix.

Namespace: Euresys::Open_eVision::Easy3D

Methods

CreateAnisotropicScalingMatrix	Creates an anisotropic scaling E3DTransformMatrix.
CreateIdentityMatrix	Creates an identity (neutral) E3DTransformMatrix.
CreateIsotropicScalingMatrix	Creates an isotropic scaling E3DTransformMatrix.
CreateOrthoBasis	Creates a orthonormal E3DTransformMatrix basis (corresponds to a rigid transformation). The vector e1, e2, e3 should form a right-handed orthogonal basis.
CreateOrthographicProjectionMatrix	Creates an orthographic projection E3DTransformMatrix.
CreatePerspectiveProjectionMatrix	Creates a perspective projection E3DTransformMatrix.
CreateRotationMatrix	Creates a rotation E3DTransformMatrix around the given axis for the given angle.
CreateRotationXMatrix	Creates a rotation E3DTransformMatrix around the X axis matrix.
CreateRotationYMatrix	Creates a rotation E3DTransformMatrix around the Y axis matrix.
CreateRotationZMatrix	Creates a rotation E3DTransformMatrix around the Z axis matrix.
CreateTranslationMatrix	Creates a translation E3DTransformMatrix.

E3DTransformMatrix	Creates an E3DTransformMatrix object.
GetAzimuthElevationAngles	Gets the azimuth and elevation angles for the Z axis of the coordinate system represented by the transformation matrix. Azimuth angle is oriented trigonometrically around the z axis. The x axis corresponds to an azimuth of 0 degrees. Elevation angle represents the height of the normal w.r.t. the z = 0 plane. The E3DTransformMatrix must be rigid (translation and rotation only).
GetEulerAngles	Gets the Euler angles for the rotation represented by this transformation. The returned value is a 3D point with Euler angles around X, Y and Z axis. The E3DTransformMatrix must be rigid (translation and rotation only).
GetOrthoBasis	Gets the orthogonal basis represented by this transformation or throws an exception if it is not a rigid transformation.
GetValue	Gets a value from the E3DTransformMatrix object.
Inverse	Returns the inverted E3DTransformMatrix. An exception will be thrown if the determinant of the matrix is 0.
IsRigid	Checks that the transformation is a rigid transformation (keep the distances and angles).
Load	Loads the E3DTransformMatrix object. The given ESerializer must have been created for reading.
operator-	E3DTransformMatrix difference. Subtract the current and the given matrix, returns the result.
operator!=	Checks if two E3DTransformMatrix objects are strictly different (binary level).
operator*	E3DTransformMatrix product. Combines the transformations of the two matrices.
operator+	E3DTransformMatrix sum. Sums the current and the given matrix, returns the result.
operator=	Assignment operator.
operator==	Checks if two E3DTransformMatrix objects are strictly equals (binary level).
Save	Saves the E3DTransformMatrix object. The given ESerializer must have been created for writing.
SetValue	Sets a value in the E3DTransformMatrix object.
Transpose	Returns the transposed E3DTransformMatrix. If the matrix is orthogonal (rotation only transformation), the transposed matrix is the inverse transformation.

E3DTransformMatrix::CreateAnisotropicScalingMatrix

Creates an anisotropic scaling E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreateAnisotropicScalingMatrix(
 float scaleX,
 float scaleY,
 float scaleZ
 )
```

Parameters

scaleX
Scaling factor along the X axis.

scaleY
Scaling factor along the Y axis.

scaleZ
Scaling factor along the Z axis.

E3DTransformMatrix::CreateIdentityMatrix

Creates an identity (neutral) E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreateIdentityMatrix(
 )
```

E3DTransformMatrix::CreateIsotropicScalingMatrix

Creates an isotropic scaling E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreateIsotropicScalingMatrix(
 float scale
 )
```

Parameters

scale
Scaling factor.

E3DTransformMatrix::CreateOrthoBasis

Creates a orthonormal E3DTransformMatrix basis (corresponds to a rigid transformation).
The vector e1, e2, e3 should form a right-handed orthogonal basis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateOrthoBasis(
 const E3DPoint& e1,
 const E3DPoint& e2,
 const E3DPoint& e3,
 const E3DPoint& t
 )
```

Parameters

e1
Vector 1.

e2
Vector 2.

e3
Vector 3.

t
Translation.

E3DTransformMatrix::CreateOrthographicProjectionMatrix

Creates an orthographic projection E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateOrthographicProjectionMatrix(
 float width,
 float height
 )
```

Parameters

width
Width of the viewport.

height
Height of the viewport.

E3DTransformMatrix::CreatePerspectiveProjectionMatrix

Creates a perspective projection E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreatePerspectiveProjectionMatrix(
 float distance,
 float width,
 float height
 )
```

Parameters

distance
Distance of the viewport to the origin.

width
Width of the viewport.

height
Height of the viewport.

E3DTransformMatrix::CreateRotationMatrix

Creates a rotation E3DTransformMatrix around the given axis for the given angle.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreateRotationMatrix(
 const E3DPoint& axis,
 float angle
 )
```

Parameters

axis
Rotation axis.

angle
Rotation angle.

E3DTransformMatrix::CreateRotationXMatrix

Creates a rotation E3DTransformMatrix around the X axis matrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreateRotationXMatrix(
 float angle
 )
```

Parameters

angle

Rotation angle.

E3DTransformMatrix::CreateRotationYMatrix

Creates a rotation E3DTransformMatrix around the Y axis matrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateRotationYMatrix(
 float angle
 )
```

Parameters

angle

Rotation angle.

E3DTransformMatrix::CreateRotationZMatrix

Creates a rotation E3DTransformMatrix around the Z axis matrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateRotationZMatrix(
 float angle
 )
```

Parameters

angle

Rotation angle.

E3DTransformMatrix::CreateTranslationMatrix

Creates a translation E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateTranslationMatrix(
 float dX,
 float dY,
 float dZ
 )
```

Parameters

dX

Translation along the X axis.

dY

Translation along the Y axis.

dZ

Translation along the Z axis.

E3DTransformMatrix::E3DTransformMatrix

Creates an E3DTransformMatrix object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DTransformMatrix(
 )

void E3DTransformMatrix(
 const E3DTransformMatrix& other
 )

void E3DTransformMatrix(
 double m00,
 double m10,
 double m20,
 double m30,
 double m01,
 double m11,
 double m21,
 double m31,
 double m02,
 double m12,
 double m22,
 double m32,
 double m03,
 double m13,
 double m23,
 double m33
 )
```

Parameters

other
Another E3DTransformMatrix used for the initialization.

m00
Matrix value line 0 column 0.

m10
Matrix value line 0 column 1.

m20
Matrix value line 0 column 2.

m30
Matrix value line 0 column 3.

m01
Matrix value line 1 column 0.

m11
Matrix value line 1 column 1.

m21
Matrix value line 1 column 2.

m31
Matrix value line 1 column 3.

m02
Matrix value line 2 column 0.

m12
Matrix value line 2 column 1.

m22
Matrix value line 2 column 2.

m32
Matrix value line 2 column 3.

m03
Matrix value line 3 column 0.

m13
Matrix value line 3 column 1.

m23
Matrix value line 3 column 2.

m33
Matrix value line 3 column 3.

Remarks

The matrix is initialized with the given values.
By default, the matrix is initialized as an identity (neutral) matrix.
The value indices are m(column, row).

E3DTransformMatrix::GetEulerAngles

Gets the Euler angles for the rotation represented by this transformation. The returned value is a 3D point with Euler angles around X, Y and Z axis. The E3DTransformMatrix must be rigid (translation and rotation only).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetEulerAngles() const
```

E3DTransformMatrix::GetAzimuthElevationAngles

Gets the azimuth and elevation angles for the Z axis of the coordinate system represented by the transformation matrix.
Azimuth angle is oriented trigonometrically around the z axis. The x axis corresponds to an azimuth of 0 degrees.
Elevation angle represents the height of the normal w.r.t. the z = 0 plane.
The E3DTransformMatrix must be rigid (translation and rotation only).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetAzimuthElevationAngles(
 float& azimuth,
 float& elevation
 )
```

Parameters

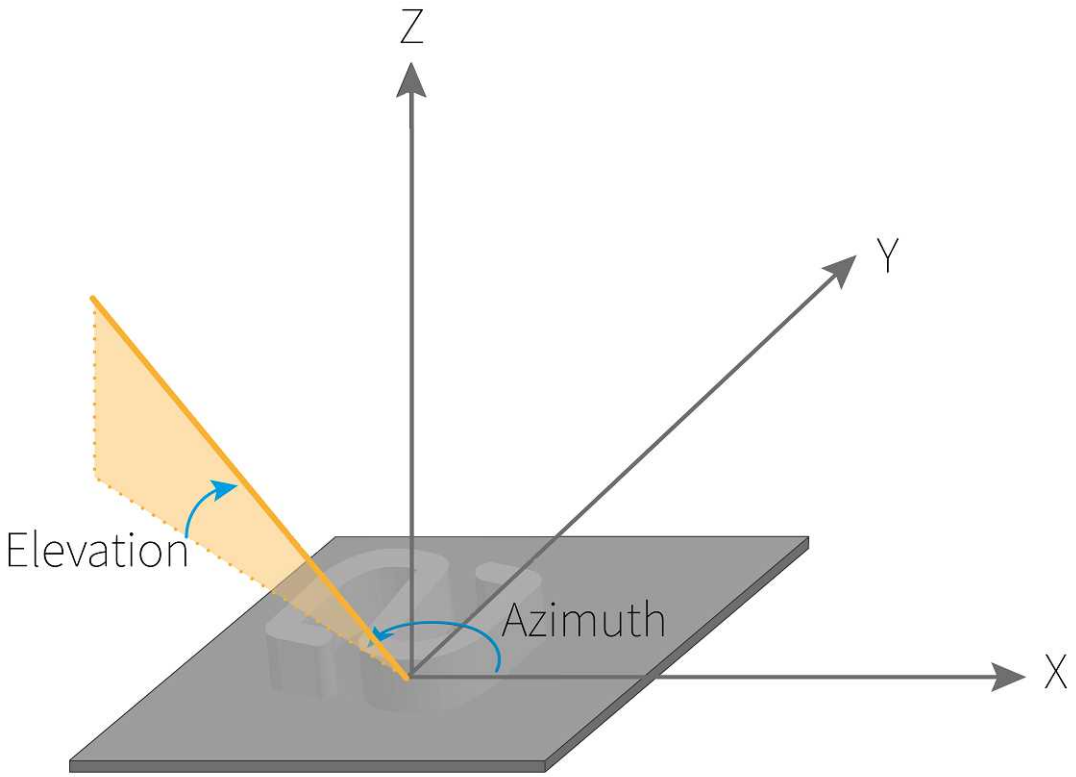
azimuth

The returned azimuth angle.

elevation

The returned elevation angle.

Remarks



E3DTransformMatrix::GetOrthoBasis

Gets the orthogonal basis represented by this transformation or throws an exception if it is not a rigid transformation.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetOrthoBasis(
 E3DPoint& e1,
 E3DPoint& e2,
 E3DPoint& e3,
 E3DPoint& t
 )
```

Parameters

e1
Vector 1.

e2
Vector 2.

e3
Vector 3.

t
Translation.

E3DTransformMatrix::GetValue

Gets a value from the E3DTransformMatrix object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetValue(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
```

Parameters

column
Column of the value to get, from 0 to 3.

row
Row of the value to get, from 0 to 3.

E3DTransformMatrix::Inverse

Returns the inverted E3DTransformMatrix.
An exception will be thrown if the determinant of the matrix is 0.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix Inverse(
 )
```

E3DTransformMatrix::IsRigid

Checks that the transformation is a rigid transformation (keep the distances and angles).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsRigid(
 )
```

E3DTransformMatrix::Load

Loads the E3DTransformMatrix object. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

E3DTransformMatrix::operator-

E3DTransformMatrix difference. Subtract the current and the given matrix, returns the result.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix operator-(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix
Matrix to subtract from the current matrix.

E3DTransformMatrix::operator!=

Checks if two E3DTransformMatrix objects are strictly different (binary level).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator!=(
 const E3DTransformMatrix& other
 )
```

Parameters

other

The other matrix.

E3DTransformMatrix::operator*

E3DTransformMatrix product. Combines the transformations of the two matrices.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix operator*(
 const E3DTransformMatrix& matrix
 )
E3DPoint operator*(
 E3DPoint P
 )
```

Parameters

matrix

Matrix to combine with the current matrix.

P

Point to transform with the current matrix.

E3DTransformMatrix::operator+

E3DTransformMatrix sum. Sums the current and the given matrix, returns the result.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix operator+(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix

Matrix to add with the current matrix.

E3DTransformMatrix::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix& operator=(
 const E3DTransformMatrix& other
 )
```

Parameters

other

An other E3DTransformMatrix.

E3DTransformMatrix::operator==

Checks if two E3DTransformMatrix objects are strictly equals (binary level).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const E3DTransformMatrix& other
 )
```

Parameters

other

The other matrix.

E3DTransformMatrix::Save

Saves the E3DTransformMatrix object. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

E3DTransformMatrix::SetValue

Sets a value in the E3DTransformMatrix object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetValue(
 OEV_UINT32 column,
 OEV_UINT32 row,
 float value
 )
```

Parameters

column
Column of the value to set, from 0 to 3.

row
Row of the value to set, from 0 to 3.

value
Value to set.

E3DTransformMatrix::Transpose

Returns the transposed E3DTransformMatrix.
If the matrix is orthogonal (rotation only transformation), the transposed matrix is the inverse transformation.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix Transpose(
 )
```

4.18. E3DViewer Class

Manages a viewer window for EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddRenderSource	Adds a new Render Source. The given render source (point cloud, mesh, ZMap, sphere, box, plane or line) is added to the current display.
AddTextLabel	Adds a text label linked to an E3DPoint.
ClearRenderSource	Clears the 3D data, nothing will be displayed.
ClearTextLabels	Removes all text labels.
ConfigureRenderSource	Sets a unique 3D source to be rendered. It replaces the current object.
DecPointSize	Decreases the displayed point size.
DisableFixColorRampBounds	Disables the fix color ramp bounds.
E3DViewer	Creates an E3DViewer object.
EditTextLabel	Edits a text label.
GenerateColors	Choose the current color ramp mode. Colors are calculated from point coordinates or attributes, several mappings are exposed in EColorRampMode.
GetAutoRotate	Gets the auto rotate speeds.
GetAxisOrigin	Sets and Gets the axis origin mode.
GetBackgroundColor	Sets/Gets the 3D viewer background color.
GetColorRampGraduationColor	Sets/Gets the graduation color.
GetColorRampLocation	Gets the location of the color ramp.
GetColorRampMode	Set/Gets the current color ramp mode.
GetEDLShadingFactor	Gets/Sets how strong Eye-Dome-Lighting shading is. In this mode, every pixel is shaded by how much closer its neighbor pixels are to the camera.
GetEnableEDLShading	Gets/Sets if Eye-Dome-Lighting shading mode is enabled. In this mode, every pixel is shaded by how much closer its neighbor pixels are to the camera.
GetEnableFixColorRampBounds	Gets the state of the fix color ramp bounds.
GetEnableSmartColorRamp	Sets/Gets the state of the smart generation of color. If the smart generation of color is active, then the outliers inside the pointcloud will not be taken into account in the color ramp.
GetFeatureStyleFor3DObject	Gets how the E3DObjectFeature of the registered E3DObject at the specified location idx should be rendered.
GetFieldOfView	Sets/Gets the field of view.
GetFixColorRampBounds	Gets the bounds of the color ramp.
GetFontPath	Configure the TrueType font used to display text on the 3D Viewer

GetLastPickedPoint	Returns the last picked E3DPoint if there is one. Otherwise, it throws an EException.
GetNumRenderSources	Returns the number of render sources.
GetPickingDisplay	Enables or disables the display of the picked point.
GetPickingDistanceThreshold	Sets or gets the distance threshold for the picking.
GetPickingLabelColor	Sets or gets the color of the picked point label.
GetPickingLabelFixed	Sets or gets the state to fix or not the label.
GetPickingLabelSize	Sets or gets the size of the picked point label.
GetPointSize	Displays size of the points, in pixels.
GetProjectionType	Sets/Gets the projection type.
GetRenderAxis	Enables or disables the display of the axis.
GetRenderAxisConfiguration	Sets/Gets the axis configuration.
GetRenderDecimationLevel	Sets or Gets the point cloud decimation level used during the rendering.
GetRenderGrid	Enables or disables the display of Grid.
GetRenderGridStep	Sets/Gets the same grid step for each axis.
GetRenderSourceColorMode	Gets the Source Color Mode. The Source Color Mode defines how the colors of the point cloud or mesh are chosen. See ESourceColorMode.
GetRenderSourceConstantColor	Gets the constant color used to display a point cloud or a mesh, when ESourceColorMode is set to ESourceColorMode_Constant.
GetRenderSourceName	Returns the name of the ith render source.
GetRenderSourceOpacity	Gets the opacity used to display the render source. Only compatible with ESourceColorMode_Constant.
GetRenderSourcePointSize	Gets the point size used to display the point clouds.
GetRenderSourceWireFrame	Gets the wireframe mode used to render a mesh.
GetRenderSourceWireFrameColor	Gets the wireframe color.
GetRotationMatrix	Gets the view rotation matrix.
GetTextLabel	Gets the text label information.
GetViewAngle	Gets view angles.
GetViewDistance	Sets/Gets view distance.
GetViewTarget	Gets view target position (by default, the camera position is 'look at center of the point cloud').
GetWireframeMode	Enables or disables the display of wireframe triangles.
Has3DObjects	Return true if at least one 3D object is registered

HasRenderSource	Is the given render source name exists ?
HideColorRampLegend	Disables the display of a color ramp legend.
HideFeatureFor3DObject	Sets the E3DObjectFeature of the registered E3DObject at the specified location idx to be not rendered.
HideFeatureForAll3DObjects	Sets the E3DObjectFeature of all the registered E3DObject to be not rendered.
HideRenderSource	Hides the given render source.
IncPointSize	Increases the displayed point size.
InitRendering	Initializes the rendering state, must be called one time before the first E3DViewer::Show.
IsAutoRotate	Returns true if the auto rotation is active.
IsColorRampLegendVisible	Whether the color ramp legend is visible or not.
IsEDLShadingSupported	Indicates whether Eye-Dome-Lighting shading mode is supported.
IsFeatureVisibleFor3DObject	Whether the E3DObjectFeature of the registered E3DObject si rendered or not.
IsRenderSourceVisible	Returns true if the render source is currently visible.
LockRotationFinalPosition	Finalizes a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is released.
LockRotationInitialPosition	Starts a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is pressed.
LockTranslationFinalPosition	Finalizes a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is released.
LockTranslationInitialPosition	Starts a translation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is pressed.
Pick3DPoint	Returns and displays the picked E3DPoint in the EPointCloud.
Register3DObjects	Sets the list of E3DObject from which their features can be rendered.
RemoveAllRenderSources	Removes all render sources. The E3DViewer display will be empty.
RemoveCurrent3DObjects	Removes all E3DObject currently registered.
RemoveRenderSource	Removes the given render source.
RemoveTextLabel	Removes a text label.
ResetPicking	Resets the last picked point and disable the coordinate display.
ResetView	Resets the view point to a view that frame the object.
Resize	Update the size of the 3D viewer window.

SetAutoRotate	Enables and configures the auto rotate display.
SetAxisOrigin	Sets and Gets the axis origin mode.
SetBackgroundColor	Sets/Gets the 3D viewer background color.
SetColorRampGraduationColor	Sets/Gets the graduation color.
SetColorRampLocation	Sets the location of the color ramp.
SetColorRampMode	Set/Gets the current color ramp mode.
SetEDLShadingFactor	Gets/Sets how strong Eye-Dome-Lighting shading is. In this mode, every pixel is shaded by how much closer its neighbor pixels are to the camera.
SetEnableEDLShading	Gets/Sets if Eye-Dome-Lighting shading mode is enabled. In this mode, every pixel is shaded by how much closer its neighbor pixels are to the camera.
SetEnableSmartColorRamp	Sets/Gets the state of the smart generation of color. If the smart generation of color is active, then the outliers inside the pointcloud will not be taken into account in the color ramp.
SetFeatureStyleFor3DObject	Sets how the E3DObjectFeature of the registered E3DObject at the specified location idx should be rendered.
SetFeatureStyleForAll3DObjects	Sets how the E3DObjectFeature of all the registered E3DObject should be rendered.
SetFieldOfView	Sets/Gets the field of view.
SetFixColorRampBounds	Sets the bounds of the color ramp.
SetFocus	The viewer window takes the focus.
SetFontPath	Configure the TrueType font used to display text on the 3D Viewer
SetPickedPointCallBack	Sets a callback function that will be called when a new E3DPoint is picked.
SetPickingDisplay	Enables or disables the display of the picked point.
SetPickingDistanceThreshold	Sets or gets the distance threshold for the picking.
SetPickingLabelColor	Sets or gets the color of the picked point label.
SetPickingLabelFixed	Sets or gets the state to fix or not the label.
SetPickingLabelSize	Sets or gets the size of the picked point label.
SetPointSize	Displays size of the points, in pixels.
SetPosition	Sets the position of the 3D viewer window.
SetProjectionType	Sets/Gets the projection type.
SetRenderAxis	Enables or disables the display of the axis.
SetRenderAxisConfiguration	Sets/Gets the axis configuration.

SetRenderDecimationLevel	Sets or Gets the point cloud decimation level used during the rendering.
SetRenderGrid	Enables or disables the display of Grid.
SetRenderGridStep	Sets/Gets the same grid step for each axis.
SetRenderSource	Changes the content of a render source with a new point cloud, mesh or ZMap.
SetRenderSourceColorMode	Sets the Source Color Mode. The Source Color Mode defines how the colors of the point cloud or mesh are chosen. See ESourceColorMode.
SetRenderSourceConstantColor	Sets the constant color used to display a point cloud or a mesh, when ESourceColorMode is set to ESourceColorMode_Constant.
SetRenderSourceOpacity	Sets the opacity used to display the render source. Only compatible with ESourceColorMode_Constant.
SetRenderSourcePointSize	Sets the point size used to display the point clouds.
SetRenderSourceWireFrame	Sets the wireframe mode used to render a mesh.
SetRenderSourceWireFrameColor	Sets the wireframe color.
SetRotationMatrix	Sets the view rotation matrix.
SetViewAngle	Sets view angles.
SetViewDistance	Sets/Gets view distance.
SetViewTarget	Sets view target position (by default, the camera position is 'look at center of the point cloud').
SetWireframeMode	Enables or disables the display of wireframe triangles.
Show	Shows the viewer window, refresh the display.
ShowColorRampLegend	Enables the display of a color ramp legend.
ShowFeatureFor3DObject	Sets the E3DObjectFeature of the registered E3DObject at the specified location idx to be rendered.
ShowFeatureForAll3DObjects	Sets the E3DObjectFeature of all the registered E3DObject to be rendered.
ShowRenderSource	Shows the given render source.
StopAutoRotate	Stops the display automatic rotation
ToggleRenderAxis	Toggles the display of the axis.
ToggleWireframeMode	Toggles the display of wireframe triangles.
UpdateRotationPosition	Updates a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when the mouse moves.
UpdateTranslationPosition	Updates a translation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is pressed.

UpdateViewDistance	Applies a delta factor to the view distance. Usually this control is mapped on the mouse wheel.

E3DViewer::AddRenderSource

Adds a new Render Source. The given render source (point cloud, mesh, ZMap, sphere, box, plane or line) is added to the current display.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddRenderSource(
 const std::string& name,
 const EPointCloud& source
 )

void AddRenderSource(
 const std::string& name,
 const EMesh& source
 )

void AddRenderSource(
 const std::string& name,
 const EZMap* source
 )

void AddRenderSource(
 const std::string& name,
 const E3DSphere& sphere,
 EC24 color,
 OEV_UINT8 opacity
 )

void AddRenderSource(
 const std::string& name,
 const E3DBox& box,
 EC24 color,
 OEV_UINT8 opacity
 )

void AddRenderSource(
 const std::string& name,
 const E3DLine& line,
 EC24 color
 )

void AddRenderSource(
 const std::string& name,
 const E3DPlane& plane,
 EC24 color,
 OEV_UINT8 opacity
 )
```

Parameters

name
A name for the render source, to be used to access and configure the render source.

source
An EPointCloud, EMesh or EZMap to be added as render source.

sphere
An E3DSphere to be added as render source.

color
The color of the E3DSphere, E3DBox, E3DLine or E3DPlane.

opacity
The opacity of the E3DSphere, E3DBox or E3DPlane.

box
An E3DBox to be added as render source.

line
An E3DLine to be added as render source.

plane
An E3DPlane to be added as render source.

E3DViewer::AddTextLabel

Adds a text label linked to an E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int AddTextLabel(
 const TextLabel& label
 )

int AddTextLabel(
 const E3DPoint& anchor,
 float posX,
 float posY,
 EC24 color,
 float size,
 const std::string& text,
 bool showAnchor,
 EC24A backgroundcolor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
```

```
int AddTextLabel(
 const E3DPoint& anchor,
 EC24 color,
 float size,
 const std::string& text,
 bool fixLabelPosition,
 bool showAnchor,
 EC24A backgroundcolor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
int AddTextLabel(
 float posX,
 float posY,
 EC24 color,
 float size,
 const std::string& text,
 EC24A backgroundcolor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
```

Parameters

label

The label to add.

anchor

The E3DPoint linked to the text label.

posX

The x coordinate of the text box. Value between -1 (left) and 1 (right).

posY

The y coordinate of the text box. Value between -1 (bottom) and 1 (top).

color

The color of the text label.

size

The size of the text font. Value between 0 et 1.

text

The text of the text label.

showAnchor

If set to true, a line is drawn between the anchor and the label (default: true).

backgroundcolor

If specified, the background of the label is set to this color (default: black).

alignment

-

fixLabelPosition

If set to true, the label stays fixed when the view changes (default: false).

Remarks

See also E3DViewer::EditTextLabel. and E3DViewer::GetTextLabel.

Deprecation notice: The overloads taking multiple arguments are deprecated.

E3DViewer::GetAxisOrigin

E3DViewer::SetAxisOrigin

Sets and Gets the axis origin mode.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::EAxisOriginMode GetAxisOrigin() const
void SetAxisOrigin(Euresys::Open_eVision::Easy3D::EAxisOriginMode mode)
```

Remarks

The default mode is EAxisOriginMode.

E3DViewer::GetBackgroundColor

E3DViewer::SetBackgroundColor

Sets/Gets the 3D viewer background color.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ERGBColor GetBackgroundColor() const
void SetBackgroundColor(ERGBColor backgroundColor)
```

E3DViewer::ClearRenderSource

Clears the 3D data, nothing will be displayed.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearRenderSource(
 )
```

Remarks

See also E3DViewer::ConfigureRenderSource

E3DViewer::ClearTextLabels

Removes all text labels.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearTextLabels(
 )
```

E3DViewer::GetColorRampGraduationColor

E3DViewer::SetColorRampGraduationColor

Sets/Gets the graduation color.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ERGBColor GetColorRampGraduationColor() const
void SetColorRampGraduationColor(ERGBColor color)
```

Remarks

The default color is white.

E3DViewer::GetColorRampMode

E3DViewer::SetColorRampMode

Set/Gets the current color ramp mode.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::EColorRampMode GetColorRampMode() const
void SetColorRampMode(Euresys::Open_eVision::Easy3D::EColorRampMode mode)
```

E3DViewer::ConfigureRenderSource

Sets a unique 3D source to be rendered. It replaces the current object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConfigureRenderSource(
 const EZMap* sourceObject,
 bool keepCurrentView
 )

void ConfigureRenderSource(
 const EPointCloud& sourceObject,
 bool keepCurrentView
 )

void ConfigureRenderSource(
 const EMesh& sourceObject,
 bool keepCurrentView
 )

void ConfigureRenderSource(
 const EZMap8& sourceObject,
 bool keepCurrentView
 )

void ConfigureRenderSource(
 const EZMap16& sourceObject,
 bool keepCurrentView
 )

void ConfigureRenderSource(
 const EZMap32f& sourceObject,
 bool keepCurrentView
 )
```

Parameters

sourceObject

A 3D source (EPointCloud, EMesh, EZMap) to render.

keepCurrentView

An optional boolean, use true to keep the current view or false to reset the view and center the new object.
The default value resets the view.

Remarks

For display performance purposes, the object geometry is copied into the viewer.
Subsequent modifications on the object will thus not be visible until a new call to E3DViewer::ConfigureRenderSource has been made.
The initial viewing position looks at the object center.

E3DViewer::DecPointSize

Decreases the displayed point size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DecPointSize(
 )
```

E3DViewer::DisableFixColorRampBounds

Disables the fix color ramp bounds.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DisableFixColorRampBounds(
 )
```

Remarks

See also E3DViewer

E3DViewer::E3DViewer

Creates an E3DViewer object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DViewer(
 int orgX,
 int orgY,
 int width,
 int height,
 void* parent
 )

void E3DViewer(
 Euresys::Open_eVision::Easy3D::EUIAPI uiApi,
 int orgX,
 int orgY,
 int width,
 int height,
 void* parent
 )
```

```
void E3DViewer(
 Euresys::Open_eVision::Easy3D::EUIAPI uiApi
 )
```

Parameters

orgX

X coordinate of the top left corner of the viewer window (only if uiApi is EUIAPI_Win32).

orgY

Y coordinate of the top left corner of the viewer window (only if uiApi is EUIAPI_Win32).

width

Width of the viewer window (only if uiApi is EUIAPI_Win32).

height

Height of the viewer window (only if uiApi is EUIAPI_Win32).

parent

Handle of the parent window of the viewer. If NULL, the viewer is built as a independent floating window (only if uiApi is EUIAPI_Win32).

uiApi

The User Interface API used by the parent application. See EUIAPI.

Remarks

The origin point (orgX, orgY) defines the offset of the top left corner of the viewer from the top left corner of its parent window client area.
If the window has no parent, it defines the offset from the top left corner of the screen.
If the parent window is too small to contain the viewer, the viewer will be cropped accordingly.

When the parent application that use E3DViewer is a Qt application, call the constructor with EUIAPI_Qt.

E3DViewer::EditTextLabel

Edits a text label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EditTextLabel(
 int id,
 const TextLabel& label
 )
```

```
void EditTextLabel(
 int id,
 float posX,
 float posY,
 EC24 color,
 float size,
 const std::string& text,
 bool showAnchor,
 EC24 backgroundcolor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
void EditTextLabel(
 int id,
 EC24 color,
 float size,
 const std::string& text,
 bool fixLabelPosition,
 bool showAnchor,
 EC24 backgroundcolor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
```

Parameters

id
The id of the text label to edit.

label
-

posX
The x coordinate of the text box. Value between -1 (left) and 1 (right).

posY
The y coordinate of the text box. Value between -1 (bottom) and 1 (top).

color
The color of the text label.

size
The size of the text font. Value between 0 et 1.

text
The text of the text label.

showAnchor
If set to true, a line is drawn between the anchor and the label (default: true).

backgroundcolor
If specified, the background of the label is set to this color (default: black).

alignment
-

fixLabelPosition
If set to true, the label stays fixed when the view changes (default: false).

Remarks

See also E3DViewer::AddTextLabel and E3DViewer::GetTextLabel.

Deprecation notice: The overloads taking more than two arguments are deprecated.

E3DViewer::GetEDLShadingFactor

E3DViewer::SetEDLShadingFactor

Gets/Sets how strong Eye-Dome-Lighting shading is. In this mode, every pixel is shaded by how much closer its neighbor pixels are to the camera.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetEDLShadingFactor() const
void SetEDLShadingFactor(float factor)
```

Remarks

Factor value ranges between 0 and 1. 0 means no shading, 1 means the strongest shading.

E3DViewer::GetEnableEDLShading

E3DViewer::SetEnableEDLShading

Gets/Sets if Eye-Dome-Lighting shading mode is enabled. In this mode, every pixel is shaded by how much closer its neighbor pixels are to the camera.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableEDLShading() const
void SetEnableEDLShading(bool state)
```

E3DViewer::GetEnableFixColorRampBounds

Gets the state of the fix color ramp bounds.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableFixColorRampBounds() const
```

Remarks

To enable fix color ramp bounds, use E3DViewer::SetFixColorRampBounds.

E3DViewer::GetEnableSmartColorRamp

E3DViewer::SetEnableSmartColorRamp

Sets/Gets the state of the smart generation of color. If the smart generation of color is active, then the outliers inside the pointcloud will not be taken into account in the color ramp.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetEnableSmartColorRamp() const
void SetEnableSmartColorRamp(bool state)
```

Remarks

Default: true. When set to true, fix color ramp bounds are disabled (see also E3DViewer::DisableFixColorRampBounds).

E3DViewer::GetFieldOfView

E3DViewer::SetFieldOfView

Sets/Gets the field of view.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetFieldOfView() const
void SetFieldOfView(float fieldOfView)
```

Remarks

If the projection type is EProjectionType_Orthographic, the field of view is not taken into account. See also E3DViewer::ProjectionType. For the angle unit see Easy::AngleUnit.

E3DViewer::GetFontPath

E3DViewer::SetFontPath

Configure the TrueType font used to display text on the 3D Viewer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetFontPath() const

void SetFontPath(const std::string& path)
```

Remarks

Setup a font file must be done before the E3DViewer::InitRendering method is called.
By default, the TTF file is:
- on Windows: C:\\Windows\\Fonts\\Arial.ttf
- on Linux: /usr/share/fonts/liberation-sans/LiberationSans-Regular.ttf or
- /usr/share/fonts/truetype/liberation/LiberationSans-Regular.ttf" or
- /usr/share/fonts/truetype/noto/NotoMono-Regular.ttf"

E3DViewer::GenerateColors

This method is deprecated.

Choose the current color ramp mode. Colors are calculated from point coordinates or attributes, several mappings are exposed in EColorRampMode.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GenerateColors(
 Euresys::Open_eVision::Easy3D::EColorRampMode mode
 )
```

Parameters

mode

The color ramp mode from EColorRampMode.

Remarks

This method is deprecated in favor of E3DViewer.

E3DViewer::GetAutoRotate

Gets the auto rotate speeds.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetAutoRotate(
 float& vx,
 float& vy,
 float& vz
 )
```

Parameters

vx
Rotation speed around axis X.

vy
Rotation speed around axis Y.

vz
Rotation speed around axis Z.

E3DViewer::GetColorRampLocation

Gets the location of the color ramp.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetColorRampLocation(
 float& xmin,
 float& xmax,
 float& ymin,
 float& ymax
 )
```

Parameters

xmin
The left most coordinate of the color ramp.

xmax
The right most coordinate of the color ramp.

ymin
The bottom most coordinate of the color ramp.

ymax
The top most coordinate of the color ramp.

E3DViewer::GetFeatureStyleFor3DObject

Gets how the E3DObjectFeature of the registered E3DObject at the specified location idx should be rendered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ERenderStyle GetFeatureStyleFor3DObject(
 int idx,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

idx

the position in the list of registered E3DObject

feature

the feature

E3DViewer::GetFixColorRampBounds

Gets the bounds of the color ramp.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetFixColorRampBounds(
 float& min,
 float& max
 )
```

Parameters

min

Where the lowest value of the color ramp will be stored.

max

Where the highest value of the color ramp will be stored.

Remarks

See also E3DViewer::DisableFixColorRampBounds

E3DViewer::GetRenderSourceColorMode

Gets the Source Color Mode. The Source Color Mode defines how the colors of the point cloud or mesh are chosen. See ESourceColorMode.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::ESourceColorMode GetRenderSourceColorMode(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be considered.

Remarks

See also E3DViewer::GetRenderSourceConstantColor and E3DViewer::GenerateColors.

E3DViewer::GetRenderSourceConstantColor

Gets the constant color used to display a point cloud or a mesh, when ESourceColorMode is set to Constant.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EC24 GetRenderSourceConstantColor(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be considered.

Remarks

See also E3DViewer::GetRenderSourceColorMode.

E3DViewer::GetRenderSourceName

Returns the name of the ith render source.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
std::string GetRenderSourceName(
 int index
 )
```

Parameters

index

The index of the render source to consider.

Remarks

The number of render sources is given by E3DViewer.

E3DViewer::GetRenderSourceOpacity

Gets the opacity used to display the render source. Only compatible with Constant.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
OEV_UINT8 GetRenderSourceOpacity(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be considered.

E3DViewer::GetRenderSourcePointSize

Gets the point size used to display the point clouds.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRenderSourcePointSize(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be considered.

E3DViewer::GetRenderSourceWireFrame

Gets the wireframe mode used to render a mesh.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetRenderSourceWireFrame(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be considered.

E3DViewer::GetRenderSourceWireFrameColor

Gets the wireframe color.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EC24 GetRenderSourceWireFrameColor(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be considered.

E3DViewer::GetRotationMatrix

Gets the view rotation matrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetRotationMatrix(
 E3DTransformMatrix& matrix
 )
```

Parameters

matrix

A matrix representing the view orientation.

E3DViewer::GetTextLabel

Gets the text label information.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
TextLabel GetTextLabel(
 int id
 )

void GetTextLabel(
 int id,
 E3DPoint& anchor,
 float& posX,
 float& posY,
 EC24& color,
 float& size,
 std::string& text,
 bool& fixLabelPosition,
 bool& showAnchor,
 EC24A& backgroundcolor
 )
```

```
void GetTextLabel(
 int id,
 E3DPoint& anchor,
 float& posX,
 float& posY,
 EC24& color,
 float& size,
 std::string& text,
 bool& fixLabelPosition,
 bool& showAnchor
 )
```

Parameters

id
The id of the text label.

anchor
The E3DPoint linked to the text label.

posX
The x coordinate of the text box. Value between -1 (left) and 1 (right).

posY
The y coordinate of the text box. Value between -1 (bottom) and 1 (top).

color
The color of the text label.

size
The size of the text font. Value between 0 et 1.

text
The text of the text label.

fixLabelPosition
If set to true, the label stays fixed when the view changes (default: false).

showAnchor
If set to true, a line is drawn between the anchor and the label.

backgroundcolor
The background color of the label.

Remarks

See also E3DViewer::AddTextLabel and E3DViewer::EditTextLabel.

Deprecation notice: The overloads taking multiple arguments are deprecated.

E3DViewer::GetViewAngle

Gets view angles.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetViewAngle(
 float& angleX,
 float& angleY,
 float& angleZ
 )
```

Parameters

angleX
Rotation around the X axis.

angleY
Rotation around the Y axis.

angleZ
Rotation around the Z axis.

E3DViewer::GetViewTarget

Gets view target position (by default, the camera position is 'look at center of the point cloud').

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetViewTarget(
 float& targetX,
 float& targetY,
 float& targetZ
 )
```

Parameters

targetX
X axis target position.

targetY
Y axis target position.

targetZ
Z axis target position.

E3DViewer::Has3DObjects

Return true if at least one 3D object is registered

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool Has3DObjects(
 )
```

Remarks

See also E3DViewer::Register3DObjects.

E3DViewer::HasRenderSource

Is the given render source name exists ?

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool HasRenderSource(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be checked.

E3DViewer::HideColorRampLegend

Disables the display of a color ramp legend.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void HideColorRampLegend(
 )
```

Remarks

See also E3DViewer::ShowColorRampLegend.

E3DViewer::HideFeatureFor3DObject

Sets the E3DObjectFeature of the registered E3DObject at the specified location idx to be not rendered.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void HideFeatureFor3DObject(
 int idx,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

idx
the position in the list of registered E3DObject

feature
the feature

E3DViewer::HideFeatureForAll3DObjects

Sets the E3DObjectFeature of all the registered E3DObject to be not rendered.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void HideFeatureForAll3DObjects(
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

feature
the feature

E3DViewer::HideRenderSource

Hides the given render source.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void HideRenderSource(
 const std::string& name
 )
```

Parameters

name
The name of the render source.

Remarks

See also E3DViewer::ShowRenderSource.

E3DViewer::IncPointSize

Increases the displayed point size.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void IncPointSize(
 )
```

E3DViewer::InitRendering

Initializes the rendering state, must be called one time before the first E3DViewer::Show.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void InitRendering(
 )
```

E3DViewer::IsAutoRotate

Returns true if the auto rotation is active.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsAutoRotate(
 )
```

E3DViewer::IsColorRampLegendVisible

Whether the color ramp legend is visible or not.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsColorRampLegendVisible(
 )
```

Remarks

See also E3DViewer::ShowColorRampLegend and E3DViewer::HideColorRampLegend.

E3DViewer::IsEDLShadingSupported

Indicates whether Eye-Dome-Lighting shading mode is supported.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsEDLShadingSupported(
 )
```

Remarks

Support of Eye-Dome-Lighting shading is detected when E3DViewer::InitRendering is called. Before, this method will always return false.

E3DViewer::IsFeatureVisibleFor3DObject

Whether the E3DObjectFeature of the registered E3DObject si rendered or not.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsFeatureVisibleFor3DObject(
 int idx,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

idx

the position in the list of registered E3DObject

feature

the feature

E3DViewer::IsRenderSourceVisible

Returns true if the render source is currently visible.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsRenderSourceVisible(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be queried.

Remarks

See also E3DViewer::ShowRenderSource and E3DViewer::HideRenderSource.

E3DViewer::GetLastPickedPoint

Returns the last picked E3DPoint if there is one. Otherwise, it throws an EException.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetLastPickedPoint() const
```

E3DViewer::LockRotationFinalPosition

Finalizes a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is released.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LockRotationFinalPosition(
 int x,
 int y
 )
```

Parameters

x

X coordinate.

y

Y coordinate.

Remarks

See also E3DViewer::LockRotationInitialPosition and E3DViewer::UpdateRotationPosition.

E3DViewer::LockRotationInitialPosition

Starts a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is pressed.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LockRotationInitialPosition(
 int x,
 int y
 )
```

Parameters

x

X coordinate.

y

Y coordinate.

Remarks

See also E3DViewer::UpdateRotationPosition and E3DViewer::LockRotationFinalPosition.

E3DViewer::LockTranslationFinalPosition

Finalizes a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is released.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LockTranslationFinalPosition(
 int x,
 int y
 )
```

Parameters

x

X coordinate.

y

Y coordinate.

Remarks

See also E3DViewer::LockTranslationInitialPosition and E3DViewer::UpdateTranslationPosition.

E3DViewer::LockTranslationInitialPosition

Starts a translation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is pressed.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LockTranslationInitialPosition(
 int x,
 int y
 )
```

Parameters

x

X coordinate.

y

Y coordinate.

Remarks

See also E3DViewer::UpdateTranslationPosition and E3DViewer::LockTranslationFinalPosition.

E3DViewer::GetNumRenderSources

Returns the number of render sources.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumRenderSources() const
```

Remarks

See also E3DViewer::GetRenderSourceName.

E3DViewer::Pick3DPoint

Returns and displays the picked E3DPoint in the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint Pick3DPoint(
 int x,
 int y
 )
```

Parameters

x

The X pixel coordinate in the 3DViewer windows

y

The Y pixel coordinate in the 3DViewer windows

Remarks

If there is no point close enough to the picking ray, an EException is thrown (see also E3DViewer::PickingDistanceThreshold). If a callback function is configured (see E3DViewer::SetPickedPointCallBack), then no EException is thrown when there is no point found.

E3DViewer::GetPickingDisplay

E3DViewer::SetPickingDisplay

Enables or disables the display of the picked point.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetPickingDisplay() const
void SetPickingDisplay(bool state)
```

Remarks

See also E3DViewer::PickingLabelSize, E3DViewer::PickingLabelFixed and E3DViewer::PickingLabelColor.

E3DViewer::GetPickingDistanceThreshold

E3DViewer::SetPickingDistanceThreshold

Sets or gets the distance threshold for the picking.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetPickingDistanceThreshold() const
void SetPickingDistanceThreshold(float thresh)
```

Remarks

See also E3DViewer::Pick3DPoint.

E3DViewer::GetPickingLabelColor

E3DViewer::SetPickingLabelColor

Sets or gets the color of the picked point label.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EC24 GetPickingLabelColor() const
void SetPickingLabelColor(EC24 color)
```

Remarks

See also E3DViewer::PickingDisplay, E3DViewer::PickingLabelFixed and E3DViewer::PickingLabelSize.

E3DViewer::GetPickingLabelFixed

E3DViewer::SetPickingLabelFixed

Sets or gets the state to fix or not the label.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetPickingLabelFixed() const
void SetPickingLabelFixed(bool state)
```

Remarks

Default state is false. See also E3DViewer::PickingDisplay, E3DViewer::PickingLabelSize and E3DViewer::PickingLabelColor.

E3DViewer::GetPickingLabelSize

E3DViewer::SetPickingLabelSize

Sets or gets the size of the picked point label.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetPickingLabelSize() const
void SetPickingLabelSize(float size)
```

Remarks

Default value for size is 0.05. See also E3DViewer::PickingDisplay, E3DViewer::PickingLabelFixed and E3DViewer::PickingLabelColor.

E3DViewer::GetPointSize

E3DViewer::SetPointSize

Displays size of the points, in pixels.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetPointSize() const
void SetPointSize(int size)
```

Remarks

The size of the point (range value is 1 to 5 pixels, and 2 by default).
This value is used only to draw an EPointCloud, not for an EMesh.

E3DViewer::GetProjectionType

E3DViewer::SetProjectionType

Sets/Gets the projection type.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EProjectionType GetProjectionType() const
void SetProjectionType(Euresys::Open_eVision::Easy3D::EProjectionType projectionType)
```

Remarks

If the projection type is EProjectionType_Perspective, then the field of view can be set with the method E3DViewer::FieldOfView.

E3DViewer::Register3DObjects

Sets the list of E3DObject from which their features can be rendered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Register3DObjects(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DObject>& objects
 )
```

Parameters

objects

List of E3DObject

Remarks

The features that need to be visualize are set with show methods. The registered features have a default style. Previous set styles is ignored. Remove the currently registered E3DObject.

E3DViewer::RemoveAllRenderSources

Removes all render sources. The E3DViewer display will be empty.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void RemoveAllRenderSources(
 )
```

E3DViewer::RemoveCurrent3DObjects

Removes all E3DObject currently registered.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void RemoveCurrent3DObjects(
 )
```

E3DViewer::RemoveRenderSource

Removes the given render source.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void RemoveRenderSource(
 const std::string& name
 )
```

Parameters

name

The name of the render source to be changed.

E3DViewer::RemoveTextLabel

Removes a text label.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void RemoveTextLabel(
 int id
 )
```

Parameters

id

The id of the text label.

Remarks

See also E3DViewer::AddTextLabel.

E3DViewer::GetRenderAxis

E3DViewer::SetRenderAxis

Enables or disables the display of the axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetRenderAxis() const
void SetRenderAxis(bool state)
```

Remarks

The default state is true.

E3DViewer::GetRenderAxisConfiguration

E3DViewer::SetRenderAxisConfiguration

Sets/Gets the axis configuration.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DAxisDisplay& GetRenderAxisConfiguration()
void SetRenderAxisConfiguration(const E3DAxisDisplay& axis)
```

E3DViewer::GetRenderDecimationLevel

E3DViewer::SetRenderDecimationLevel

Sets or Gets the point cloud decimation level used during the rendering.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRenderDecimationLevel() const
void SetRenderDecimationLevel(int decimationLevel)
```

Remarks

The viewer will only render one point every [Decimation Level] points (1 by default, and need to be > 0).
This decimation depends on the order of the points in the EPointCloud.
Irrespectively from this parameter, the viewer decimates the rendered point clouds if it detects lags in its display.

E3DViewer::GetRenderGrid

E3DViewer::SetRenderGrid

Enables or disables the display of Grid.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetRenderGrid() const
void SetRenderGrid(bool state)
```

Remarks

Display Grid with true (true by default).

E3DViewer::GetRenderGridStep

E3DViewer::SetRenderGridStep

Sets/Gets the same grid step for each axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetRenderGridStep() const
void SetRenderGridStep(float value)
```

Remarks

The unit of the grid step value is the same as the one from the EPointCloud. If value is equal to 0, then the step is auto computed and if the value is smaller than 0, then there is no step on the axis.
Default value is the size of each axis divided by ten.
For the Get function, if the step is not the same for each axis, then the mean is returned.

E3DViewer::ResetPicking

Resets the last picked point and disable the coordinate display.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ResetPicking(
 )
```

E3DViewer::ResetView

Resets the view point to a view that frame the object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ResetView(
 Euresys::Open_eVision::Easy3D::EViewDirection viewDirection
 )
```

Parameters

viewDirection
The view direction from EViewDirection (optional)

Remarks

The default view direction is from positive Z.

E3DViewer::Resize

Update the size of the 3D viewer window.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Resize(
 int width,
 int height
 )
```

Parameters

width
Width of the viewer window.

height
Height of the viewer window.

E3DViewer::SetAutoRotate

Enables and configures the auto rotate display.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetAutoRotate(
 float vx,
 float vy,
 float vz
 )
```

Parameters

vx
Rotation speed around axis X.

vy
Rotation speed around axis Y.

vz
Rotation speed around axis Z.

E3DViewer::SetColorRampLocation

Sets the location of the color ramp.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetColorRampLocation(
 float xmin,
 float xmax,
 float ymin,
 float ymax
 )
```

Parameters

xmin
The left most coordinate of the color ramp.

xmax
The right most coordinate of the color ramp.

ymin
The bottom most coordinate of the color ramp.

ymax
The top most coordinate of the color ramp.

Remarks

By default, the color ramp is located at the right of the window. The color ramp is always vertical. The value should be between 0 (left/bottom) and 100 (right/top) and corresponds to the % of the window.

E3DViewer::SetFeatureStyleFor3DObject

Sets how the E3DObjectFeature of the registered E3DObject at the specified location idx should be rendered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFeatureStyleFor3DObject(
 int idx,
 const ERenderStyle& style,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

idx
-

style
the style

feature
the feature

E3DViewer::SetFeatureStyleForAll3DObjects

Sets how the E3DObjectFeature of all the registered E3DObject should be rendered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFeatureStyleForAll3DObjects(
 const ERenderStyle& style,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

style
the style

feature
the feature

E3DViewer::SetFixColorRampBounds

Sets the bounds of the color ramp.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFixColorRampBounds(
 float min,
 float max
 )
```

Parameters

min
The lowest value of the color ramp.

max
The highest value of the color ramp.

Remarks

This function also sets to false E3DViewer::EnableSmartColorRamp. See also E3DViewer::DisableFixColorRampBounds

E3DViewer::SetFocus

The viewer window takes the focus.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFocus(
 )
```

E3DViewer::SetPickedPointCallBack

Sets a callback function that will be called when a new E3DPoint is picked.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPickedPointCallBack(
 void* pickedPointCallBack,
 void* context,
 bool callCallBackWhenPointNotFound
 )
```

Parameters

pickedPointCallBack

A pointer to a function that will be called when a new E3DPoint is picked.

context

Some context information that will be given in argument to the callback pickedPointCallBack function.

callCallBackWhenPointNotFound

Set to true to call the callback function even if there is not point picked. Default: false.

Remarks

The function pointer should be of type: `void CallBackFunction(void* context, bool pickedPointFound, float pX, float pY, float pZ, int x, int y)`
Where context is the context parameter that was given in argument, pickedPointFound is set to false if a there were no point close enough to the picking ray (true otherwise), pX, pY, pZ are the coordinates of the point that was picked, x and y are the position of the mouse in the window.

E3DViewer::SetPosition

Sets the position of the 3D viewer window.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPosition(
 int orgX,
 int orgY,
 int width,
 int height,
 void* hWndInsertAfter
 )
```

Parameters

orgX

X coordinate of the top left corner of the viewer window.

orgY

Y coordinate of the top left corner of the viewer window.

width

Width of the viewer window.

height

Height of the viewer window.

hWndInsertAfter

A handle to the window to precede the positioned window in the Z order. Only useful with EUIAPI_Win32. see https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-setwindowpos

E3DViewer::SetRenderSource

Changes the content of a render source with a new point cloud, mesh or ZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetRenderSource(
 const std::string& name,
 const EPointCloud& source
 )

void SetRenderSource(
 const std::string& name,
 const EMesh& source
 )

void SetRenderSource(
 const std::string& name,
 const EZMap* source
 )

void SetRenderSource(
 const std::string& name,
 const E3DSphere& sphere,
 EC24 color,
 OEV_UINT8 opacity
 )

void SetRenderSource(
 const std::string& name,
 const E3DBox& box,
 EC24 color,
 OEV_UINT8 opacity
 )
```

```
void SetRenderSource(
 const std::string& name,
 const E3DLine& line,
 EC24 color
 )
void SetRenderSource(
 const std::string& name,
 const E3DPlane& plane,
 EC24 color,
 OEV_UINT8 opacity
 )
```

Parameters

name
The name of the render source to be changed.

source
An EPointCloud, an EMesh or an EZMap to replace the existing render source.

sphere
An E3DSphere to replace the existing render source.

color
The color of the E3DSphere, E3DBox, E3DLine or E3DPlane.

opacity
The opacity of the E3DSphere, E3DBox or E3DPlane.

box
An E3DBox to replace the existing render source.

line
An E3DLine to replace the existing render source.

plane
An E3DPlane to replace the existing render source.

E3DViewer::SetRenderSourceColorMode

Sets the Source Color Mode. The Source Color Mode defines how the colors of the point cloud or mesh are chosen. See ESourceColorMode.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetRenderSourceColorMode(
 const std::string& name,
 Euresys::Open_eVision::Easy3D::ESourceColorMode colorMode
 )
```

Parameters

name

The name of the render source to be considered.

colorMode

The source color mode.

Remarks

See also E3DViewer::SetRenderSourceConstantColor and E3DViewer::GenerateColors.

E3DViewer::SetRenderSourceConstantColor

Sets the constant color used to display a point cloud or a mesh, when ESourceColorMode is set to Constant.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetRenderSourceConstantColor(
 const std::string& name,
 EC24 color
 )
```

Parameters

name

The name of the render source to be considered.

color

The color.

Remarks

See also E3DViewer::SetRenderSourceColorMode.

E3DViewer::SetRenderSourceOpacity

Sets the opacity used to display the render source. Only compatible with Constant.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetRenderSourceOpacity(
 const std::string& name,
 OEV_UINT8 opacity
 )
```

Parameters

name
The name of the render source to be considered.

opacity
The opacity between fully transparent (0) and fully opaque (255).

E3DViewer::SetRenderSourcePointSize

Sets the point size used to display the point clouds.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetRenderSourcePointSize(
 const std::string& name,
 int size
 )
```

Parameters

name
The name of the render source to be considered.

size
The number of pixels used to render a point.

E3DViewer::SetRenderSourceWireFrame

Sets the wireframe mode used to render a mesh.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetRenderSourceWireFrame(
 const std::string& name,
 bool state
 )
```

Parameters

name
The name of the render source to be considered.

state
Enable or disable the wireframe rendering mode.

E3DViewer::SetRenderSourceWireFrameColor

Sets the wireframe color.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetRenderSourceWireFrameColor(
 const std::string& name,
 EC24 color
 )
```

Parameters

name

The name of the render source to be considered.

color

-

E3DViewer::SetRotationMatrix

Sets the view rotation matrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetRotationMatrix(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix

A matrix representing the view orientation.

E3DViewer::SetViewAngle

Sets view angles.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetViewAngle(
 float angleX,
 float angleY,
 float angleZ
 )
```

Parameters

angleX
Rotation around the X axis.

angleY
Rotation around the Y axis.

angleZ
Rotation around the Z axis. Default: 0

E3DViewer::SetViewTarget

Sets view target position (by default, the camera position is 'look at center of the point cloud').

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetViewTarget(
 float targetX,
 float targetY,
 float targetZ
 )
```

Parameters

targetX
X axis target position.

targetY
Y axis target position.

targetZ
Z axis target position.

E3DViewer::Show

Shows the viewer window, refresh the display.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Show(
 )
```

E3DViewer::ShowColorRampLegend

Enables the display of a color ramp legend.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ShowColorRampLegend(
 )
```

Remarks

See also E3DViewer::HideColorRampLegend, E3DViewer, E3DViewer::ColorRampGraduationColor.

E3DViewer::ShowFeatureFor3DObject

Sets the E3DObjectFeature of the registered E3DObject at the specified location idx to be rendered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ShowFeatureFor3DObject(
 int idx,
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

idx

the position in the list of registered E3DObject

feature

the feature

E3DViewer::ShowFeatureForAll3DObjects

Sets the E3DObjectFeature of all the registered E3DObject to be rendered.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ShowFeatureForAll3DObjects(
 Euresys::Open_eVision::Easy3D::E3DObjectFeature feature
 )
```

Parameters

feature

the feature

E3DViewer::ShowRenderSource

Shows the given render source.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ShowRenderSource(
 const std::string& name
 )
```

Parameters

name

The name of the render source.

Remarks

See also E3DViewer::HideRenderSource.

E3DViewer::StopAutoRotate

Stops the display automatic rotation

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void StopAutoRotate(
 )
```

E3DViewer::ToggleRenderAxis

Toggles the display of the axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ToggleRenderAxis(
 )
```

E3DViewer::ToggleWireframeMode

Toggles the display of wireframe triangles.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ToggleWireframeMode(
 )
```

E3DViewer::UpdateRotationPosition

Updates a rotation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when the mouse moves.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UpdateRotationPosition(
 int x,
 int y
 )
```

Parameters

x

X coordinate.

y

Y coordinate.

Remarks

See also E3DViewer::LockRotationInitialPosition and E3DViewer::LockRotationFinalPosition.

E3DViewer::UpdateTranslationPosition

Updates a translation transformation of the view point using the (x,y) coordinate. Usually (x,y) is the mouse cursor coordinate when a button is pressed.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UpdateTranslationPosition(
 int x,
 int y
 )
```

Parameters

x

X coordinate.

y

Y coordinate.

Remarks

See also E3DViewer::LockTranslationInitialPosition and E3DViewer::LockTranslationFinalPosition.

E3DViewer::UpdateViewDistance

Applies a delta factor to the view distance. Usually this control is mapped on the mouse wheel.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UpdateViewDistance(
 float delta
 )
```

Parameters

delta

The factor to change the view distance: move the view point closer to the object when delta is lower than 1 and further to the object when delta is larger than 1.

E3DViewer::GetViewDistance

E3DViewer::SetViewDistance

Sets/Gets view distance.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetViewDistance() const
void SetViewDistance(float distance)
```

Remarks

Distance between the point of the view and the object.

E3DViewer::GetWireframeMode

E3DViewer::SetWireframeMode

Enables or disables the display of wireframe triangles.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetWireframeMode() const
void SetWireframeMode(bool state)
```

Remarks

Display wireframe triangles with true (false by default).

4.19. EAffineTransformer Class

Manages a 3D coordinates transformation context.

Remarks

By default, no transformation is done (identity matrix). The transformations are applied in the order in which the calls to AddTransform are done.

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddAnisotropicScalingTransform	Adds anisotropic scaling to the current E3DTransformMatrix.
AddIsotropicScalingTransform	Adds isotropic scaling to the current E3DTransformMatrix.
AddOrthographicProjectionTransform	Adds an orthographic projection to the current E3DTransformMatrix.
AddPerspectiveProjectionTransform	Adds a perspective projection to the current E3DTransformMatrix.
AddRotationXTransform	Adds rotation around the X axis to the current E3DTransformMatrix.
AddRotationYTransform	Adds rotation around the Y axis to the current E3DTransformMatrix.
AddRotationZTransform	Adds rotation around the Z axis to the current E3DTransformMatrix.
AddTransform	Composes a custom transformation with the current E3DTransformMatrix.
AddTranslationTransform	Adds translation to the current E3DTransformMatrix.
ApplyMatrix	Applies a E3DTransformMatrix to a EPointCloud or a points list. If the second parameter is present, puts the transformed points in another point cloud or points list. With a single parameter, the transformation is performed in place.
ApplyTransform	Applies the current transformation to a EPointCloud or a points list. If the second parameter is present, puts the transformed points in another point cloud or points list. With a single parameter, transformation is performed in place.
CreateAnisotropicScalingMatrix	Creates an anisotropic scaling E3DTransformMatrix.
CreateIdentityMatrix	Creates an identity (neutral) E3DTransformMatrix.

CreateIsotropicScaling Matrix	Creates an isotropic scaling E3DTransformMatrix.
CreateOrthographicProjectionMatrix	Creates an orthographic projection E3DTransformMatrix.
CreatePerspectiveProjectionMatrix	Creates a perspective projection E3DTransformMatrix.
CreateRotationXMatrix	Creates a rotation E3DTransformMatrix around the X axis.
CreateRotationYMatrix	Creates a rotation E3DTransformMatrix around the Y axis.
CreateRotationZMatrix	Creates a rotation E3DTransformMatrix around the Z axis.
CreateTranslationMatrix	Creates a translation E3DTransformMatrix.
EAffineTransformer	Creates an EAffineTransformer object.
GetTransform	Sets the E3DTransformMatrix that will be used.
operator=	Assignment operator.
Reset	Resets the transformation E3DTransformMatrix to the identity.
SetTransform	Sets the E3DTransformMatrix that will be used.

EAffineTransformer::AddAnisotropicScalingTransform

Adds anisotropic scaling to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddAnisotropicScalingTransform(
 float scaleX,
 float scaleY,
 float scaleZ
 )
```

Parameters

scaleX

Scaling factor along the X axis.

scaleY

Scaling factor along the Y axis.

scaleZ

Scaling factor along the Z axis.

EAffineTransformer::AddIsotropicScalingTransform

Adds isotropic scaling to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddIsotropicScalingTransform(
 float scale
 )
```

Parameters

scale

Scaling factor.

EAffineTransformer::AddOrthographicProjectionTransform

Adds an orthographic projection to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddOrthographicProjectionTransform(
 float width,
 float height
 )
```

Parameters

width

Width of the viewport.

height

Height of the viewport.

EAffineTransformer::AddPerspectiveProjectionTransform

Adds a perspective projection to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddPerspectiveProjectionTransform(
 float distance,
 float width,
 float height
 )
```

Parameters

distance
Distance of the viewport to the origin.
width
Width of the viewport.
height
Height of the viewport.

EAffineTransformer::AddRotationXTransform

Adds rotation around the X axis to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const E3DTransformMatrix& AddRotationXTransform(
 float Angle
 )
```

Parameters

Angle
Rotation angle.

EAffineTransformer::AddRotationYTransform

Adds rotation around the Y axis to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const E3DTransformMatrix& AddRotationYTransform(
 float Angle
 )
```

Parameters

Angle
Rotation angle.

EAffineTransformer::AddRotationZTransform

Adds rotation around the Z axis to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddRotationZTransform(
 float Angle
 )
```

Parameters

Angle

Rotation angle.

EAffineTransformer::AddTransform

Composes a custom transformation with the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddTransform(
 const E3DTransformMatrix& matrix
 )
```

Parameters

matrix

Transformation matrix.

EAffineTransformer::AddTranslationTransform

Adds translation to the current E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& AddTranslationTransform(
 float dX,
 float dY,
 float dZ
 )
```

Parameters

dX

Translation along the X axis.

dY

Translation along the Y axis.

dZ

Translation along the Z axis.

EAffineTransformer::ApplyMatrix

Applies a E3DTransformMatrix to a EPointCloud or a points list.
If the second parameter is present, puts the transformed points in another point cloud or points list.
With a single parameter, the transformation is performed in place.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ApplyMatrix(
 const E3DTransformMatrix& matrix,
 const EPointCloud& cloud,
 EPointCloud& transformedCloud
 )

void ApplyMatrix(
 const E3DTransformMatrix& matrix,
 EPointCloud& cloud
 )

void ApplyMatrix(
 const E3DTransformMatrix& matrix,
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& sourcePoints,
 std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& transformedPoints
 )
```

Parameters

matrix
Transformation matrix.

cloud
Cloud to transform.

transformedCloud
Transformed cloud.

sourcePoints
Points list to transform.

transformedPoints
Transformed points list.

EAffineTransformer::ApplyTransform

Applies the current transformation to a EPointCloud or a points list.
If the second parameter is present, puts the transformed points in another point cloud or points list.
With a single parameter, transformation is performed in place.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ApplyTransform(
 const EPointCloud& cloud,
 EPointCloud& transformedCloud
 )

void ApplyTransform(
 EPointCloud& cloud
 )

void ApplyTransform(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& sourcePoints,
 std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& transformedPoints
 )
```

Parameters

cloud
Cloud to transform.

transformedCloud
Transformed cloud.

sourcePoints
Points list to transform.

transformedPoints
Transformed points list.

EAffineTransformer::CreateAnisotropicScalingMatrix

Creates an anisotropic scaling E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix CreateAnisotropicScalingMatrix(
 float scaleX,
 float scaleY,
 float scaleZ
 )
```

Parameters

scaleX
Scaling factor along the X axis.

scaleY
Scaling factor along the Y axis.

scaleZ
Scaling factor along the Z axis.

EAffineTransformer::CreateIdentityMatrix

Creates an identity (neutral) E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateIdentityMatrix(
 )
```

EAffineTransformer::CreateIsotropicScalingMatrix

Creates an isotropic scaling E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateIsotropicScalingMatrix(
 float scale
 )
```

Parameters

scale
Scaling factor.

EAffineTransformer::CreateOrthographicProjectionMatrix

Creates an orthographic projection E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateOrthographicProjectionMatrix(
 float width,
 float height
 )
```

Parameters

width
Width of the viewport.
height
Height of the viewport.

EAffineTransformer::CreatePerspectiveProjectionMatrix

Creates a perspective projection E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreatePerspectiveProjectionMatrix(
 float distance,
 float width,
 float height
 )
```

Parameters

distance
Distance of the viewport to the origin.

width
Width of the viewport.

height
Height of the viewport.

EAffineTransformer::CreateRotationXMatrix

Creates a rotation E3DTransformMatrix around the X axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateRotationXMatrix(
 float Angle
 )
```

Parameters

Angle
Rotation angle.

EAffineTransformer::CreateRotationYMatrix

Creates a rotation E3DTransformMatrix around the Y axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateRotationYMatrix(
 float Angle
 )
```

Parameters

Angle

Rotation angle.

EAffineTransformer::CreateRotationZMatrix

Creates a rotation E3DTransformMatrix around the Z axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateRotationZMatrix(
 float Angle
 )
```

Parameters

Angle

Rotation angle.

EAffineTransformer::CreateTranslationMatrix

Creates a translation E3DTransformMatrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DTransformMatrix CreateTranslationMatrix(
 float dX,
 float dY,
 float dZ
 )
```

Parameters

dX

Translation along the X axis.

dY

Translation along the Y axis.

dZ

Translation along the Z axis.

EAffineTransformer::EAffineTransformer

Creates an EAffineTransformer object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EAffineTransformer(
 )
void EAffineTransformer(
 const EAffineTransformer& other
 )
```

Parameters

other

Another EAffineTransformer.

EAffineTransformer::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EAffineTransformer& operator=(
 const EAffineTransformer& other
 )
```

Parameters

other

Another EAffineTransformer.

EAffineTransformer::Reset

Resets the transformation E3DTransformMatrix to the identity.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Reset(
 )
```

EAffineTransformer::GetTransform

EAffineTransformer::SetTransform

Sets the E3DTransformMatrix that will be used.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetTransform() const
void SetTransform(const E3DTransformMatrix& matrix)
```

4.20. EAngleRectifier Class

-

Namespace: Euresys::Open_eVision

Methods

Rectify	-

EAngleRectifier::Rectify

-

Namespace: Euresys::Open_eVision

[C++]

```
double Rectify(
 double angle,
 double mean
 )
double Rectify(
 double angle,
 double mean,
 Euresys::Open_eVision::EAngleUnit currentUnit
 )
```

Parameters

angle
-

mean
-

currentUnit
-

4.21. Easy Class

This class contains static properties and methods specific to the Easy library.

Namespace: Euresys::Open_eVision

Methods

CheckLicense	Checks if a given license is available.
CheckLicenses	Check if at least one license is available. Otherwise, an exception is thrown.
CheckOemKey	Checks if the OEM key, if any, matches a given argument.
CloseImageGraphicContext	Releases the device context associated to an image. Deprecated: use EWindowsDrawAdapter class by passing an image to its constructor instead.
FromDegrees	Returns the angle, converted from degrees to the current angle unit.
FromRadians	Returns the angle, converted from radians to the current angle unit.
GetAngleUnit	Current angular unit.
GetBestMatchingImageType	Returns the best matching image type for a given file on disk.
GetDongleCount	Get the number of available dongle on the system.
GetDongleInternalSerialNumber	Get the serial number of the selected dongle.
GetEnableEnhancedImageDisplay	The enhanced mode for displaying the images.
GetErrorText	Returns the description associated to a given error code.
GetGPUComputeCapability	CUDA compute capability for the specified GPU.
GetGPUName	Name of the GPU.
GetMaxNumberOfProcessingThreads	Maximum number of threads used internally by the Open eVision tools (default value: 1). This number cannot be higher than the number of processor cores available to the system. See Easy::NumberOfAvailableProcessorCores. This value is thread local. It means that this value can be controlled independently for each thread you create.
GetNumberOfAvailableProcessorCores	Number of processor cores available to the system. This is the upper limit for the number of threads usable internally by the Open eVision tools. See Easy::MaxNumberOfProcessingThreads
GetNumGPUs	Number of GPUs available to the system.
GetResourcesRootPath	Retrieves the resources installation path.
GetSampleImagesRootPath	Retrieves the sample images installation path.
GetSampleProgramsRootPath	Retrieves the sample programs installation path.
GetVersion	Returns a pointer to a NULL terminated character string that contains the current version number of Open eVision.

Initialize	Initializes Open eVision.
IsGPUAvailable	Indicates if a supported GPU is available for Open eVision computation.
LogInMemento	Logs a message in eGrabber Memento
OpenImageGraphicContext	Allows to draw in a gray-level or a color image, using any of the Windows device context functions (the image contents is altered, allowing destructive overlays). Deprecated: use EWindowsDrawAdapter class by passing an image to its constructor instead.
Render3D	Prepares a three dimensional rendering of an image where the gray-level values are taken for altitudes.
RenderColorHistogram	Prepares a three dimensional rendering of the histogram of a color image: the pixels are drawn in the RGB space rather than in the XY plane.
Resize	Resizes an image without interpolation.
SetAngleUnit	Current angular unit.
SetEnableEnhancedImageDisplay	The enhanced mode for displaying the images.
SetMaxNumberOfProcessingThreads	Maximum number of threads used internally by the Open eVision tools (default value: 1). This number cannot be higher than the number of processor cores available to the system. See Easy::NumberOfAvailableProcessorCores. This value is thread local. It means that this value can be controlled independently for each thread you create.
SetOemKey	Writes an OEM key value on a dongle.
StartTiming	Starts timing, using the system clock or performance counter.
StopTiming	Returns the time, in specified time units, elapsed since the last invocation of Easy::StartTiming.
Terminate	Method not obligatory, but necessary for close dll cleanly.
ToDegrees	Returns the angle, converted from current angle unit to degrees.
ToRadians	Returns the angle, converted from current angle unit to radians.
TrueTimingResolution	Returns the actual resolution of the timing clock, in ticks per seconds.

Easy::GetAngleUnit

Easy::SetAngleUnit

Current angular unit.

Namespace: Euresys::Open_eVision

[C++]

```
static Euresys::Open_eVision::EAngleUnit GetAngleUnit()
```

```
static void SetAngleUnit(Euresys::Open_eVision::EAngleUnit unit)
```

Remarks

All angles are computed using some angular unit, as well on input as on output. The desired unit can be changed at any time. By default, all angles are given in degrees (0..360).

Easy::CheckLicense

Checks if a given license is available.

Namespace: Euresys::Open_eVision

[C++]

```
bool CheckLicense(
 Euresys::Open_eVision::LicenseFeatures::Features license
 )
```

Parameters

license

The license to check

Easy::CheckLicenses

Check if at least one license is available. Otherwise, an exception is thrown.

Namespace: Euresys::Open_eVision

[C++]

```
void CheckLicenses(
 )
```

Easy::CheckOemKey

Checks if the OEM key, if any, matches a given argument.

Namespace: Euresys::Open_eVision

[C++]

```
bool CheckOemKey(
 const std::vector<char>& key,
 Euresys::Open_eVision::EDongleType type,
 int dongleIndex
 )
```

```
bool CheckOemKey(
 OEV_UINT32 keyIndex,
 const std::vector<char>& key,
 Euresys::Open_eVision::EDongleType type,
 int dongleIndex
 )
```

Parameters

key

The expected value of the OEM key

type

The EDongleType for which you want to check the OEM key.

dongleIndex

The index of the dongle where the OEM key is expected. By default, the first dongle found is selected.

keyIndex

The index of the key in the array of keys, must be in [0, 11].
This option is not compatible with EDongleType_Legacy

Remarks

The length of the OEM key must be in [8, 64] characters.

Easy::CloseImageGraphicContext

This method is deprecated.

Releases the device context associated to an image. Deprecated: use EWindowsDrawAdapter class by passing an image to its constructor instead.

Namespace: Euresys::Open_eVision

[C++]

```
void CloseImageGraphicContext(
 EImageBW8* pImage,
 HDC hDC
 )

void CloseImageGraphicContext(
 EImageC24* pImage,
 HDC hDC
 )
```

Parameters

pImage

Pointer to the target image (must be the same as that passed to Easy::OpenImageGraphicContext).

hDC

Handle to a device context that was produced by Easy::OpenImageGraphicContext.

Easy::GetEnableEnhancedImageDisplay

Easy::SetEnableEnhancedImageDisplay

The enhanced mode for displaying the images.

Namespace: Euresys::Open_eVision

[C++]

```
static bool GetEnableEnhancedImageDisplay()
static void SetEnableEnhancedImageDisplay(bool state)
```

Remarks

When enabled, enhanced mode performs advanced interpolation for image display. By default, enhanced mode is disabled.
Enhanced mode has a significant impact on drawing performances. Enhanced display mode is set for the current thread.

Easy::FromDegrees

Returns the angle, converted from degrees to the current angle unit.

Namespace: Euresys::Open_eVision

[C++]

```
float FromDegrees(
 float angle
 )
```

Parameters

angle

Angle to be converted

Easy::FromRadians

Returns the angle, converted from radians to the current angle unit.

Namespace: Euresys::Open_eVision

[C++]

```
float FromRadians(
 float angle
 )
```

Parameters

angle

Angle to be converted

Easy::GetBestMatchingImageType

Returns the best matching image type for a given file on disk.

Namespace: Euresys::Open_eVision

```
[C++]

Euresys::Open_eVision::EImageType GetBestMatchingImageType(
 const std::string& path
 )
```

Parameters

path

The path to the file on disk.

Easy::GetDongleCount

Get the number of available dongle on the system.

Namespace: Euresys::Open_eVision

```
[C++]

OEV_UINT32 GetDongleCount(
 Euresys::Open_eVision::EDongleType type
 )
```

Parameters

type

The EDongleType you want to enumerate.

Easy::GetDongleInternalSerialNumber

Get the serial number of the selected dongle.

Namespace: Euresys::Open_eVision

```
[C++]

std::string GetDongleInternalSerialNumber(
 Euresys::Open_eVision::EDongleType type,
 int index
 )
```

Parameters

type
The EDongleType.

index
The index of the dongle.

Easy::GetErrorText

Returns the description associated to a given error code.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetErrorText(
 Euresys::Open_eVision::EError error
 )
```

Parameters

error
Error code.

Easy::GetGPUComputeCapability

CUDA compute capability for the specified GPU.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetGPUComputeCapability(
 int gpuId
 )
```

Parameters

gpuId
Index of the GPU between 0 and Easy::NumGPUs - 1.

Easy::GetGPUName

Name of the GPU.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetGPUName(
 int gpuId
 )
```

Parameters

gpuId

Index of the GPU between 0 and Easy::NumGPUs - 1.

Easy::Initialize

Initializes Open eVision.

Namespace: Euresys::Open_eVision

[C++]

```
void Initialize(
 )
```

Easy::IsGPUAvailable

Indicates if a supported GPU is available for Open eVision computation.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsGPUAvailable(
 )
```

Easy::LogInMemento

Logs a message in eGrabber Memento

Namespace: Euresys::Open_eVision

[C++]

```
void LogInMemento(
 const std::string& message,
 Euresys::Open_eVision::EVerbosity verbosity,
 OEV_UINT8 user
 )
```

Parameters

message

The message to log.

verbosity

The verbosity level of the message.

user

The user index, from 0 to 15

Remarks

The default verbosity level is EVerbosity_Info and the default user ID is 0. For more information about how Memento handles messages, please refer to the Memento documentation.

Easy::GetMaxNumberOfProcessingThreads

Easy::SetMaxNumberOfProcessingThreads

Maximum number of threads used internally by the Open eVision tools (default value: 1). This number cannot be higher than the number of processor cores available to the system. See Easy::NumberOfAvailableProcessorCores. This value is thread local. It means that this value can be controlled independently for each thread you create.

Namespace: Euresys::Open_eVision

[C++]

```
static int GetMaxNumberOfProcessingThreads()
static void SetMaxNumberOfProcessingThreads(int maxNumThreads)
```

Easy::GetNumberOfAvailableProcessorCores

Number of processor cores available to the system. This is the upper limit for the number of threads usable internally by the Open eVision tools. See Easy::MaxNumberOfProcessingThreads

Namespace: Euresys::Open_eVision

[C++]

```
static int GetNumberOfAvailableProcessorCores()
```

Easy::GetNumGPUs

Number of GPUs available to the system.

Namespace: Euresys::Open_eVision

[C++]

```
static int GetNumGPUs()
```

Easy::OpenImageGraphicContext

This method is deprecated.

Allows to draw in a gray-level or a color image, using any of the Windows device context functions (the image contents is altered, allowing destructive overlays). Deprecated: use EWindowsDrawAdapter class by passing an image to its constructor instead.

Namespace: Euresys::Open_eVision

[C++]

```
HDC OpenImageGraphicContext(
 EImageBW8* pImage
 )

HDC OpenImageGraphicContext(
 EImageC24* pImage
 )
```

Parameters

pImage

Pointer to the target image.

Remarks

The function returns a handle to a device context associated to the image pixel data. When the device context is no more needed, call the Easy::CloseImageGraphicContext function with the same argument.

Easy::Render3D

Prepares a three dimensional rendering of an image where the gray-level values are taken for altitudes.

Namespace: Euresys::Open_eVision

[C++]

```
void Render3D(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 float phi,
 float psi,
 float xScale,
 float yScale,
 float zScale,
 int dotSize
 )
void Render3D(
 EROIC24* sourceImage,
 EROIBW8* zImage,
 EROIC24* destinationImage,
 float phi,
 float psi,
 float xScale,
 float yScale,
 float zScale,
 int dotSize
 )
```

Parameters

sourceImage
Pointer to the source image.

destinationImage
Pointer to the destination image.

phi
Rotation angle about the X-axis.

psi
Rotation angle about the Y-axis.

xScale
Magnification factor along X (should remain close to 1).

yScale
Magnification factor along Y (should remain close to 1).

zScale
Magnification factor along Z (should remain close to 1).

dotSize
Size of the rendered dots; allowed values are 1, 4, 5 or 9.

zImage
Pointer to the altitude image.

Remarks

The image is viewed by rotating it about the X-axis, then about the Y-axis. Magnification factors in the three directions (X = width, Y = height, and Z = depth) can be given.
The rendered image appears as independent dots. The dot size can be adjusted so that the surface appears more or less opaque.
The function does not display the rendered image by itself. Rather, it prepares a destination image to be displayed.

Easy::RenderColorHistogram

Prepares a three dimensional rendering of the histogram of a color image: the pixels are drawn in the RGB space rather than in the XY plane.

Namespace: Euresys::Open_eVision

[C++]

```
void RenderColorHistogram(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 float phi,
 float psi,
 float xScale,
 float yScale,
 float zScale
 )
void RenderColorHistogram(
 EROIC24* sourceImage,
 EROIC24* sysImage,
 EROIC24* destinationImage,
 float phi,
 float psi,
 float xScale,
 float yScale,
 float zScale
 )
```

Parameters

sourceImage
Pointer to the raw source image.

destinationImage
Pointer to the destination image.

phi
Rotation angle about the X-axis.

psi
Rotation angle about the Y-axis.

xScale
Magnification factor along X (should remain close to 1).

yScale
Magnification factor along Y (should remain close to 1).

zScale

Magnification factor along Z (should remain close to 1).

sysImage

Pointer to the source image transformed into another color system.

Remarks

This allows to observe the clustering and dispersion of the RGB values.
The image is viewed by rotating it about the X-axis, then about the Y-axis. Magnification factors in the three directions (X = Red, Y = Green, and Z = Blue) can be given.
In a more advanced version, prepares a three dimensional rendering of the pixels in another system than RGB (EasyColor provides conversion means). However, the raw RGB image must still be provided to allow the display of the pixels in their usual colors.
The rendered image appears as independent dots.
The function does not display the rendered image by itself. Rather, it prepares a destination image to be displayed.

Easy::Resize

Resizes an image without interpolation.

Namespace: Euresys::Open_eVision

[C++]

```
void Resize(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Resize(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void Resize(
 EROIC15* sourceImage,
 EROIC15* destinationImage
 )

void Resize(
 EROIC16* sourceImage,
 EROIC16* destinationImage
 )

void Resize(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void Resize(
 EROIC24A* sourceImage,
 EROIC24A* destinationImage
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

destinationImage
Pointer to the destination image/ROI.

Easy::GetResourcesRootPath

Retrieves the resources installation path.

Namespace: Euresys::Open_eVision

```
[C++]
static std::string GetResourcesRootPath()
```

Easy::GetSampleImagesRootPath

Retrieves the sample images installation path.

Namespace: Euresys::Open_eVision

```
[C++]
static std::string GetSampleImagesRootPath()
```

Easy::GetSampleProgramsRootPath

Retrieves the sample programs installation path.

Namespace: Euresys::Open_eVision

```
[C++]
static std::string GetSampleProgramsRootPath()
```

Easy::SetOemKey

Writes an OEM key value on a dongle.

Namespace: Euresys::Open_eVision

[C++]

```
void SetOemKey(
 const std::vector<char>& key,
 Euresys::Open_eVision::EDongleType type,
 int dongleIndex
 )

void SetOemKey(
 OEV_UINT32 keyIndex,
 const std::vector<char>& key,
 Euresys::Open_eVision::EDongleType type,
 int dongleIndex
 )
```

Parameters

key

The OEM key value to write

type

The EDongleType for which you want to set the OEM key.

dongleIndex

The index of the dongle where the OEM key must be written. By default, the first dongle found is selected.

keyIndex

The index of the key in the array of keys, must be in [0, 11].
This option is not compatible with EDongleType_Legacy

Remarks

The length of the OEM key must be in [8, 64] characters. This method raises an EError_CannotWriteOEMKey error if the value cannot be set properly.

Easy::StartTiming

Starts timing, using the system clock or performance counter.

Namespace: Euresys::Open_eVision

[C++]

```
void StartTiming(
 )
```

Easy::StopTiming

Returns the time, in specified time units, elapsed since the last invocation of Easy::StartTiming.

Namespace: Euresys::Open_eVision

```
[C++]
int StopTiming(
 int resolution
 )
```

Parameters

resolution

Temporal resolution, in ticks per second.

Easy::Terminate

Method not obligatory, but necessary for close dll cleanly.

Namespace: Euresys::Open_eVision

```
[C++]
void Terminate(
 )
```

Easy::ToDegrees

Returns the angle, converted from current angle unit to degrees.

Namespace: Euresys::Open_eVision

```
[C++]
float ToDegrees(
 float angle
 )
```

Parameters

angle

Angle to be converted.

Easy::ToRadians

Returns the angle, converted from current angle unit to radians.

Namespace: Euresys::Open_eVision

```
[C++]
float ToRadians(
 float angle
 )
```

Parameters

angle

Angle to be converted.

Easy::TrueTimingResolution

Returns the actual resolution of the timing clock, in ticks per seconds.

Namespace: Euresys::Open_eVision

```
[C++]
int TrueTimingResolution(
 )
```

Remarks

Timing granularity is hardware-dependent, but is usually better than 1 microsecond. This function can be used to select an appropriate timing resolution when using Easy::StopTiming.

Easy::GetVersion

Returns a pointer to a NULL terminated character string that contains the current version number of Open eVision.

Namespace: Euresys::Open_eVision

```
[C++]
static std::string GetVersion()
```

4.22. EasyColor Class

This class contains static properties and methods specific to the EasyColor library.

Namespace: Euresys::Open_eVision

Methods

AlphaBlend	Draws an image over an other.
AssignNearestClass	Assigns to every pixel of the source image the nearest class index *plus one* and stores its value in the destination image.
AssignNearestClassCenter	Assigns to every pixel of the source image the nearest class center and stores its value in the destination image.
BayerToC24	Converts a Bayer pattern encoded image into a color image. Prefer the function EasyColor::BayerToC24 with EBayerConfiguration and mode parameters.

C24ToBayer	Converts a color image into a Bayer pattern encoded image. Deprecation notice: the version of this method taking two bool as argument is deprecated. You should use the one taking an EBayerConfiguration instead.
ClassAverages	Computes the average source pixel colors for every class separately.
ClassVariances	Computes the averages and variances of the image pixel colors for every class separately.
Compose	Combines three gray-level images, considered as three color planes, into a color image.
Decompose	Extracts the three color planes, considered as gray-level images, from a color image.
Dequantize	Convert a quantized value to a unquantized value of a given color system.
Format422To444	Converts a YUV 4:2:2 image to a YUV 4:4:4 image using interpolation.
Format444To422	Converts a YUV 4:4:4 image to a YUV 4:2:2 image using filtering
GetCieAB	CIE AB white illuminant
GetCieAG	CIE AG white illuminant
GetCieAR	CIE AR white illuminant
GetCieD50B	CIE D50B white illuminant
GetCieD50G	CIE D50G white illuminant
GetCieD50R	CIE D50R white illuminant
GetCieD55B	CIE D55B white illuminant
GetCieD55G	CIE D55G white illuminant
GetCieD55R	CIE D55R white illuminant
GetCieD65B	CIE D65B white illuminant
GetCieD65G	CIE D65G white illuminant
GetCieD65R	CIE D65R white illuminant
GetCieFB	CIE FB white illuminant
GetCieFG	CIE FG white illuminant
GetCieFR	CIE FR white illuminant
GetCompensateNtscGamma	NTSC inverse gamma exponent
GetCompensatePalGamma	PAL inverse gamma exponent
GetCompensateSmpteGamma	NTSC inverse gamma exponent
GetComponent	Extracts one color plane, considered as a gray-level image, from a color image.
GetDstQuantization	Quantization mode for output values.

GetNtscGamma	NTSC gamma exponent
GetPalGamma	PAL gamma exponent
GetRgbStandard	RGB definition to be used when converting between RGB and other color systems.
GetSmpteGamma	SMPTE gamma exponent
GetSrcQuantization	Quantization mode for input values.
ImproveClassCenters	Redefines the class centers by computing the average color value of the pixels assigned to each class in the source image.
IshToRgb	Convert a color from any system to RGB.
LabToRgb	Convert a color from any system to RGB.
LabToXyz	Convert a color from one system to another.
LchToRgb	Convert a color from any system to RGB.
LshToRgb	Convert a color from any system to RGB.
LuvToRgb	Convert a color from any system to RGB.
LuvToXyz	Convert a color from one system to another.
PseudoColor	Applies the mapping defined by the pseudo-color lookup table to transform a gray-level image into a color image.
Quantize	Convert an unquantized color of a given color system to a quantized color.
RegisterPlanes	Sets a color plane of a color image by using a gray-level image as component.
RgbToIsh	Convert a color from RGB to another system.
RgbToLab	Convert a color from RGB to another system.
RgbToLch	Convert a color from RGB to another system.
RgbToLsh	Convert a color from RGB to another system.
RgbToLuv	Convert a color from RGB to another system.
RgbToReducedXyz	Convert a color from one system to another.
RgbToVsh	Convert a color from RGB to another system.
RgbToXyz	Convert a color from RGB to another system.
RgbToYiq	Convert a color from RGB to another system.
RgbToYsh	Convert a color from RGB to another system.
RgbToYuv	Convert a color from RGB to another system.
SetComponent	Sets a color plane of a color image by using a gray-level image as component.
SetDstQuantization	Quantization mode for output values.
SetRgbStandard	RGB definition to be used when converting between RGB and other color systems.
SetSrcQuantization	Quantization mode for input values.

Transform	Applies a color transformation to a specified image.
TransformBayer	Converts an image, using the transformation defined by a color lookup. Deprecation notice: the version of this method taking two bool as argument is deprecated. You should use the one taking an EBayerConfiguration instead.
VshToRgb	Convert a color from any system to RGB.
XyzToLab	Convert a color from one system to another.
XyzToLuv	Convert a color from one system to another.
XyzToRgb	Convert a color from any system to RGB.
YiqToRgb	Convert a color from any system to RGB.
YshToRgb	Convert a color from any system to RGB.
YuvToRgb	Convert a color from any system to RGB.

EasyColor::AlphaBlend

Draws an image over an other.

Namespace: Euresys::Open_eVision

[C++]

```
void AlphaBlend(
 EROIC24& source,
 EROIC24& destination,
 double opacity
 )
```

Parameters

source

Foreground image.

destination

Background image.

opacity

Opacity of the foreground image.

EasyColor::AssignNearestClass

Assigns to every pixel of the source image the nearest class index *plus one* and stores its value in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void AssignNearestClass(
 EROIC24* sourceImage,
 EROIBW8* destinationImage,
 EC24Vector* classCenters
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination gray-level image/ROI.

classCenters

Pointer to the vector of the class centers.

Remarks

This generates a labeled gray-level image for use with EasyObject (see EImageEncoder and ELabeledImageSegmenter).

Note. The class index plus one is stored instead of the class index because EasyObject will never code class 0 objects.
Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center.
Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions.
To use the color segmentation functions, the set of class centers must be specified as a vector of EC24 elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI.
Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor::AssignNearestClassCenter

Assigns to every pixel of the source image the nearest class center and stores its value in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void AssignNearestClassCenter(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 EC24Vector* classCenters
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

classCenters

Pointer to the vector of the class centers.

Remarks

This generates a labeled color image.
Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center.
Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions.
To use the color segmentation functions, the set of class centers must be specified as a vector of EC24 elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI.
Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor::BayerToC24

Converts a Bayer pattern encoded image into a color image.
Prefer the function EasyColor::BayerToC24 with EBayerConfiguration and mode parameters.

Namespace: Euresys::Open_eVision

[C++]

```
void BayerToC24(
 EROIBW8* sourceImage,
 EROIC24* destinationImage,
 bool evenCol,
 bool evenRow,
 bool interpolate,
 bool improved
 )

void BayerToC24(
 EROIBW8* sourceImage,
 EROIC24* destinationImage,
 Euresys::Open_eVision::EBayerConfiguration bayerConfiguration,
 int mode
 )
```

Parameters

sourceImage

Pointer to the Bayer pattern input image/ROI, stored using the 8 bits per pixel format.

destinationImage

Pointer to the color output image/ROI.

evenCol

true if the leftmost image column contains no blue pixels.

evenRow

true if the topmost image row contains no red pixels.

interpolate

Interpolation mode to be used for pixel reconstruction. When false, the missing color components are merely copied from northern/western pixels; when true, they are computed by averaging from relevant neighbors. By default, interpolation is used.

improved

Provides an access to an improved interpolation mode that reduces visible artifacts along object edges. The running time of the improved method is longer. By default, it is not used.

bayerConfiguration

The color configuration of the bayer image. The color configuration is defined by the component of the first 2 pixels of the image, see EBayerConfiguration.

mode

Interpolation mode to be used for RGB pixel reconstruction, from 0 to 4 (increasing quality). By default, interpolation mode 1 is used.
- mode 0: No interpolation (fastest)
- mode 1: Linear interpolation on 3x3 kernel
- mode 2: Advanced interpolation on 3x3 kernel
- mode 3: Interpolation on 5x5 kernel
- mode 4: Interpolation on 9x9 kernel (slowest)
- mode 5: Linear interpolation on 2x2 kernel

Remarks

The pattern can be shifted by one pixel horizontally and vertically as needed to deal with a non standard pattern origin.
See also Bayer Filter.

EasyColor::C24ToBayer

Converts a color image into a Bayer pattern encoded image. Deprecation notice: the version of this method taking two bool as argument is deprecated. You should use the one taking an EBayerConfiguration instead.

Namespace: Euresys::Open_eVision

[C++]

```
void C24ToBayer(
 EROIC24* sourceImage,
 EROIBW8* destinationImage,
 Euresys::Open_eVision::EBayerConfiguration bayerConfiguration
 )
```

```
void C24ToBayer(
 EROIC24* sourceImage,
 EROIBW8* destinationImage,
 bool evenCol,
 bool evenRow
 )
```

Parameters

sourceImage

Pointer to the color input image/ROI.

destinationImage

Pointer to the Bayer pattern output image/ROI, stored using the 8 bits per pixel format.

bayerConfiguration

The color configuration of the bayer image. The color configuration is defined by the component of the first 2 pixels of the image, see EBayerConfiguration.

evenCol

true if the leftmost destination image column can't contain blue pixels.

evenRow

true if the topmost destination image row can't contain red pixels.

Remarks

The pattern can be shifted by one pixel horizontally and vertically as needed to deal with a non standard pattern origin.
See also Bayer Filter.

EasyColor::GetCieAB

CIE AB white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieAB()
```

EasyColor::GetCieAG

CIE AG white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieAG()
```

EasyColor::GetCieAR

CIE AR white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieAR()
```

EasyColor::GetCieD50B

CIE D50B white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD50B()
```

EasyColor::GetCieD50G

CIE D50G white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD50G()
```

EasyColor::GetCieD50R

CIE D50R white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD50R()
```

EasyColor::GetCieD55B

CIE D55B white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD55B()
```

EasyColor::GetCieD55G

CIE D55G white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD55G()
```

EasyColor::GetCieD55R

CIE D55R white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD55R()
```

EasyColor::GetCieD65B

CIE D65B white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD65B()
```

EasyColor::GetCieD65G

CIE D65G white illuminant

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCieD65G()
```

EasyColor::GetCieD65R

CIE D65R white illuminant

Namespace: Euresys::Open_eVision

```
[C++]
static float GetCieD65R()
```

EasyColor::GetCieFB

CIE FB white illuminant

Namespace: Euresys::Open_eVision

```
[C++]
static float GetCieFB()
```

EasyColor::GetCieFG

CIE FG white illuminant

Namespace: Euresys::Open_eVision

```
[C++]
static float GetCieFG()
```

EasyColor::GetCieFR

CIE FR white illuminant

Namespace: Euresys::Open_eVision

```
[C++]
static float GetCieFR()
```

EasyColor::ClassAverages

Computes the average source pixel colors for every class separately.

Namespace: Euresys::Open_eVision

```
[C++]
void ClassAverages(
 EROIC24* sourceImage,
 EC24Vector* classCenters,
 EColorVector* averages
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

classCenters

Pointer to the vector of the class centers.

averages

Pointer to the vector of the average color values.

Remarks

This allows measuring the actual average color of the segmented regions.
Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center.
Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions.
To the color segmentation functions, the set of class centers must be specified as a vector of EC24 elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI.
Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor::ClassVariances

Computes the averages and variances of the image pixel colors for every class separately.

Namespace: Euresys::Open_eVision

[C++]

```
void ClassVariances(
 EROIC24* sourceImage,
 EC24Vector* classCenters,
 EColorVector* averages,
 EColorVector* variances
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

classCenters

Pointer to the vector of the class centers.

averages

Pointer to the vector of the average color values.

variances

Pointer to the vector of the variance color values.

Remarks

This allows quantifying the homogeneity of the segmented regions.
Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center. Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions.
To the color segmentation functions, the set of class centers must be specified as a vector of EC24 elements. In this sense, the method is termed supervised clustering. A good way to obtain these values is to compute the average color in an ROI.
Unsupervised clustering is also made available by implementing the so called K-means method that automatically improves an initial choice of class centers.

EasyColor::GetCompensateNtscGamma

NTSC inverse gamma exponent

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCompensateNtscGamma()
```

EasyColor::GetCompensatePalGamma

PAL inverse gamma exponent

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCompensatePalGamma()
```

EasyColor::GetCompensateSmpteGamma

NTSC inverse gamma exponent

Namespace: Euresys::Open_eVision

[C++]

```
static float GetCompensateSmpteGamma()
```

EasyColor::Compose

Combines three gray-level images, considered as three color planes, into a color image.

Namespace: Euresys::Open_eVision

[C++]

```
void Compose(
 EROIBW8* sourceImageOfColor0,
 EROIBW8* sourceImageOfColor1,
 EROIBW8* sourceImageOfColor2,
 EROIC24* colorDestinationImage,
 EColorLookup* lookup
 )
void Compose(
 EROIBW16* sourceImageOfColor0,
 EROIBW16* sourceImageOfColor1,
 EROIBW16* sourceImageOfColor2,
 EROIC48* colorDestinationImage
 )
```

Parameters

sourceImageOfColor0

Pointers to the three input gray-level component images/ROIs.

sourceImageOfColor1

Pointers to the three input gray-level component images/ROIs.

sourceImageOfColor2

Pointers to the three input gray-level component images/ROIs.

colorDestinationImage

Pointer to the output color image/ROI.

lookup

Pointer to the color lookup table, or NULL.

Remarks

If a color lookup is used, the resulting image undergoes the corresponding color transform. This way, it is possible to build an RGB image from the color planes of another system, or conversely.

A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An EImageC24 contains three EImageBW8.

EasyColor::Decompose

Extracts the three color planes, considered as gray-level images, from a color image.

Namespace: Euresys::Open_eVision

```
[C++]
void Decompose(
 EROIC24* colorSourceImage,
 EROIBW8* destinationImageOfColor0,
 EROIBW8* destinationImageOfColor1,
 EROIBW8* destinationImageOfColor2,
 EColorLookup* lookup
 )
void Decompose(
 EROIC48* colorSourceImage,
 EROIBW16* destinationImageOfColor0,
 EROIBW16* destinationImageOfColor1,
 EROIBW16* destinationImageOfColor2
 )
```

Parameters

colorSourceImage

Pointer to the input color image/ROI.

destinationImageOfColor0

Pointers to the three output gray level component images/ROIs.

destinationImageOfColor1

Pointers to the three output gray level component images/ROIs.

destinationImageOfColor2

Pointers to the three output gray level component images/ROIs.

lookup

Pointer to the color lookup table, or NULL.

Remarks

If a color lookup is used, the source image undergoes the corresponding color transform. This way, it is possible to get the RGB components from an image of another system, of conversely.
A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An EImageC24 contains three EImageBW8.

EasyColor::Dequantize

Convert a quantized value to a unquantized value of a given color system.

Namespace: Euresys::Open_eVision

[C++]

```
void Dequantize(
 EC24 colorIn,
 ERGB& colorOut
 )

void Dequantize(
 EC24 colorIn,
 EXYZ& colorOut
 )

void Dequantize(
 EC24 colorIn,
 EYUV& colorOut
 )

void Dequantize(
 EC24 colorIn,
 EYIQ& colorOut
 )

void Dequantize(
 EC24 colorIn,
 ELSH& colorOut
 )

void Dequantize(
 EC24 colorIn,
 EVSH& colorOut
 )

void Dequantize(
 EC24 colorIn,
 EISH& colorOut
 )

void Dequantize(
 EC24 colorIn,
 EYSH& colorOut
 )

void Dequantize(
 EC24 colorIn,
 ELAB& colorOut
 )

void Dequantize(
 EC24 colorIn,
 ELCH& colorOut
 )

void Dequantize(
 EC24 colorIn,
 ELUV& colorOut
 )
```

Parameters

colorIn

Input quantized color.

colorOut

Output unquantized color, as defined by the corresponding structure.

Remarks

Quantization is the process that transforms a continuous value, usually represented as a floating-point value in the [0..1] interval, into a discrete one, usually represented as an integer in the [0..255] interval.
Dequantization is the reverse process. For RGB color system, it undoes the gamma-correction corresponding to the current ERgbStandard.

EasyColor::GetDstQuantization

EasyColor::SetDstQuantization

Quantization mode for output values.

Namespace: Euresys::Open_eVision

[C++]

```
static Euresys::Open_eVision::EColorQuantization GetDstQuantization()

static void SetDstQuantization(Euresys::Open_eVision::EColorQuantization quantization)
```

Remarks

These settings remain permanent and influence the relevant quantized and unquantized color conversions (during lookup table initialization or image color transformation).
Quantization modes are set for the current thread.

EasyColor::Format422To444

Converts a YUV 4:2:2 image to a YUV 4:4:4 image using interpolation.

Namespace: Euresys::Open_eVision

[C++]

```
void Format422To444(
 EROIBW16* sourceImage,
 EROIC24* destinationImage,
 bool yFirst
 )
```

Parameters

sourceImage

Pointer to the input image/ROI, stored using the 16 bits per pixel format.

destinationImage

Pointer to the output image/ROI.

yFirst

Flag indicating if the format is YUYVYUYV (true) or UYVYUYVY (false).

Remarks

In the YUV system, it has been established that sub-sampling the chroma components does not degrade the visual image quality. The 4:4:4 format uses three bytes of information by pixel. The 4:2:2 format is such that only the U and V chroma of the even pixels are kept and they are stored with the even and odd luma, as follows: Thus, only two bytes per pixel are required. $Y_{[even]}$ $U_{[even]}$ $Y_{[odd]}$ $V_{[even]}$

EasyColor::Format444To422

Converts a YUV 4:4:4 image to a YUV 4:2:2 image using filtering

Namespace: Euresys::Open_eVision

[C++]

```
void Format444To422(
 EROIC24* sourceImage,
 EROIBW16* destinationImage,
 bool yFirst
 )
```

Parameters

sourceImage

Pointer to the input image/ROI.

destinationImage

Pointer to the output image/ROI, stored using the 16 bits per pixel format.

yFirst

Flag indicating if the format is YUYVYUYV (true) or UYVYUYVY (false).

Remarks

In the YUV system, it has been established that sub-sampling the chroma components does not degrade the visual image quality. The 4:4:4 format uses three bytes of information by pixel. The 4:2:2 format is such that only the U and V chroma of the even pixels are kept and they are stored with the even and odd luma, as follows: Thus, only two bytes per pixel are required. $Y_{[even]}$ $U_{[even]}$ $Y_{[odd]}$ $V_{[even]}$

EasyColor::GetComponent

Extracts one color plane, considered as a gray-level image, from a color image.

Namespace: Euresys::Open_eVision

[C++]

```
void GetComponent(
 const EROIC24* colorSourceImage,
 EROIBW8* bWDestinationImage,
 OEV_UINT32 component,
 EColorLookup* lookup
 )
void GetComponent(
 EROIC48* colorSourceImage,
 EROIBW16* bWDestinationImage,
 OEV_UINT32 component
 )
```

Parameters

colorSourceImage

Pointer to the input color image/ROI.

bWDestinationImage

Pointers to the output gray-level component image/ROI.

component

Color component index (0, 1, or 2).

lookup

Pointer to the color lookup table, or NULL.

EasyColor::ImproveClassCenters

Redefines the class centers by computing the average color value of the pixels assigned to each class in the source image.

Namespace: Euresys::Open_eVision

[C++]

```
void ImproveClassCenters(
 EROIC24* sourceImage,
 EC24Vector* classCenters
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

classCenters

Pointer to the vector of the class centers.

Remarks

This implements a step of the K-means method for unsupervised clustering.
Color image segmentation allows you to decompose a color image in different regions by assigning a "class" (integer index) to every pixel. The nearest neighbor method is used, i.e. for each class a representative center is specified, and a given pixel is associated to the class with the closest center.
Color image segmentation can be used in conjunction with EasyObject to perform blob analysis on the segmented regions.

EasyColor::IshToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void IshToRgb(
 EISH colorIn,
 ERGB& colorOut
 )
void IshToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::LabToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void LabToRgb(
 ELAB colorIn,
 ERGB& colorOut
 )
void LabToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::LabToXyz

Convert a color from one system to another.

Namespace: Euresys::Open_eVision

[C++]

```
void LabToXyz(
 ELAB colorIn,
 EXYZ& colorOut
 )
void LabToXyz(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::LchToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void LchToRgb(
 ELCH colorIn,
 ERGB& colorOut
 )
void LchToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::LshToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void LshToRgb(
 ELSH colorIn,
 ERGB& colorOut
 )
void LshToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::LuvToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void LuvToRgb(
 ELUV colorIn,
 ERGB& colorOut
 )
void LuvToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::LuvToXyz

Convert a color from one system to another.

Namespace: Euresys::Open_eVision

[C++]

```
void LuvToXyz(
 ELUV colorIn,
 EXYZ& colorOut
 )
void LuvToXyz(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::GetNtscGamma

NTSC gamma exponent

Namespace: Euresys::Open_eVision

[C++]

```
static float GetNtscGamma()
```

EasyColor::GetPalGamma

PAL gamma exponent

Namespace: Euresys::Open_eVision

[C++]

```
static float GetPalGamma()
```

EasyColor::PseudoColor

Applies the mapping defined by the pseudo-color lookup table to transform a gray-level image into a color image.

Namespace: Euresys::Open_eVision

[C++]

```
void PseudoColor(
 EROIBW8* sourceImage,
 EROIC24* destinationImage,
 EPseudoColorLookup* lookup
 )
```

Parameters

sourceImage

Pointer to the source gray-level image.

destinationImage

Pointer to the destination color image.

lookup

Pointer to the pseudo-color lookup table.

Remarks

Pseudo-coloring is a convenient way to display gray-level images with enhanced contrast: a shade of colors is associated to the shade of gray-level values. A simple way to define the shade of colors is to specify a path in color space.
In order to use pseudo-coloring, a special lookup table is used: EPseudoColorLookup. It handles the mapping between the gray-level and color values.

EasyColor::Quantize

Convert an unquantized color of a given color system to a quantized color.

Namespace: Euresys::Open_eVision

[C++]

```
void Quantize(
 ERGB colorIn,
 EC24& colorOut
 )

void Quantize(
 EXYZ colorIn,
 EC24& colorOut
 )

void Quantize(
 EYUV colorIn,
 EC24& colorOut
 )

void Quantize(
 EYIQ colorIn,
 EC24& colorOut
 )
```

```
void Quantize(
 ELSH colorIn,
 EC24& colorOut
 )
void Quantize(
 EVSH colorIn,
 EC24& colorOut
 )
void Quantize(
 EISH colorIn,
 EC24& colorOut
 )
void Quantize(
 EYSH colorIn,
 EC24& colorOut
 )
void Quantize(
 ELAB colorIn,
 EC24& colorOut
 )
void Quantize(
 ELCH colorIn,
 EC24& colorOut
 )
void Quantize(
 ELUV colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input unquantized color, as defined by the corresponding structure.

colorOut

Output quantized color.

Remarks

Quantization is the process that transforms a continuous value, usually represented as a floating-point value in the [0..1] interval, into a discrete one, usually represented as an integer in the [0..255] interval.
Dequantization is the reverse process. For RGB color system, it applies gamma-correction corresponding to the current ERgbStandard.

EasyColor::RegisterPlanes

Sets a color plane of a color image by using a gray-level image as component.

Namespace: Euresys::Open_eVision

```
[C++]
void RegisterPlanes(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 int rShiftX,
 int gShiftX,
 int bShiftX,
 int rShiftY,
 int gShiftY,
 int bShiftY
 )
```

Parameters

sourceImage

Pointers to the input image/ROI.

destinationImage

Pointer to the output image/ROI.

rShiftX

Horizontal shifting of the first plane (the red one in case of an RGB image), expressed in pixels.

gShiftX

Horizontal shifting of the second plane (the green one in case of an RGB image), expressed in pixels.

bShiftX

Horizontal shifting of the third plane (the blue one in case of an RGB image), expressed in pixels.

rShiftY

Vertical shifting of the first plane (the red one in case of an RGB image), expressed in pixels.

gShiftY

Vertical shifting of the second plane (the green one in case of an RGB image), expressed in pixels.

bShiftY

Vertical shifting of the third plane (the blue one in case of an RGB image), expressed in pixels.

Remarks

A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An EImageC24 contains three EImageBW8.

EasyColor::GetRgbStandard

EasyColor::SetRgbStandard

RGB definition to be used when converting between RGB and other color systems.

Namespace: Euresys::Open_eVision

[C++]

```
static Euresys::Open_eVision::ERgbStandard GetRgbStandard()
static void SetRgbStandard(Euresys::Open_eVision::ERgbStandard rgbStandard)
```

Remarks

Some variant of the color systems can be used. The EasyColor::SrcQuantization and EasyColor::DstQuantization functions are used to activate them.
These settings remain permanent and influence the relevant quantized and unquantized color conversions (during lookup table initialization or image color transformation).
RgbStandard is set for the current thread.

EasyColor::RgbToIsh

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToIsh(
 ERGB colorIn,
 EISH& colorOut
 )
void RgbToIsh(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToLab

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToLab(
 ERGB colorIn,
 ELAB& colorOut
 )
void RgbToLab(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToLch

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToLch(
 ERGB colorIn,
 ELCH& colorOut
 )
void RgbToLch(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToLsh

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

```
[C++]
void RgbToLsh(
 ERGB colorIn,
 ELSH& colorOut
 )
void RgbToLsh(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToLuv

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

```
[C++]
void RgbToLuv(
 ERGB colorIn,
 ELUV& colorOut
 )
void RgbToLuv(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToReducedXyz

Convert a color from one system to another.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToReducedXyz(
 ERGB colorIn,
 EXYZ& colorOut
 )
void RgbToReducedXyz(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToVsh

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToVsh(
 ERGB colorIn,
 EVSH& colorOut
 )
void RgbToVsh(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToXyz

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToXyz(
 ERGB colorIn,
 EXYZ& colorOut
 )
void RgbToXyz(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToYiq

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

```
[C++]
void RgbToYiq(
 ERGB colorIn,
 EYIQ& colorOut
 )
void RgbToYiq(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToYsh

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

```
[C++]
void RgbToYsh(
 ERGB colorIn,
 EYSH& colorOut
 )
void RgbToYsh(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::RgbToYuv

Convert a color from RGB to another system.

Namespace: Euresys::Open_eVision

[C++]

```
void RgbToYuv(
 ERGB colorIn,
 EYUV& colorOut
 )
void RgbToYuv(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color, as defined by the corresponding structure.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::SetComponent

Sets a color plane of a color image by using a gray-level image as component.

Namespace: Euresys::Open_eVision

[C++]

```
void SetComponent(
 EROIBW8* bWSourceImage,
 EROIC24* colorDestinationImage,
 OEV_UINT32 component
 )

void SetComponent(
 EROIBW16* bWSourceImage,
 EROIC48* colorDestinationImage,
 OEV_UINT32 component
 )
```

Parameters

bWSourceImage

Pointers to the input gray level component image/ROI.

colorDestinationImage

Pointer to the output color image/ROI.

component

Color component index (0, 1, or 2).

Remarks

A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An EImageC24 contains three EImageBW8.

EasyColor::GetSmpteGamma

SMPTE gamma exponent

Namespace: Euresys::Open_eVision

[C++]

```
static float GetSmpteGamma()
```

EasyColor::GetSrcQuantization

EasyColor::SetSrcQuantization

Quantization mode for input values.

Namespace: Euresys::Open_eVision

[C++]

```
static Euresys::Open_eVision::EColorQuantization GetSrcQuantization()

static void SetSrcQuantization(Euresys::Open_eVision::EColorQuantization quantization)
```

Remarks

These settings remain permanent and influence the relevant quantized and unquantized color conversions (during lookup table initialization or image color transformation). Quantization modes are set for the current thread.

EasyColor::Transform

Applies a color transformation to a specified image.

Namespace: Euresys::Open_eVision

[C++]

```
void Transform(
 ERGBIC24* sourceImage,
 ERGBIC24* destinationImage,
 EColorLookup* lookup
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lookup

Pointer to the color lookup.

Remarks

In the first case, the transformation is defined by a color lookup. See Initialization (EColorLookup).
In the two other cases, the user defines a quantized or unquantized color transformation. No intermediate color lookup table is used.
A color image can be seen as a set of three color planes, each corresponding to a color component. The color planes are themselves continuous tone images. An EImageC24 contains three EImageBW8.

EasyColor::TransformBayer

Converts an image, using the transformation defined by a color lookup. Deprecation notice: the version of this method taking two bool as argument is deprecated. You should use the one taking an EBayerConfiguration instead.

Namespace: Euresys::Open_eVision

[C++]

```
void TransformBayer(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EColorLookup* lookup,
 Euresys::Open_eVision::EBayerConfiguration bayerConfiguration
 )
void TransformBayer(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EColorLookup* lookup,
 bool evenCol,
 bool evenRow
 )
```

Parameters

sourceImage

Pointer to the source image/ROI. This image must be encoded using the Bayer color pattern.

destinationImage

Pointer to the destination image/ROI. This image must be encoded using the Bayer color pattern.

lookup

Pointer to the color lookup table holding the color adjustment transform. The lookup table must be previously set up by EColorLookup::WhiteBalance method (no other transforms are supported).

bayerConfiguration

The color configuration of the bayer image. The color configuration is defined by the component of the first 2 pixels of the image, see EBayerConfiguration.

evenCol

true if the leftmost destination image column can't contain blue pixels.

evenRow

true if the topmost destination image row can't contain red pixels.

Remarks

By contrast with EasyColor::Transform, the transformation is applied directly to Bayer-encoded data. This allows efficient processing to take place before conversion to the C24 format.

EasyColor::VshToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void VshToRgb(
 EVSH colorIn,
 ERGB& colorOut
 )
void VshToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::XyzToLab

Convert a color from one system to another.

Namespace: Euresys::Open_eVision

[C++]

```
void XyzToLab(
 EXYZ colorIn,
 ELAB& colorOut
 )
void XyzToLab(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::XyzToLuv

Convert a color from one system to another.

Namespace: Euresys::Open_eVision

```
[C++]
void XyzToLuv(
 EXYZ colorIn,
 ELUV& colorOut
 )
void XyzToLuv(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::XyzToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

```
[C++]
void XyzToRgb(
 EXYZ colorIn,
 ERGB& colorOut
 )
void XyzToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::YiqToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void YiqToRgb(
 EYIQ colorIn,
 ERGB& colorOut
 )
void YiqToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::YshToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void YshToRgb(
 EYSH colorIn,
 ERGB& colorOut
 )
void YshToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

EasyColor::YuvToRgb

Convert a color from any system to RGB.

Namespace: Euresys::Open_eVision

[C++]

```
void YuvToRgb(
 EYUV colorIn,
 ERGB& colorOut
 )
void YuvToRgb(
 EC24 colorIn,
 EC24& colorOut
 )
```

Parameters

colorIn

Input color.

colorOut

Output color.

Remarks

These functions transform the color components (of a pixel) expressed in some color system to the corresponding components in another system. These functions take into account the current ERgbStandard and the associated white point if necessary.

4.23. EasyImage Class

This class contains static properties and methods specific to the EasyImage library.

Namespace: Euresys::Open_eVision

Methods

AdaptiveThreshold	Performs a locally adaptive threshold on the source image.
AlphaBlend	Draws an image over an other.
AnalyseHistogram	Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).
AnalyseHistogramBW16	Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).
Area	Counts the pixels whose values are above (or on) a threshold.
AreaDoubleThreshold	Counts the pixels whose values are comprised between (or on) two thresholds.
ArgumentImage	Prepares a lookup-table image for use for gradient argument computation.
AutoThreshold	Returns a suitable threshold value for a gray-level image binarization.
BiLevelBlackTopHatBox	Performs a top-hat filtering on a bilevel image (closed image minus source image) on a rectangular kernel.
BiLevelBlackTopHatDisk	Performs a top-hat filtering on a bilevel image (closed image minus source image) on a quasi-circular kernel.
BiLevelCloseBox	Performs a closing on a bilevel image (dilation followed by erosion) on a rectangular kernel.
BiLevelCloseDisk	Performs a closing on a bilevel image (dilation followed by erosion) on a quasi-circular kernel.
BiLevelDilateBox	Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a rectangular kernel. For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.
BiLevelDilateDisk	Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a quasi-circular kernel. For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.
BiLevelErodeBox	Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a rectangular kernel. For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

BiLevelErodeDisk	Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a quasi-circular kernel. For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)
BiLevelMedian	Applies a median filter to a bilevel image (median of the gray values in a 3x3 neighborhood).
BiLevelMorphoGradientBox	Computes the morphological gradient of a bilevel image using a rectangular kernel.
BiLevelMorphoGradientDisk	Computes the morphological gradient of a bilevel image using a quasi-circular kernel.
BiLevelOpenBox	Performs an opening on a bilevel image (erosion followed by dilation) on a rectangular kernel.
BiLevelOpenDisk	Performs an opening on a bilevel image (erosion followed by dilation) on a quasi-circular kernel.
BiLevelThick	Applies a thickening operation on a bilevel image, using a 3x3 kernel.
BiLevelThin	Applies a thinning operation on a bilevel image, using a 3x3 kernel.
BiLevelWhiteTopHatBox	Performs a top-hat filtering on a bilevel image (source image minus open image) on a rectangular kernel.
BiLevelWhiteTopHatDisk	Performs a top-hat filtering on a bilevel image (source image minus open image) on a quasi-circular kernel.
BinaryMoments	Computes the zero-th, first or second order moments on the binarized image, i.e. with a unit weight for those pixels with a value above or equal to the threshold, and zero otherwise.
BlackTopHatBox	Performs a top-hat filtering on an image (closed image minus source image) on a rectangular kernel.
BlackTopHatDisk	Performs a top-hat filtering on an image (closed image minus source image) on a circular kernel.
CloseBox	Performs a closing on an image (dilation followed by erosion) on a rectangular kernel.
CloseDisk	Performs a closing on an image (dilation followed by erosion) on a circular kernel.
Contour	Follows the *contour* of an object.
Convert	Transforms the contents of an image to an image of another type.
ConvertTo422	Turns an 8-bit gray-level image into a YUV 4:2:2 encoded color image.
ConvolGabor	Computes the Gabor filter response of the source image and stores the result in the destination image.
ConvolGaussian	Applies Gaussian filtering (binomial weights) in rectangular kernel of odd size.
ConvolGradient	Extracts the edges of an image by summing the absolute values of the Gradient X and Gradient Y derivatives and stores the absolute value (magnitude) of the result in the destination image.
ConvolGradientX	Derives an image along X using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.

ConvolGradientY	Derives an image along Y using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolHighpass1	Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolHighpass2	Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolKernel	Performs a convolution in image space, i.e. applies a convolution kernel.
ConvolLaplacian4	Takes the second derivative of an image using a 4-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolLaplacian8	Takes the second derivative of an image using an 8-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolLaplacianX	Takes the horizontal second derivative and stores the absolute value (magnitude) of the result in the destination image.
ConvolLaplacianY	Takes the vertical second derivative and stores the absolute value (magnitude) of the result in the destination image.
ConvolLowpass1	Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolLowpass2	Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolLowpass3	Filters an image using a 3x3 low-pass kernel.
ConvolPrewitt	Extracts the edges of an image by summing the absolute values of the Prewitt X and Prewitt Y derivatives and stores the absolute value (magnitude) of the result in the destination image.
ConvolPrewittX	Derives an image along X using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolPrewittY	Derives an image along Y using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolRoberts	The Roberts edge extraction filter is based on a 2x2 kernel.
ConvolSobel	Extracts the edges of an image by summing the absolute values of the Sobel X and Sobel Y derivatives.
ConvolSobelX	Derives an image along X using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolSobelY	Derives an image along Y using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.
ConvolSymmetricKernel	Performs a convolution in image space, i.e. applies a square symmetric convolution kernel of size 3x3, 5x5 or 7x7.
ConvolUniform	Applies strong low-pass filtering to an image by using a uniform rectangular kernel of odd size.
Copy	Copies a source image or a constant in a destination image.
CumulateHistogram	Cumulates histogram values in another histogram.

DilateBox	Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a rectangular kernel.
DilateDisk	Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a circular kernel.
Distance	Computes the morphological distance function on a binary image (0 for black, non 0 for white).
DoubleThreshold	Converts an image by setting all pixels below the low threshold to a low value, all pixels above the high threshold to a high value, and the remaining pixels to an intermediate value.
Equalize	Equalizes an image histogram (the gray levels are re-mapped so that the histogram becomes as close to uniform as possible).
ErodeBox	Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a rectangular kernel.
ErodeDisk	Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a circular kernel.
Flip	Flip an image around either the vertical axis, the horizontal axis or both.
Focusing	Returns a measure of the focusing of an image by computing the total gradient energy.
Gain	Transforms an image, applying a gain and offset to all pixels.
GainOffset	Transforms an image, applying a gain and offset to all pixels.
GetFrame	Extracts the frame of given parity from an image.
GetOverlayColor	Gets/Sets the color of the overlay in the destination image when a BW8 Image is used as overlay source image in functions.
GetProfilePeaks	Detects peaks in a gray-level profile. Maxima as well as minima are considered.
GradientScalar	Computes the (scalar) gradient image derived from a given source image.
GravityCenter	Computes the coordinates of the gravity center of an image, i.e. the average coordinates of the pixels above (or on) the threshold.
HDRFusion	Fuses two images using HDR principles.
Histogram	Computes the histogram of an image (count of each gray-level value).
HistogramThreshold	Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.
HistogramThresholdBW16	Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.
HitAndMiss	Apply the morphological hit-and-miss transform to detect a particular pattern of foreground and background pixels in a BW8, BW16 or C24 image/ROI.
HorizontalMirror	Mirrors an image horizontally (the columns are swapped).
ImageToLineSegment	Copies the pixel values along a given line segment (arbitrarily oriented) to a vector.

ImageToPath	Given a path described by coordinates in a path vector, copies the corresponding pixel values into the same vector.
IsodataThreshold	Computes a suitable threshold value for a histogram.
IsodataThresholdBW16	Computes a suitable threshold value for a histogram.
LinearTransform	Applies a general affine transformation.
LineSegmentToImage	Copies the pixel values from a vector or a constant to the pixels of a given line segment (arbitrarily oriented).
LocalAverage	Computes the average in a rectangular window centered on every pixel.
LocalDeviation	Computes the standard deviation in a rectangular window centered on every pixel.
Lut	Transforms the gray levels of an image, using a lookup table stored in a vector (of unsigned values).
MatchFrames	Determines the optimal shift amplitude by comparing two successive lines of the image.
Median	Applies a median filter to an image (median of the gray values in a neighborhood). Kernel may be of an arbitrary size except for EROIBW1 where it is always 3*3.
ModulusImage	Prepares a lookup-table image for use for gradient magnitude computation.
MorphoGradientBox	Computes the morphological gradient of an image using a rectangular kernel.
MorphoGradientDisk	Computes the morphological gradient of an image using a circular kernel.
Normalize	Normalizes an image, i.e. applies a linear transform to the gray levels so that their average and standard deviation are imposed.
Offset	Transforms an image, applying a gain and offset to all pixels.
OpenBox	Performs an opening on an image (erosion followed by dilation) on a rectangular kernel.
OpenDisk	Performs an opening on an image (erosion followed by dilation) on a circular kernel.
Oper	Applies the desired arithmetic or logic pixel-wise operator between two images or constants.
Overlay	Overlays an image on the top of a color image, at a given position.
PathToImage	Given a path described by coordinates in a path vector, copies the pixel values from the path vector to the corresponding image pixels.
PixelAverage	Computes the average pixel value in a gray-level or color image.
PixelCompare	Counts the number of pixels differing between two images.
PixelCount	Counts the pixels in the three value classes separated by two thresholds.
PixelMax	Computes the maximum gray-level value in an image.

PixelMaxBW16	Computes the maximum gray-level value in an image.
PixelMaxBW8	Computes the maximum gray-level value in an image.
PixelMin	Computes the minimum gray-level value in an image.
PixelMinBW16	Computes the minimum gray-level value in an image.
PixelMinBW8	Computes the minimum gray-level value in an image.
PixelStat	Computes the minimum, maximum and average gray-level values in an image.
PixelStatBW16	Computes the minimum, maximum and average gray-level values in an image.
PixelStatBW8	Computes the minimum, maximum and average gray-level values in an image.
PixelStdDev	Computes the average gray-level or color value in an image, the standard deviation of the color components, and the correlation between the color components (in the case of color images).
PixelVariance	For a gray-level image, computes the mean and variance of the pixel values.
ProfileDerivative	Computes the first derivative of a profile extracted from a gray-level image.
ProjectOnAColumn	Projects an image horizontally onto a column.
ProjectOnARow	Projects an image vertically onto a row.
RealignFrame	Shifts one frame of the image horizontally.
RebuildFrame	Rebuilds one frame of the image by interpolation between the lines of the other frame.
RecursiveAverage	Applies stronger noise reduction to small variations and conversely.
Register	Registers an image by realigning one, two or three pivot points to reference positions.
RmsNoise	Computes the root-mean-square amplitude of noise, by comparing a given image to a reference image.
Rotate	Rotate an image by an increment of a quarter of a turn (right angle).
ScaleRotate	Re-scales an image by an arbitrary factor and/or rotates it by an arbitrary angle.
SetCircleWarp	Prepares suitable warp images for use with function EasyImage::Warp to unwarp a circular ring-wedge shape into a straight rectangle. This is a cartesian to polar image conversion function. See also EasyImage::SetInvCircleWarp.
SetFrame	Replaces the frame of given parity in an image.
SetInvCircleWarp	Prepares suitable warp images for use with function EasyImage::Warp to unwarp a straight rectangle into a circular ring-wedge shape. This is a polar to cartesian image conversion function. See also EasyImage::SetCircleWarp.
SetOverlayColor	Gets/Sets the color of the overlay in the destination image when a BW8 Image is used as overlay source image in functions.

SetRecursiveAverageLUT	Pre-compute the required non-linear transfer function for noise reduction by recursive temporal averaging.
SetupEqualize	Prepares a lookup-table for image equalization, using an image histogram.
SetupInverseWarp	Prepares suitable inverse warp images for use with function EasyImage::Warp to unwarp an invertible LUT given by the warpImageX and warpImageY.
Shrink	Resizes an image to a smaller size. Pre-filtering is applied to avoid aliasing.
SignalNoiseRatio	Computes the signal to noise ratio, in dB, by comparing a given image to a reference image.
SwapFrames	Interchanges the even and odd rows of an image.
Thick	Applies a thickening operation on an image, using a 3x3 kernel.
Thin	Applies a thinning operation on an image, using a 3x3 kernel.
ThreeLevelsMinResidueThreshold	Computes the two threshold values used to separate the pixels of an image in three classes.
Threshold	Binarize an image by setting pixels to two different possible values in a destination image, according to their value in a source image.
Transpose	Transpose an image.
TwoLevelsMinResidueThreshold	Computes the threshold value used to separate the pixels of an image in two classes.
Uniformize	Shading correction is the process of transforming the gray or color component values of an image, using one or two reference images or vectors.
VerticalMirror	Mirrors an image vertically (the rows are swapped).
Warp	Transforms an image so that each pixels are moved to the locations specified in the "warp" images used as look-up tables.
WeightedMoments	Computes the zero-th, first, second, third or fourth order weighted moments on the gray-level image. The weight of a pixel is its gray-level value.
WhiteTopHatBox	Performs a top-hat filtering on an image (source image minus open image) on a rectangular kernel.
WhiteTopHatDisk	Performs a top-hat filtering on an image (source image minus open image) on a circular kernel.

EasyImage::AdaptiveThreshold

Performs a locally adaptive threshold on the source image.

Namespace: Euresys::Open_eVision

```
[C++]
void AdaptiveThreshold(
 const EROIBW8* src,
 EROIBW8* dst,
 Euresys::Open_eVision::EAdaptiveThresholdMethod method,
 int halfKernelSize,
 int constant
 )
```

Parameters

src

-

dst

-

method

The thresholding mode, as defined by the enumeration EAdaptiveThresholdMethod.

halfKernelSize

Half width of the kernel rounded down

constant

Constant offset applied to the threshold value. By default (argument omitted) 0, i.e. no change.

Remarks

Kernel size is always odd.

EasyImage::AlphaBlend

Draws an image over an other.

Namespace: Euresys::Open_eVision

```
[C++]
void AlphaBlend(
 const EROIBW8& sourceImage,
 EROIBW8& destinationImage,
 double opacity
 )
```

Parameters

sourceImage

Foreground image.

destinationImage

Background image.

opacity

Opacity of the foreground image.

EasyImage::AnalyseHistogram

Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).

Namespace: Euresys::Open_eVision

```
[C++]
float AnalyseHistogram(
 EBWHistogramVector* histogram,
 Euresys::Open_eVision::EHistogramFeature operation,
 int minimumIndex,
 int maximumIndex
 )
```

Parameters

histogram

Pointer to the histogram vector.

operation

Parameter to be computed, as defined by EHistogramFeature.

minimumIndex

Starting index of the gray-level range.

maximumIndex

Ending index of the gray-level range.

EasyImage::AnalyseHistogramBW16

Returns a floating-point statistical parameter extracted from a range of gray levels in an image histogram (most/least frequent value/frequency, min/max value, count, average, standard deviation).

Namespace: Euresys::Open_eVision

```
[C++]
float AnalyseHistogramBW16(
 EBWHistogramVector* histogram,
 Euresys::Open_eVision::EHistogramFeature operation,
 int minimumIndex,
 int maximumIndex
 )
```

Parameters

histogram
Pointer to the histogram vector.

operation
Parameter to be computed, as defined by EHistogramFeature.

minimumIndex
Starting index of the gray-level range.

maximumIndex
Ending index of the gray-level range.

EasyImage::Area

Counts the pixels whose values are above (or on) a threshold.

Namespace: Euresys::Open_eVision

[C++]

```
void Area(
 const EROIBW8* sourceImage,
 EBW8 threshold,
 int& numberOfPixelsAboveThreshold
 )

void Area(
 const EROIBW16* sourceImage,
 EBW16 threshold,
 int& numberOfPixelsAboveThreshold
 )

void Area(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 threshold,
 int& numberOfPixelsAboveThreshold
 )

void Area(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 threshold,
 int& numberOfPixelsAboveThreshold
 )

void Area(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 threshold,
 int& numberOfPixelsAboveThreshold
 )
```

```
void Area(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 threshold,
 int& numberOfPixelsAboveThreshold
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

threshold

The pixel thresholding value used to count the pixels

numberOfPixelsAboveThreshold

Reference to the count of pixels above or equal to the threshold.

region

Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::AreaDoubleThreshold

Counts the pixels whose values are comprised between (or on) two thresholds.

Namespace: Euresys::Open_eVision

[C++]

```
void AreaDoubleThreshold(
 const EROIBW8* sourceImage,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 int& numberOfPixelsBetweenThresholds
 )

void AreaDoubleThreshold(
 const EROIBW16* sourceImage,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 int& numberOfPixelsBetweenThresholds
 )

void AreaDoubleThreshold(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 int& numberOfPixelsBetweenThresholds
 )
```

```
void AreaDoubleThreshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 int& numberOfPixelsBetweenThresholds
 )
void AreaDoubleThreshold(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 int& numberOfPixelsBetweenThresholds
 )
void AreaDoubleThreshold(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 int& numberOfPixelsBetweenThresholds
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

lowThreshold

Inferior threshold.

highThreshold

Superior threshold.

numberOfPixelsBetweenThresholds

Reference to the count of pixels that are above or equal to the inferior threshold, and strictly below the superior threshold.

region

Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::ArgumentImage

Prepares a lookup-table image for use for gradient argument computation.

Namespace: Euresys::Open_eVision

[C++]

```
void ArgumentImage(
 EImageBW8* destinationImage,
 EBW8 phase,
 float period
 )

void ArgumentImage(
 EImageBW8* destinationImage
 )

void ArgumentImage(
 EImageBW8* destinationImage,
 EBW8 phase
 )
```

Parameters

destinationImage

Pointer to the destination image.

phase

Argument value corresponding to the horizontal direction, in 256-th (65,536-th) of the period (by default, phase = 0).

period

Range of argument values corresponding to the 0..255 (0..65535) interval, in the current angle unit (by default, period = 0).

Remarks

The scale and phase of the gradient argument can be adjusted. The argument angles are counter clockwise on a 0..255 scale in the BW8 context and on a 0..65535 scale in the BW16 one, corresponding to a specified range (full turn by default, specified period otherwise). The argument phase is counted on a 0..255 scale or on a 0..65535 scale too. Angle values outside the 0..255 (0..65535) interval are wrapped. The period length is given in the current angle unit. EasyImage::ArgumentImage sets a lookup-table image for use with function EasyImage::GradientScalar, ready to compute the argument of the gradient in the source image, i.e. its direction. The argument will be returned as a value in range 0..255 suitable for storage in an EImageBW8 or as a value in the range 0..65535 suitable for storage in an EImageBW16. The phase of the argument can be adjusted.

EasyImage::AutoThreshold

Returns a suitable threshold value for a gray-level image binarization.

Namespace: Euresys::Open_eVision

[C++]

```
EBW8 AutoThreshold(
 const EROIBW8* sourceImage,
 Euresys::Open_eVision::EThresholdMode thresholdMode,
 float relativeThresholdMode
 )
```

```
EBW16 AutoThreshold(
 const EROIBW16* sourceImage,
 Euresys::Open_eVision::EThresholdMode thresholdMode,
 float relativeThresholdMode
 )

EBW8 AutoThreshold(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EThresholdMode thresholdMode,
 float relativeThresholdMode
 )

EBW16 AutoThreshold(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EThresholdMode thresholdMode,
 float relativeThresholdMode
 )

EBW8 AutoThreshold(
 const EROIBW8* sourceImage,
 const ERegion& region,
 Euresys::Open_eVision::EThresholdMode thresholdMode,
 float relativeThresholdMode
 )

EBW16 AutoThreshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 Euresys::Open_eVision::EThresholdMode thresholdMode,
 float relativeThresholdMode
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

thresholdMode

The thresholding mode, as defined by the enumeration EThresholdMode. To use absolute thresholding, use directly the threshold value instead.

relativeThresholdMode

Fraction of the image pixels that will be set below the threshold. Only used when the threshold value is EThresholdMode_Relative (by default, relativeThresholdMode = 0.5).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

region

An Eregion object to apply the function only on a particular region in the image.

Remarks

Several modes are available: absolute (the threshold value is given readily in the thresholdMode parameter), relative (the threshold value is computed to obtain a desired fraction of the image pixels) or automatic (using three different criteria).

It is possible that, in the automatic or relative thresholding modes, the computed threshold exceeds the dynamic range of the return type. In this case, the value is clipped to the maximum value that is representable in the return type.

EasyImage::BiLevelBlackTopHatBox

Performs a top-hat filtering on a bilevel image (closed image minus source image) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelBlackTopHatBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

Remarks

This filter enhances the thin black features.

EasyImage::BiLevelBlackTopHatDisk

Performs a top-hat filtering on a bilevel image (closed image minus source image) on a quasi-circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelBlackTopHatDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

Remarks

This filter enhances the thin black features.

EasyImage::BiLevelCloseBox

Performs a closing on a bilevel image (dilation followed by erosion) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelCloseBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

EasyImage::BiLevelCloseDisk

Performs a closing on a bilevel image (dilation followed by erosion) on a quasi-circular kernel.

Namespace: Euresys::Open_eVision

```
[C++]
void BiLevelCloseDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

EasyImage::BiLevelDilateBox

Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a rectangular kernel.
For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.

Namespace: Euresys::Open_eVision

```
[C++]
void BiLevelDilateBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

EasyImage::BiLevelDilateDisk

Performs a dilation on a bilevel image (maximum of the pixel values in a defined neighborhood) on a quasi-circular kernel.
For bilevel images, this maximum can either be 0 (if all pixels are black in the given neighborhood), or the maximum possible pixel value.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelDilateDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth =1; 0 is allowed).

EasyImage::BiLevelErodeBox

Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a rectangular kernel.
For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelErodeBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

EasyImage::BiLevelErodeDisk

Performs an erosion on a bilevel image (minimum of the pixel values in a defined neighborhood) on a quasi-circular kernel.
For bilevel images, this minimum can either be 0, or the maximum possible pixel value (if all pixels are white in the given neighborhood)

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelErodeDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

EasyImage::BiLevelMedian

Applies a median filter to a bilevel image (median of the gray values in a 3x3 neighborhood).

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelMedian(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half height of the kernel minus one (by default, same as halfOfKernelWidth; 0 is allowed).

EasyImage::BiLevelMorphoGradientBox

Computes the morphological gradient of a bilevel image using a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelMorphoGradientBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element.
The kernel size is a pair of odd numbers; they must be halved before they are passed. For instance, a 3x5 kernel is passed as 1x2.

EasyImage::BiLevelMorphoGradientDisk

Computes the morphological gradient of a bilevel image using a quasi-circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelMorphoGradientDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element.

EasyImage::BiLevelOpenBox

Performs an opening on a bilevel image (erosion followed by dilation) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelOpenBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

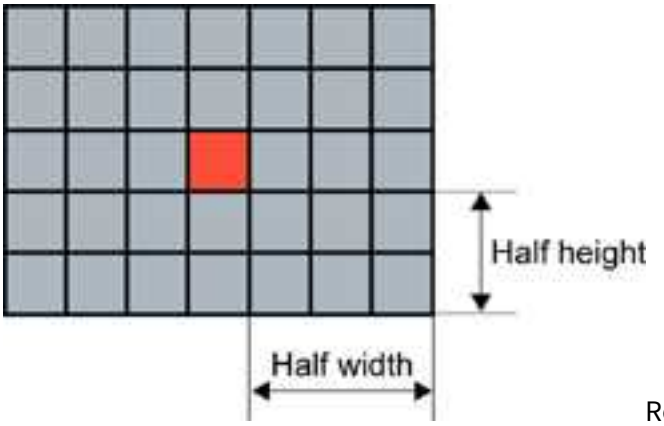
halfOfKernelWidth

Half of the box width minus one, as shown on the picture below (by default, halfOfKernelWidth =1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one, as shown on the picture below (by default, same as halfOfKernelWidth; 0 is allowed).

Remarks



Rectangular kernel of half width = 3 and half height = 2

EasyImage::BiLevelOpenDisk

Performs an opening on a bilevel image (erosion followed by dilation) on a quasi-circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelOpenDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one, as shown on the picture below (by default, halfOfKernelWidth = 1; 0 is allowed).

EasyImage::BiLevelThick

Applies a thickening operation on a bilevel image, using a 3x3 kernel.

Namespace: Euresys::Open_eVision

```
[C++]
void BiLevelThick(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EKernel* thickeningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thickeningKernel

Pointer to the thickening kernel.

rotationMode

Rotation mode, as defined by EKernelRotation.

numberOfIterations

Number of iterations to apply. 0 indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either EKernelRotation_Clockwise or EKernelRotation_Anticlockwise, a pass comprises eight kernel rotations.

Remarks

The thickening kernel coefficients must be 0 (matching black pixel, value 0), 1 (matching non black pixel, value > 0) or -1 (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to 255.

EasyImage::BiLevelThin

Applies a thinning operation on a bilevel image, using a 3x3 kernel.

Namespace: Euresys::Open_eVision

```
[C++]
void BiLevelThin(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EKernel* thinningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thinningKernel

Pointer to the thinning kernel.

rotationMode

Rotation mode, as defined by EKernelRotation.

numberOfIterations

Number of iterations to apply. 0 indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either EKernelRotation_Clockwise or EKernelRotation_Anticlockwise, a pass comprises eight kernel rotations.

Remarks

The thinning kernel coefficients must be 0 (matching black pixel, value 0), 1 (matching non black pixel, value > 0) or -1 (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to 0.

EasyImage::BiLevelWhiteTopHatBox

Performs a top-hat filtering on a bilevel image (source image minus open image) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelWhiteTopHatBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

Remarks

This filter enhances the thin white features.

EasyImage::BiLevelWhiteTopHatDisk

Performs a top-hat filtering on a bilevel image (source image minus open image) on a quasi-circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BiLevelWhiteTopHatDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

Remarks

This filter enhances the thin white features.

EasyImage::BinaryMoments

Computes the zero-th, first or second order moments on the binarized image, i.e. with a unit weight for those pixels with a value above or equal to the threshold, and zero otherwise.

Namespace: Euresys::Open_eVision

[C++]

```
void BinaryMoments(
 const EROIBW8* sourceImage,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My
 )

void BinaryMoments(
 const EROIBW16* sourceImage,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My
 )

void BinaryMoments(
 const EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My
 )

void BinaryMoments(
 const EROIBW16* sourceImage,
 const ERegion& region,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My
 )

void BinaryMoments(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My
 )
```

```
void BinaryMoments(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My
 )

void BinaryMoments(
 const EROIBW8* sourceImage,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )

void BinaryMoments(
 const EROIBW16* sourceImage,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )

void BinaryMoments(
 const EROIBW8* sourceImageconst,
 const ERegion& region,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )

void BinaryMoments(
 const EROIBW16* sourceImage,
 const ERegion& region,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
```

```
void BinaryMoments(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
void BinaryMoments(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 OEV_UINT32 threshold,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

threshold
Binarization threshold.

M
Reference to the zero-th order moment (area).

Mx
Reference to the first-order, uncentered moments (weighted sum of abscissas).

My
Reference to the first-order, uncentered moments (weighted sum of ordinates).

region
Pointer to a region to apply the function only on a particular region in the image.

mask
Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Mxx
Reference to the second-order, uncentered moments (weighted sum of squared abscissas).

Mxy
Reference to the second-order, uncentered moments (weighted sum of cross-product of abscissas and ordinates).

Myy
Reference to the second-order, uncentered moments (weighted sum of squared ordinates).

sourceImageconst

-

EasyImage::BlackTopHatBox

Performs a top-hat filtering on an image (closed image minus source image) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BlackTopHatBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void BlackTopHatBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void BlackTopHatBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void BlackTopHatBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void BlackTopHatBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void BlackTopHatBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void BlackTopHatBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

Remarks

This filter enhances the thin black features.

EasyImage::BlackTopHatDisk

Performs a top-hat filtering on an image (closed image minus source image) on a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void BlackTopHatDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void BlackTopHatDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void BlackTopHatDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void BlackTopHatDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void BlackTopHatDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void BlackTopHatDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void BlackTopHatDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

region

Region to apply the function on.

Remarks

This filter enhances the thin black features.

EasyImage::CloseBox

Performs a closing on an image (dilation followed by erosion) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void CloseBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void CloseBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void CloseBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void CloseBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void CloseBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void CloseBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void CloseBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

EasyImage::CloseDisk

Performs a closing on an image (dilation followed by erosion) on a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void CloseDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void CloseDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void CloseDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void CloseDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void CloseDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void CloseDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void CloseDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

region

Region to apply the function on.

EasyImage::Contour

Follows the *contour* of an object.

Namespace: Euresys::Open_eVision

[C++]

```
void Contour(
 EROIBW8* sourceImage,
 Euresys::Open_eVision::EContourMode contourMode,
 int startX,
 int startY,
 Euresys::Open_eVision::EContourThreshold thresholdMode,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::EConnexity connexity,
 EPathVector* path
 )
```

```
void Contour(
 EROIBW16* sourceImage,
 Euresys::Open_eVision::EContourMode contourMode,
 int startX,
 int startY,
 Euresys::Open_eVision::EContourThreshold thresholdMode,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::EConnexity connexity,
 EPathVector* path
 )
void Contour(
 EROIBW8* sourceImage,
 Euresys::Open_eVision::EContourMode contourMode,
 int startX,
 int startY,
 Euresys::Open_eVision::EContourThreshold thresholdMode,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::EConnexity connexity,
 EBW8PathVector* path,
 bool freeman
 )
void Contour(
 EROIBW16* sourceImage,
 Euresys::Open_eVision::EContourMode contourMode,
 int startX,
 int startY,
 Euresys::Open_eVision::EContourThreshold thresholdMode,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::EConnexity connexity,
 EBW16PathVector* path,
 bool freeman
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

contourMode
Traversal mode, as defined by EContourMode.

startX
Start point abscissa.

startY
Start point ordinate.

thresholdMode
Thresholding mode as defined by EThresholdMode.

threshold
Threshold level.

connexity
Contour connexity, as defined by EConnexity.

path

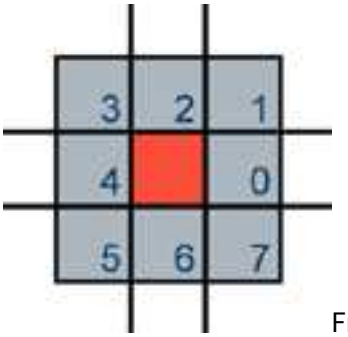
Pointer to the destination vector.

freeman

Specifies if Freeman codes are to be retrieved rather than pixel values.

Remarks

A threshold is applied so that objects become blobs. The contour is a closed or not (see property Get/SetClosed) connected path, forming the boundary of the blob.
When destination vector is an EBW8PathVector or a EBW16PathVector, this vector can contain two different information. If the bFreeman argument is false, which is the default value, member m_bw8(16)Pixel in the vector elements contains the gray-level value of the contour pixels. If it is true, the member instead gives the Freeman code leading from a pixel to next. The Freeman codes are numbered from 0 in the horizontal direction and incremented anticlockwise.



Freeman code, leading from a pixel to another adjacent pixel

EasyImage::Convert

Transforms the contents of an image to an image of another type.

Namespace: Euresys::Open_eVision

[C++]

```
void Convert(
 const EROIC24* sourceImage,
 EROIBW8* destinationImage
 )

void Convert(
 const EROIBW8* sourceImage,
 EROIC24* destinationImage
 )

void Convert(
 const EROIC24* sourceImage,
 EROIC15* destinationImage
 )

void Convert(
 const EROIC15* sourceImage,
 EROIC24* destinationImage
 )
```

```
void Convert(
 const EROIBW8* sourceImage,
 EROIC15* destinationImage
 )

void Convert(
 const EROIC15* sourceImage,
 EROIBW8* destinationImage
 )

void Convert(
 const EROIC24* sourceImage,
 EROIC16* destinationImage
 )

void Convert(
 const EROIC16* sourceImage,
 EROIC24* destinationImage
 )

void Convert(
 const EROIBW8* sourceImage,
 EROIC16* destinationImage
 )

void Convert(
 const EROIC16* sourceImage,
 EROIBW8* destinationImage
 )

void Convert(
 const EROIC24* sourceImage,
 EROIC24A* destinationImage
 )

void Convert(
 const EROIC24A* sourceImage,
 EROIC24* destinationImage
 )

void Convert(
 const EROIBW32* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 rightShift
 )

void Convert(
 const EROIBW32* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 rightShift
 )

void Convert(
 const EROIBW16* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 rightShift
 )
```

```
void Convert(
 const EROIBW8* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 leftShift
 )

void Convert(
 const EROIBW8* sourceImage,
 EROIBW32* destinationImage,
 OEV_UINT32 leftShift
 )

void Convert(
 const EROIBW16* sourceImage,
 EROIBW32* destinationImage,
 OEV_UINT32 leftShift
 )

void Convert(
 const EROIC24* sourceImage,
 const EROIBW8* sourceImageAlpha,
 EROIC24A* destinationImage
 )

void Convert(
 const EROIC24A* sourceImage,
 EROIC24* destinationImage,
 EROIBW8* destinationImageAlpha
 )

void Convert(
 const EROIC48* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 rightShift
 )

void Convert(
 const EROIC24* sourceImage,
 EROIC48* destinationImage,
 OEV_UINT32 leftShift
 )

void Convert(
 const EROIBW8* sourceImage,
 EROIBW1* destinationImage
 )

void Convert(
 const EROIBW16* sourceImage,
 EROIBW1* destinationImage
 )

void Convert(
 const EROIBW32* sourceImage,
 EROIBW1* destinationImage
 )
```

```
void Convert(
 const EROIBW1* sourceImage,
 EROIBW8* destinationImage,
 EBW8 highValue
 )

void Convert(
 const EROIBW1* sourceImage,
 EROIBW8* destinationImage
 )

void Convert(
 const EROIBW1* sourceImage,
 EROIBW16* destinationImage,
 EBW16 highValue
 )

void Convert(
 const EROIBW1* sourceImage,
 EROIBW16* destinationImage
 )

void Convert(
 const EROIBW1* sourceImage,
 EROIBW32* destinationImage,
 EBW32 highValue
 )

void Convert(
 const EROIBW1* sourceImage,
 EROIBW32* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

rightShift

Right shift amplitude. By default, left justified data is assumed.

leftShift

Left shift amplitude. By default, left justified data is assumed.

sourceImageAlpha

Pointer to the source alpha component (EImageBW8/EROIBW8).

destinationImageAlpha

Pointer to the destination alpha component (EImageBW8/EROIBW8).

highValue

In the case of black and white source images/ROIs, indicates to which gray level the value 1 should be mapped. By default, 1 is mapped to the highest allowed value for the destination image/ROI.

Remarks

Conversion to a black and white image (BW1)
Turns an 8-bit gray-level image into a black and white image.
Turns a 16-bit gray-level image into a black and white image.
Turns a 32-bit gray-level image into a black and white image.
Source pixels whose values is 0 are converted to black. All other source pixel values are converted to white.
Conversion to a 8-bit gray-level image (BW8)
Turns a black and white image into an 8-bit gray-level image.
Turns a 16-bit gray-level image into an 8-bit gray-level image. A right shift can be applied to the 16-bit data to adjust the magnitude, depending on how the 16-bit data is organized. For instance, if the source image holds 10 significant bits right justified, a right shift of 2 is required to drop the 2 low order bits; if the source image holds 12 bits left justified, a right shift of 8 is required and the 4 low order bits will be truncated.
Turns an EC15, EC16 or EC24 color image into an EBW8 gray-level image. The 3 color components are averaged following an equation based on the current ERgbStandard.
Conversion to a 16-bit gray-level image (BW16)
Turns a black and white image into a 16-bit gray-level image.
Turns an 8-bit gray-level image into a 16-bit gray-level image. A left shift can be applied to the 8-bit data to adjust the magnitude, depending on how the 16-bit data is organized. For instance, if the destination image holds 10 significant bits right justified, a shift of 2 is required; if the destination image holds 12 bits left justified, a shift of 8 is required.
Conversion to a 32-bit gray-level image (BW32)
Turns a black and white image into a 32-bit gray-level image.
Conversion to color images
Turns an 8-bit gray-level image into a true color equivalent. The color components are all set equal to the corresponding gray-level value.
Converts between standard and Windows' packing RGB color formats. When converting from an EC24 image to a EC15 or EC16 one, only the 5 (or 6) most significant bits of each color component are retained.
Converts between RGB 24-bit color image and RGB32 (also known as RGBA) color image. When converting from EC24 to EC24A, you can choose to provide or not the alpha component. On the other hand, when converting from EC24A to EC24, you can choose to conserve or not the alpha component. The alpha component is retrieved and set using an EImageBW8/EROIBW8.

EasyImage::ConvertTo422

Turns an 8-bit gray-level image into a YUV 4:2:2 encoded color image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvertTo422(
 const EROIBW8* sourceImage,
 EROIBW16* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Remarks

The Y component is set to the corresponding gray-level values, while the U and V components are set to 128 (achromatic light).

EasyImage::ConvolGabor

Computes the Gabor filter response of the source image and stores the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolGabor(
 EROIBW8* sourceImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )

void ConvolGabor(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )
```

```
void ConvolGabor(
 EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )

void ConvolGabor(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )

void ConvolGabor(
 EROIBW16* sourceImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )

void ConvolGabor(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )
```

```
void ConvolGabor(
 EROIBW16* sourceImage,
 const ERegion& region,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )
void ConvolGabor(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 float sigma,
 float gamma,
 float theta,
 float lambda,
 float psi,
 bool normalize
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

halfOfKernelWidth
Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight
Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

sigma
Spread of the Gaussian envelope. This value cannot be zero.

gamma
Ellipticity of the Gaussian.

theta
Orientation of the Gaussian and of the sine wave.

lambda
Wavelength of the sine wave. This value cannot be zero.

psi
Phase offset of the sine wave.

normalize
Specifies whether the kernel should undergo a normalization by the sum of its components before applying the convolution.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

EasyImage::ConvolGaussian

Applies Gaussian filtering (binomial weights) in rectangular kernel of odd size.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolGaussian(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolGaussian(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolGaussian(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolGaussian(
 const EBW8Vector* sourceImage,
 EBW8Vector* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void ConvolGaussian(
 const EBW16Vector* sourceImage,
 EBW16Vector* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void ConvolGaussian(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void ConvolGaussian(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolGaussian(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

EasyImage::ConvolGradient

Extracts the edges of an image by summing the absolute values of the Gradient X and Gradient Y derivatives and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolGradient(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolGradient(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
```

```
void ConvolGradient(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolGradient(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolGradient(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolGradient(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

EasyImage::ConvolGradientX

Derives an image along X using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolGradientX(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolGradientX(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
```

```
void ConvolGradientX(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolGradientX(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolGradientX(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolGradientX(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 0 0 0-1 0 1 0 0 0

EasyImage::ConvolGradientY

Derives an image along Y using a Gradient kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolGradientY(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
```

```
void ConvolGradientY(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
void ConvolGradientY(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
void ConvolGradientY(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )
void ConvolGradientY(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )
void ConvolGradientY(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 0 -1 00 0 00 1 0

EasyImage::ConvolHighpass1

Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolHighpass1(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolHighpass1(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolHighpass1(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolHighpass1(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolHighpass1(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolHighpass1(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 0 -1 0-1 5 -1 0 -1 0

EasyImage::ConvolHighpass2

Filters an image using a 3x3 high-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolHighpass2(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
void ConvolHighpass2(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
void ConvolHighpass2(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
void ConvolHighpass2(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )
void ConvolHighpass2(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )
void ConvolHighpass2(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: -1 -1 -1-1 9 -1-1 -1 -1

EasyImage::ConvolKernel

Performs a convolution in image space, i.e. applies a convolution kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolKernel(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EKernel* kernel
 )

void ConvolKernel(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 EKernel* kernel
 )

void ConvolKernel(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 EKernel* kernel
 )

void ConvolKernel(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 EKernel* kernel
 )

void ConvolKernel(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 EKernel* kernel
 )

void ConvolKernel(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 EKernel* kernel
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

kernel

Pointer to the convolution kernel.

region

Region to apply the function on.

EasyImage::ConvolLaplacian4

Takes the second derivative of an image using a 4-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolLaplacian4(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolLaplacian4(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolLaplacian4(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolLaplacian4(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolLaplacian4(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolLaplacian4(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 0 1 01 -4 10 1 0

EasyImage::ConvolLaplacian8

Takes the second derivative of an image using an 8-neighbor Laplacian kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolLaplacian8(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolLaplacian8(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolLaplacian8(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolLaplacian8(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolLaplacian8(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolLaplacian8(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 1 1 11 -8 11 1 1

EasyImage::ConvolLaplacianX

Takes the horizontal second derivative and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolLaplacianX(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
void ConvolLaplacianX(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
void ConvolLaplacianX(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
void ConvolLaplacianX(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )
void ConvolLaplacianX(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )
void ConvolLaplacianX(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 1 -2 1

EasyImage::ConvolLaplacianY

Takes the vertical second derivative and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

```
[C++]
void ConvolLaplacianY(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
void ConvolLaplacianY(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
void ConvolLaplacianY(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
void ConvolLaplacianY(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )
void ConvolLaplacianY(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )
void ConvolLaplacianY(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 1-2 1

EasyImage::ConvolLowpass1

Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolLowpass1(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolLowpass1(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolLowpass1(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolLowpass1(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolLowpass1(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolLowpass1(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 1 1 11 1 11 1 1

EasyImage::ConvolLowpass2

Filters an image using a 3x3 low-pass kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolLowpass2(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolLowpass2(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolLowpass2(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolLowpass2(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolLowpass2(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolLowpass2(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 1 1 11 0 11 1 1

EasyImage::ConvolLowpass3

Filters an image using a 3x3 low-pass kernel.

Namespace: Euresys::Open_eVision

```
[C++]
void ConvolLowpass3(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
void ConvolLowpass3(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
void ConvolLowpass3(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
void ConvolLowpass3(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )
void ConvolLowpass3(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )
void ConvolLowpass3(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: 1 2 12 4 21 2 1

EasyImage::ConvolPrewitt

Extracts the edges of an image by summing the absolute values of the Prewitt X and Prewitt Y derivatives and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolPrewitt(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolPrewitt(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolPrewitt(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolPrewitt(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolPrewitt(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolPrewitt(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

EasyImage::ConvolPrewittX

Derives an image along X using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolPrewittX(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolPrewittX(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolPrewittX(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolPrewittX(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolPrewittX(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolPrewittX(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: -1 0 1-1 0 1-1 0 1

EasyImage::ConvolPrewittY

Derives an image along Y using a Prewitt kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolPrewittY(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolPrewittY(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolPrewittY(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolPrewittY(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolPrewittY(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolPrewittY(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: -1 -1 -1 0 0 0 1 1 1

EasyImage::ConvolRoberts

The Roberts edge extraction filter is based on a 2x2 kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolRoberts(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolRoberts(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolRoberts(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolRoberts(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolRoberts(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolRoberts(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

It computes the sum of absolute differences of the pixel values in the diagonal directions.

EasyImage::ConvolSobel

Extracts the edges of an image by summing the absolute values of the Sobel X and Sobel Y derivatives.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolSobel(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
void ConvolSobel(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
void ConvolSobel(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
void ConvolSobel(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )
void ConvolSobel(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )
void ConvolSobel(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

EasyImage::ConvolSobelX

Derives an image along X using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolSobelX(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolSobelX(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolSobelX(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolSobelX(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolSobelX(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolSobelX(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: -1 0 1-2 0 2-1 0 1

EasyImage::ConvolSobelY

Derives an image along Y using a Sobel kernel and stores the absolute value (magnitude) of the result in the destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolSobelY(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void ConvolSobelY(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void ConvolSobelY(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void ConvolSobelY(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void ConvolSobelY(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void ConvolSobelY(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

region

Region to apply the function on.

Remarks

Filtering kernel: -1 -2 -1 0 0 0 1 2 1

EasyImage::ConvolSymmetricKernel

Performs a convolution in image space, i.e. applies a square symmetric convolution kernel of size 3x3, 5x5 or 7x7.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolSymmetricKernel(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EKernel* kernel
 )

void ConvolSymmetricKernel(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 EKernel* kernel
 )

void ConvolSymmetricKernel(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 EKernel* kernel
 )

void ConvolSymmetricKernel(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 EKernel* kernel
 )

void ConvolSymmetricKernel(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 EKernel* kernel
 )

void ConvolSymmetricKernel(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 EKernel* kernel
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

destinationImage
Pointer to the destination image/ROI.

kernel
Pointer to the convolution kernel.

region
Region to apply the function on.

Remarks

This function is a synonym for EasyImage::ConvolKernel.

EasyImage::ConvolUniform

Applies strong low-pass filtering to an image by using a uniform rectangular kernel of odd size.

Namespace: Euresys::Open_eVision

[C++]

```
void ConvolUniform(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolUniform(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolUniform(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ConvolUniform(
 EBW8Vector* sourceVector,
 EBW8Vector* destinationVector,
 OEV_UINT32 halfOfKernelWidth
 )

void ConvolUniform(
 EBW16Vector* sourceVector,
 EBW16Vector* destinationVector,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void ConvolUniform(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void ConvolUniform(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void ConvolUniform(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image) and the default value for un32HalfWidth (1) has to be used.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

sourceVector

Pointer to the source vector.

destinationVector

Pointer to the destination vector. If NULL (default), this operation is destructive (i.e. applied to the source vector) and the default value for un32HalfWidth (1) has to be used.

region

Region to apply the function on.

Remarks

This filter replaces every pixel values by the arithmetic mean of the neighboring values in a rectangular window. To handle pixels along edges, the source pixels are replicated outwards as many times as required.
A very nice feature of this function is that its running time does not depend on the kernel size!

EasyImage::Copy

Copies a source image or a constant in a destination image.

Namespace: Euresys::Open_eVision

[C++]

```
void Copy(
 const EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void Copy(
 const EROIBW32* sourceImage,
 EROIBW32* destinationImage
 )

void Copy(
 const EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void Copy(
 const EROIC24A* sourceImage,
 EROIC24A* destinationImage
 )

void Copy(
 const EROIC15* sourceImage,
 EROIC15* destinationImage
 )

void Copy(
 const EROIC16* sourceImage,
 EROIC16* destinationImage
 )

void Copy(
 const EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Copy(
 const EROIC48* sourceImage,
 EROIC48* destinationImage
 )

void Copy(
 EBW16 constant,
 EROIBW16* destinationImage
 )
```

```
void Copy(
 EBW32 constant,
 EROIBW32* destinationImage
 )

void Copy(
 EC24 constant,
 EROIC24* destinationImage
 )

void Copy(
 EC15 constant,
 EROIC15* destinationImage
 )

void Copy(
 EC16 constant,
 EROIC16* destinationImage
 )

void Copy(
 EBW8 constant,
 EROIBW8* destinationImage
 )

void Copy(
 EC24A constant,
 EROIC24A* destinationImage
 )

void Copy(
 EC48 constant,
 EROIC48* destinationImage
 )

void Copy(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void Copy(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void Copy(
 const EROIBW32* sourceImage,
 const ERegion& region,
 EROIBW32* destinationImage
 )

void Copy(
 const EROIC48* sourceImage,
 const ERegion& region,
 EROIC48* destinationImage
 )
```

```
void Copy(
 const EROIC24A* sourceImage,
 const ERegion& region,
 EROIC24A* destinationImage
 )

void Copy(
 const EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage
 )

void Copy(
 const EROIC15* sourceImage,
 const ERegion& region,
 EROIC15* destinationImage
 )

void Copy(
 const EROIC16* sourceImage,
 const ERegion& region,
 EROIC16* destinationImage
 )

void Copy(
 EBW8 constant,
 const ERegion& region,
 EROIBW8* destinationImage
 )

void Copy(
 EBW16 constant,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void Copy(
 EBW32 constant,
 const ERegion& region,
 EROIBW32* destinationImage
 )

void Copy(
 EC48 constant,
 const ERegion& region,
 EROIC48* destinationImage
 )

void Copy(
 EC24A constant,
 const ERegion& region,
 EROIC24A* destinationImage
 )
```

```
void Copy(
 EC24 constant,
 const ERegion& region,
 EROIC24* destinationImage
 )

void Copy(
 EC15 constant,
 const ERegion& region,
 EROIC15* destinationImage
 )

void Copy(
 EC16 constant,
 const ERegion& region,
 EROIC16* destinationImage
 )

void Copy(
 const EROIBW1* sourceImage,
 EROIBW1* destinationImage
 )

void Copy(
 EBW1 constant,
 EROIBW1* destinationImage
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

destinationImage
Pointer to the destination image/ROI.

constant
Gray-level or color constant.

region
Region on which to copy.

EasyImage::CumulateHistogram

Cumulates histogram values in another histogram.

Namespace: Euresys::Open_eVision

[C++]

```
void CumulateHistogram(
 EBWHistogramVector* sourceVector,
 EBWHistogramVector* destinationVector
 )
```

Parameters

sourceVector

Pointer to the source vector.

destinationVector

Pointer to the destination vector.

Remarks

Calling this function after EasyImage::Histogram allows you to compute the cumulative histogram of an image, i.e. the count of pixels below a given threshold value (instead of the count of pixels with a given gray value, as computed by EasyImage::Histogram).

EasyImage::DilateBox

Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void DilateBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void DilateBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void DilateBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void DilateBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void DilateBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void DilateBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void DilateBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

EasyImage::DilateDisk

Performs a dilation on an image (maximum of the gray values in a defined neighborhood) on a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void DilateDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void DilateDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void DilateDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void DilateDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void DilateDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void DilateDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void DilateDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth =1; 0 is allowed).

region

Region to apply the function on.

EasyImage::Distance

Computes the morphological distance function on a binary image (0 for black, non 0 for white).

Namespace: Euresys::Open_eVision

```
[C++]
void Distance(
 EROIBW8* sourceImage,
 EImageBW16* destinationImage,
 int valueOutOfImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

valueOutOfImage

Out-of-bounds image value. By default, this value is 0.

Remarks

So, each pixel of the destination image will contain, at the end of the processing, the morphological distance of the corresponding pixel in the source image. The distance function at a given pixel tells how many erosion passes are required to set it to black.

EasyImage::DoubleThreshold

Converts an image by setting all pixels below the low threshold to a low value, all pixels above the high threshold to a high value, and the remaining pixels to an intermediate value.

Namespace: Euresys::Open_eVision

```
[C++]
void DoubleThreshold(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold,
 OEV_UINT8 lowValue,
 OEV_UINT8 middleValue,
 OEV_UINT8 highValue
 )
```

```
void DoubleThreshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold,
 EBW16 lowValue,
 EBW16 middleValue,
 EBW16 highValue
 )

void DoubleThreshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold
 )

void DoubleThreshold(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold,
 OEV_UINT8 lowValue,
 OEV_UINT8 middleValue,
 OEV_UINT8 highValue
 )

void DoubleThreshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold,
 EBW16 lowValue,
 EBW16 middleValue,
 EBW16 highValue
 )

void DoubleThreshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lowThreshold

Low threshold value.

highThreshold

High threshold value.

lowValue

Value for pixels strictly below the low threshold.

middleValue

Value for pixels that are above or equal to the low threshold, and below or equal the high threshold.

highValue

Value for pixels strictly above to the high threshold.

region

Pointer to a region to apply the function only on a particular region in the image.

EasyImage::Equalize

Equalizes an image histogram (the gray levels are re-mapped so that the histogram becomes as close to uniform as possible).

Namespace: Euresys::Open_eVision

[C++]

```
void Equalize(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Equalize(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Remarks

This strongly enhances the contrast in dark areas.

EasyImage::ErodeBox

Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void ErodeBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ErodeBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ErodeBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ErodeBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ErodeBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void ErodeBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void ErodeBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

EasyImage::ErodeDisk

Performs an erosion on an image (minimum of the gray values in a defined neighborhood) on a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void ErodeDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void ErodeDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void ErodeDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void ErodeDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void ErodeDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void ErodeDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void ErodeDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

region

Region to apply the function on.

EasyImage::Flip

Flip an image around either the vertical axis, the horizontal axis or both.

Namespace: Euresys::Open_eVision

[C++]

```
void Flip(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 Euresys::Open_eVision::EFlipAxis axis
 )
```

```
void Flip(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 Euresys::Open_eVision::EFlipAxis axis
 )

void Flip(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 Euresys::Open_eVision::EFlipAxis axis
 )

void Flip(
 EROIBW8* sourceImage,
 Euresys::Open_eVision::EFlipAxis axis
 )

void Flip(
 EROIBW16* sourceImage,
 Euresys::Open_eVision::EFlipAxis axis
 )

void Flip(
 EROIC24* sourceImage,
 Euresys::Open_eVision::EFlipAxis axis
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. May not be the same as the source image.

axis

Axis around which the ROI flips.

Remarks

Destination image/roi size should be the same as the source image/roi size.

EasyImage::Focusing

Returns a measure of the focusing of an image by computing the total gradient energy.

Namespace: Euresys::Open_eVision

[C++]

```
float Focusing(
 EROIBW8* image
 )

float Focusing(
 EROIBW16* image
 )
```

```
float Focusing(
 EROIC24* image
 )
```

Parameters

image

Pointer to the source image/ROI.

Remarks

When this quantity is maximum for a given image, sharp focusing is achieved.
For more information, please refer to the section **Using Open eVision -> EasyImage -> Computing Image Statistics -> Image Focus** in the documentation.

EasyImage::Gain

Transforms an image, applying a gain and offset to all pixels.

Namespace: Euresys::Open_eVision

[C++]

```
void Gain(
 const EROIC24* sourceImage,
 EROIC24* destinationImage,
 EColor Gain
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Gain

Constant gain. By default (argument omitted) 1, i.e. no change.

Remarks

The gain should remain close to 1, and allows contrast adjustment of the image.

The offset can be positive or negative, and allows to adjust the image intensity. The resulting values are always saturated to range [0..255].

For color images, the separate gain and offset values are specified as triple of values stored in a EColor. The default values leave the image unchanged.

Internally, the computations are achieved through fixed-point arithmetic with 5 bits of precision for the fractional part. This can result in loss of precision with small gains.

EasyImage::GainOffset

Transforms an image, applying a gain and offset to all pixels.

Namespace: Euresys::Open_eVision

[C++]

```
void GainOffset(
 const EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 float gain,
 float offset
 )

void GainOffset(
 const EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float gain,
 float offset
 )

void GainOffset(
 const EROIC24* sourceImage,
 EROIC24* destinationImage,
 EColor gain,
 EColor offset
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

gain

Constant gain. By default (argument omitted) 1, i.e. no change.

offset

Constant offset. By default (argument omitted) 0, i.e. no change.

Remarks

The gain should remain close to 1, and allows contrast adjustment of the image.

The offset can be positive or negative, and allows to adjust the image intensity. The resulting values are always saturated to range [0..255].

For color images, the separate gain and offset values are specified as triple of values stored in a EColor. The default values leave the image unchanged.

Internally, the computations are achieved through fixed-point arithmetic with 5 bits of precision for the fractional part. This can result in loss of precision with small gains.

EasyImage::GetFrame

Extracts the frame of given parity from an image.

Namespace: Euresys::Open_eVision

```
[C++]
void GetFrame(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 bool odd
 )
void GetFrame(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 bool odd
 )
void GetFrame(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 bool odd
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

odd

Specifies which frame is extracted (the frame made up of all lines of the same parity as odd).

Remarks

The size of the destination image is determined as follows:
DstImage_Width = SrcImage_Width
DstImage_Height = (SrcImage_Height + 1 - odd) / 2

EasyImage::GetProfilePeaks

Detects peaks in a gray-level profile. Maxima as well as minima are considered.

Namespace: Euresys::Open_eVision

[C++]

```
void GetProfilePeaks(
 EBW8Vector* profile,
 EPeakVector* peaks,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold,
 OEV_UINT32 minimumAmplitude,
 OEV_UINT32 minimumArea
 )
void GetProfilePeaks(
 EBW16Vector* profile,
 EPeakVector* peaks,
 OEV_UINT32 lowThreshold,
 OEV_UINT32 highThreshold,
 OEV_UINT32 minimumAmplitude,
 OEV_UINT32 minimumArea
 )
```

Parameters

profile

Pointer to the source vector.

peaks

Pointer to the destination vector.

lowThreshold

Threshold used for the minimum peaks.

highThreshold

Threshold used for the maximum peaks.

minimumAmplitude

Minimum amplitude required for a peak to be kept (may be 0).

minimumArea

Minimum area required for a peak to be kept (may be 0).

Remarks

To eliminate false peaks due to noise, two selection criteria are used.
A peak is the portion of the signal that is above [below] a given threshold. The peak amplitude is defined to be the difference between the threshold value and the maximum [minimum] signal value. The peak area is defined to be the surface comprised between the signal curve and the horizontal line at the given threshold.
The result is stored in a peaks vector.

EasyImage::GradientScalar

Computes the (scalar) gradient image derived from a given source image.

Namespace: Euresys::Open_eVision

[C++]

```
void GradientScalar(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EROIBW8* lookupTable
 )
void GradientScalar(
 EROIBW32* sourceImage,
 EROIBW8* destinationImage,
 EROIBW8* lookupTable
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lookupTable

Pointer to the image/ROI used as a preset lookup-table. This lookup table can be generated by one of EasyImage::ArgumentImage or EasyImage::ModulusImage, or be user-defined.

Remarks

The scalar value derived from the gradient depends on the preset lookup-table image.
The gradient of a gray-scale image corresponds to a vector, the components of which are the partial derivatives of the gray-level signal in the horizontal and vertical direction. A vector can be characterized by a direction and a length, corresponding to the gradient orientation, here called *argument*, and the gradient magnitude, here called *magnitude*.
Function EasyImage::GradientScalar generates a gradient direction or gradient magnitude map (gray-level image) from a given gray-level image. For efficiency, a pre-computed lookup-table is used to define the desired transformation. This lookup-table is stored as a standard EImageBW8/EImageBW16. Use one of EasyImage::ArgumentImage or EasyImage::ModulusImage once before calling EasyImage::GradientScalar.

EasyImage::GravityCenter

Computes the coordinates of the gravity center of an image, i.e. the average coordinates of the pixels above (or on) the threshold.

Namespace: Euresys::Open_eVision

[C++]

```
void GravityCenter(
 const EROIBW8* sourceImage,
 OEV_UINT32 threshold,
 float& gravityX,
 float& gravityY
 )
```

```
void GravityCenter(
 const EROIBW16* sourceImage,
 OEV_UINT32 threshold,
 float& gravityX,
 float& gravityY
 )
void GravityCenter(
 const EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 threshold,
 float& gravityX,
 float& gravityY
 )
void GravityCenter(
 const EROIBW16* sourceImage,
 const ERegion& region,
 OEV_UINT32 threshold,
 float& gravityX,
 float& gravityY
 )
void GravityCenter(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 OEV_UINT32 threshold,
 float& gravityX,
 float& gravityY
 )
void GravityCenter(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 OEV_UINT32 threshold,
 float& gravityX,
 float& gravityY
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

threshold
Threshold.

gravityX
Reference to the gravity center abscissa.

gravityY
Reference to the gravity center ordinate.

region
Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::HDRFusion

Fuses two images using HDR principles.

Namespace: Euresys::Open_eVision

[C++]

```
void HDRFusion(
 const EROIBW8* darkSrc,
 const EROIBW8* lightSrc,
 int shutterSpeedFactor,
 EROIBW16* dst
 )

void HDRFusion(
 const EROIBW16* darkSrc,
 const EROIBW16* lightSrc,
 int shutterSpeedFactor,
 EROIBW16* dst
 )

void HDRFusion(
 const EROIBW16* darkSrc,
 const EROIBW16* lightSrc,
 int shutterSpeedFactor,
 EROIBW32* dst
 )

void HDRFusion(
 const EROIC24* darkSrc,
 const EROIC24* lightSrc,
 int shutterSpeedFactor,
 EROIC48* dst
 )

void HDRFusion(
 const EROIC48* darkSrc,
 const EROIC48* lightSrc,
 int shutterSpeedFactor,
 EROIC48* dst
 )
```

Parameters

darkSrc

Dark input image (high shutter speed).

lightSrc

Light input image (low shutter speed).

shutterSpeedFactor

Shutter speed factor between light and dark image.

dst

Destination image.

EasyImage::Histogram

Computes the histogram of an image (count of each gray-level value).

Namespace: Euresys::Open_eVision

[C++]

```
void Histogram(
 const EROIBW8* sourceImage,
 EBWHistogramVector* histogram
 )

void Histogram(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBWHistogramVector* histogram
 )

void Histogram(
 const EROIBW16* sourceImage,
 EBWHistogramVector* histogram,
 OEV_UINT32 mostSignificantBit,
 OEV_UINT32 numberOfSignificantBits,
 bool saturate
 )

void Histogram(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBWHistogramVector* histogram,
 OEV_UINT32 mostSignificantBit,
 OEV_UINT32 numberOfSignificantBits,
 bool saturate
 )

void Histogram(
 const EROIBW32* sourceImage,
 EBWHistogramVector* histogram,
 OEV_UINT32 mostSignificantBit,
 OEV_UINT32 numberOfSignificantBits,
 bool saturate
 )
```

```
void Histogram(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBWHistogramVector* histogram
 )
void Histogram(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBWHistogramVector* histogram,
 OEV_UINT32 mostSignificantBit,
 OEV_UINT32 numberOfSignificantBits,
 bool saturate
 )
void Histogram(
 const EROIBW32* sourceImage,
 const EROIBW8* mask,
 EBWHistogramVector* histogram,
 OEV_UINT32 mostSignificantBit,
 OEV_UINT32 numberOfSignificantBits,
 bool saturate
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

histogram

Pointer to the destination vector.

region

Pointer to a region to apply the function only on a particular region in the image.

mostSignificantBit

Index of the most significant bit of the pixels (0 has weight 1).

numberOfSignificantBits

Number of significant bits; the histogram will possess $2^{numberOfSignificantBits}$ entries.

saturate

Boolean indicating if values larger than $2^{mostSignificantBit}$-1 are saturated (default true) or not (false).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::HistogramThreshold

Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.

Namespace: Euresys::Open_eVision

```
[C++]
void HistogramThreshold(
 EBWHistogramVector* histogram,
 OEV_UINT32& threshold,
 float& averageOfPixelsBelowThreshold,
 float& averageOfPixelsAboveThreshold,
 float relativeThreshold,
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

histogram

Pointer to a vector conaining an image histogram.

threshold

reference to the threshold value. Before calling the function, must be set to the appropriate thresholding mode, as defined by EThresholdMode.

averageOfPixelsBelowThreshold

Average gray level of the dark pixels (below threshold).

averageOfPixelsAboveThreshold

Average gray level of the light pixels (above threshold).

relativeThreshold

Relative threshold value, relevant only in the EThresholdMode_Relative mode.

from

Lower bound of the analyzed gray-level range.

to

Upper bound of the analyzed gray-level range.

Remarks

Additionally, returns the average gray levels in the regions below and above the threshold. The threshold level can be computed by analyzing a range of gray levels in the histogram.

EasyImage::HistogramThresholdBW16

Determines an appropriate threshold level based on the histogram contents, using an automatic threshold mode.

Namespace: Euresys::Open_eVision

[C++]

```
void HistogramThresholdBW16(
 EBWHistogramVector* histogram,
 OEV_UINT32& threshold,
 float& averageOfPixelsBelowThreshold,
 float& averageOfPixelsAboveThreshold,
 float relativeThreshold,
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

histogram

Pointer to a vector conaining an image histogram.

threshold

reference to the threshold value. Before calling the function, must be set to the appropriate thresholding mode, as defined by EThresholdMode.

averageOfPixelsBelowThreshold

Average gray level of the dark pixels (below threshold).

averageOfPixelsAboveThreshold

Average gray level of the light pixels (above threshold).

relativeThreshold

Relative threshold value, relevant only in the EThresholdMode_Relative mode.

from

Lower bound of the analyzed gray-level range.

to

Upper bound of the analyzed gray-level range.

Remarks

Additionally, returns the average gray levels in the regions below and above the threshold. The threshold level can be computed by analyzing a range of gray levels in the histogram.

EasyImage::HitAndMiss

Apply the morphological hit-and-miss transform to detect a particular pattern of foreground and background pixels in a BW8, BW16 or C24 image/ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void HitAndMiss(
 const EROIBW8* source,
 EROIBW8* destination,
 const EHitAndMissKernel& kernel
 )
```

```
void HitAndMiss(
 const EROIBW16* source,
 EROIBW16* destination,
 const EHitAndMissKernel& kernel
 )
void HitAndMiss(
 const EROIC24* source,
 EROIC24* destination,
 const EHitAndMissKernel& kernel
 )
```

Parameters

source

The source image/ROI.

destination

The destination image/ROI.

kernel

The hit-and-miss kernel.

EasyImage::HorizontalMirror

Mirrors an image horizontally (the columns are swapped).

Namespace: Euresys::Open_eVision

[C++]

```
void HorizontalMirror(
 EROIBW8* sourceImage
 )
void HorizontalMirror(
 EROIC24* sourceImage
 )
void HorizontalMirror(
 EROIBW16* sourceImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

EasyImage::ImageToLineSegment

Copies the pixel values along a given line segment (arbitrarily oriented) to a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void ImageToLineSegment(
 const EROIBW8* sourceImage,
 EBW8Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIBW16* sourceImage,
 EBW16Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIC24* sourceImage,
 EC24Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 outOfMaskValue,
 EBW8Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 outOfMaskValue,
 EBW16Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )
```

```
void ImageToLineSegment(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EC24 outOfMaskValue,
 EC24Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIBW8* sourceImage,
 EBW8Vector* path,
 const ELine& line
 )

void ImageToLineSegment(
 const EROIBW16* sourceImage,
 EBW16Vector* path,
 const ELine& line
 )

void ImageToLineSegment(
 const EROIC24* sourceImage,
 EC24Vector* path,
 const ELine& line
 )

void ImageToLineSegment(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 outOfMaskValue,
 EBW8Vector* path,
 const ELine& line
 )

void ImageToLineSegment(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 outOfMaskValue,
 EBW16Vector* path,
 const ELine& line
 )

void ImageToLineSegment(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EC24 outOfMaskValue,
 EC24Vector* path,
 const ELine& line
 )
```

```
void ImageToLineSegment(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 outOfMaskValue,
 EBW8Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 outOfMaskValue,
 EBW16Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIC24* sourceImage,
 const ERegion& region,
 EC24 outOfMaskValue,
 EC24Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void ImageToLineSegment(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 outOfMaskValue,
 EBW8Vector* path,
 const ELine& line
 )

void ImageToLineSegment(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 outOfMaskValue,
 EBW16Vector* path,
 const ELine& line
 )
```

```
void ImageToLineSegment(
 const EROIC24* sourceImage,
 const ERegion& region,
 EC24 outOfMaskValue,
 EC24Vector* path,
 const ELine& line
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

path

Pointer to the destination vector.

X0

Coordinates of the starting point of the segment.

Y0

Coordinates of the starting point of the segment.

X1

Coordinates of the ending point of the segment.

Y1

Coordinates of the ending point of the segment.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

outOfMaskValue

The value to be given to the pixels that lie out of the mask or the region.

line

A Eline object

region

Reference to a region to apply the function only on a particular region in the image.

Remarks

The line segment must be wholly contained within the image. The vector length is adjusted automatically.

EasyImage::ImageToPath

Given a path described by coordinates in a path vector, copies the corresponding pixel values into the same vector.

Namespace: Euresys::Open_eVision

[C++]

```
void ImageToPath(
 const EROIBW8* sourceImage,
 EBW8PathVector* path
 )

void ImageToPath(
 const EROIBW16* sourceImage,
 EBW16PathVector* path
 )

void ImageToPath(
 const EROIC24* sourceImage,
 EC24PathVector* path
 )

void ImageToPath(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 outOfMaskValue,
 EBW8PathVector* path
 )

void ImageToPath(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 outOfMaskValue,
 EBW16PathVector* path
 )

void ImageToPath(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EC24 outOfMaskValue,
 EC24PathVector* path
 )

void ImageToPath(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 outOfMaskValue,
 EBW8PathVector* path
 )

void ImageToPath(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 outOfMaskValue,
 EBW16PathVector* path
 )
```

```
void ImageToPath(
 const EROIC24* sourceImage,
 const ERegion& region,
 EC24 outOfMaskValue,
 EC24PathVector* path
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

path

Pointer to the destination vector.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

outOfMaskValue

The value to be given to the pixels that lie out of the mask.

region

Reference to a region to apply the function only on a particular region in the image.

EasyImage::IsodataThreshold

Computes a suitable threshold value for a histogram.

Namespace: Euresys::Open_eVision

[C++]

```
EBW8 IsodataThreshold(
 EBWHistogramVector* histogram,
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

histogram

Pointer to a vector containing an image histogram.

from

Lower bound of the useful gray-level interval (by default, 0).

to

Upper bound of the useful gray-level interval (by default, 255).

Remarks

The "isodata" rule is used: if one computes the average gray level of pixels below the threshold and the average gray level of pixels above the threshold, the threshold lies exactly halfway between them. Optionally, the threshold selection can be restricted to a range of gray-level values.

It is possible that, in the automatic or relative thresholding modes, the computed threshold exceeds the dynamic range of the return type. In this case, the value is clipped to the maximum value that is representable in the return type.

EasyImage::IsodataThresholdBW16

Computes a suitable threshold value for a histogram.

Namespace: Euresys::Open_eVision

```
[C++]
EBW16 IsodataThresholdBW16(
 EBWHistogramVector* histogram,
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

histogram

Pointer to a vector containing an image histogram.

from

Lower bound of the useful gray-level interval (by default, 0).

to

Upper bound of the useful gray-level interval (by default, 65535).

Remarks

The "isodata" rule is used: if one computes the average gray level of pixels below the threshold and the average gray level of pixels above the threshold, the threshold lies exactly halfway between them. Optionally, the threshold selection can be restricted to a range of gray-level values.
Returns the threshold.

EasyImage::LinearTransform

Applies a general affine transformation.

Namespace: Euresys::Open_eVision

[C++]

```
void LinearTransform(
 EROIBW8* sourceImage,
 float Axx,
 float Axy,
 float Ax,
 float Ayx,
 float Ayy,
 float Ay,
 EROIBW8* destinationImage,
 int interpolationBits
 )

void LinearTransform(
 EROIBW16* sourceImage,
 float Axx,
 float Axy,
 float Ax,
 float Ayx,
 float Ayy,
 float Ay,
 EROIBW16* destinationImage,
 int interpolationBits
 )

void LinearTransform(
 EROIC24* sourceImage,
 float Axx,
 float Axy,
 float Ax,
 float Ayx,
 float Ayy,
 float Ay,
 EROIC24* destinationImage,
 int interpolationBits
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

Axx
See formula below.

Axy
See formula below.

Ax
See formula below.

Ayx
See formula below.

Ayy
See formula below.

Ay
See formula below.

destinationImage
Pointer to the destination image/ROI.

interpolationBits
Number of bits of accuracy for interpolation. Allowed values are 0 (no interpolation, nearest neighbor), 4 (linear interpolation) or 8 (cubic interpolation).

Remarks

An affine transformation is an important class of linear 2D geometric transformations which maps variables (e.g. pixel intensity values located at position (X_{Src}, Y_{Src}) in an input image) into new variables (e.g. (X_{Dst}, Y_{Dst}) in an output image) by applying a linear combination of translation, rotation, scaling and/or shearing (i.e. non-uniform scaling in some directions) operations.
The parameters of the EasyImage::LinearTransform function are the coefficients of the affine equations below:

$$X_{Dst} = A_{xx}X_{Src} + A_{xy}Y_{Src} + A_x$$

$$Y_{Dst} = A_{yx}X_{Src} + A_{yy}Y_{Src} + A_y$$

EasyImage::LineSegmentToImage

Copies the pixel values from a vector or a constant to the pixels of a given line segment (arbitrarily oriented).

Namespace: Euresys::Open_eVision

[C++]

```
void LineSegmentToImage(
 EROIBW8* destinationImage,
 EBW8 pixel,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void LineSegmentToImage(
 EROIBW16* destinationImage,
 EBW16 pixel,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void LineSegmentToImage(
 EROIC24* destinationImage,
 EC24 pixel,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void LineSegmentToImage(
 EROIBW8* destinationImage,
 EBW8Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void LineSegmentToImage(
 EROIBW16* destinationImage,
 EBW16Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )

void LineSegmentToImage(
 EROIC24* destinationImage,
 EC24Vector* path,
 int X0,
 int Y0,
 int X1,
 int Y1
 )
```

Parameters

destinationImage
Pointer to the destination image/ROI.

pixel
Constant color value.

X0
Coordinates of the starting point of the segment.

Y0
Coordinates of the starting point of the segment.

X1
Coordinates of the ending point of the segment.

Y1
Coordinates of the ending point of the segment.

path
Pointer to the source vector.

Remarks

The line segment must be wholly contained within the image.

EasyImage::LocalAverage

Computes the average in a rectangular window centered on every pixel.

Namespace: Euresys::Open_eVision

[C++]

```
void LocalAverage(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfWidth,
 OEV_UINT32 halfHeight
 )

void LocalAverage(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfWidth,
 OEV_UINT32 halfHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

halfWidth

Half of the window width minus one.

halfHeight

Half of the window height minus one.

Remarks

The window dimensions can be an arbitrary odd integer.
The running time of this function does not depend on the window size.

EasyImage::LocalDeviation

Computes the standard deviation in a rectangular window centered on every pixel.

Namespace: Euresys::Open_eVision

[C++]

```
void LocalDeviation(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfWidth,
 OEV_UINT32 halfHeight
 )

void LocalDeviation(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfWidth,
 OEV_UINT32 halfHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfWidth

Half of the window width minus one.

halfHeight

Half of the window height minus one.

Remarks

The window dimensions can be an arbitrary odd integer.
The running time of this function does not depend on the window size.

EasyImage::Lut

Transforms the gray levels of an image, using a lookup table stored in a vector (of unsigned values).

Namespace: Euresys::Open_eVision

[C++]

```
void Lut(
 const EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 const EBW16Vector* lookupTable
 )

void Lut(
 const EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 const EBW8Vector* lookupTable
 )

void Lut(
 const EROIBW16* sourceImage,
 EROIBW8* destinationImage,
 const EBW8Vector* lookupTable,
 OEV_UINT32 numberOfScalingBits
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

lookupTable

Pointer to the lookup vector.

numberOfScalingBits

Number of scaling bits (or right padding bits).

Remarks

A 16-bit image usually does not make use of its 16 bits. In most cases, only 10 or 12 bits are used. These bits are called *significant bits*. In the 16-bit information, significant bits can be left aligned, right aligned or not aligned at all. To indicate which are the significant bits, we have to tell how many bits are significant and the number of right padding bits (0 right padding bit means that significant bits are right aligned).

The number of significant bits is given by the number of Look Up table entries. For example a Lut of 1024 entries is used for an image of 10 significant bits (as $2^{10} = 1024$).

The number of right padding bits is given by means of the numberOfScalingBits parameter. Leaving this parameter undefined indicates that the significant bits are left aligned on the word.

EasyImage::MatchFrames

Determines the optimal shift amplitude by comparing two successive lines of the image.

Namespace: Euresys::Open_eVision

```
[C++]
void MatchFrames(
 EROIBW8* sourceImage,
 int fixedRow,
 int minimumOffset,
 int maximumOffset,
 int& bestOffset
 )
void MatchFrames(
 EROIBW16* sourceImage,
 int fixedRow,
 int minimumOffset,
 int maximumOffset,
 int& bestOffset
 )
void MatchFrames(
 EROIC24* sourceImage,
 int fixedRow,
 int minimumOffset,
 int maximumOffset,
 int& bestOffset
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

fixedRow

Index of the line used for comparison. Line fixedRow remains in place and is compared with line (fixedRow + 1), shifted by some amount.

minimumOffset

Minimum value of the allowed offset (positive to the right).

maximumOffset

Maximum value of the allowed offset (positive to the right).

bestOffset

Estimated shift amplitude.

Remarks

These lines should be chosen such that they cross some edges or non-uniform areas.
When an image is interlaced, the two frames (even and odd lines) are not recorded at the same time. If there is movement in the scene, a visible artifact can result (the edges of objects exhibit a "comb" effect).
When the movement is uniform and horizontal (objects on a conveyor belt), one cure to this problem is to shift one of the frames horizontally with respect to the other frame (using EasyImage::RealignFrame). The amplitude of the shift can be estimated automatically.

EasyImage::Median

Applies a median filter to an image (median of the gray values in a neighborhood). Kernel may be of an arbitrary size except for EROIBW1 where it is always 3*3.

Namespace: Euresys::Open_eVision

[C++]

```
void Median(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void Median(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void Median(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void Median(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void Median(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void Median(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void Median(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half height of the kernel minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

EasyImage::ModulusImage

Prepares a lookup-table image for use for gradient magnitude computation.

Namespace: Euresys::Open_eVision

[C++]

```
void ModulusImage(
 EImageBW8* destinationImage,
 float gain
 )
```

Parameters

destinationImage

Pointer to the destination image.

gain

Gain value to be applied to the modulus. 1 saturates; 1/Sqrt(2) does not.

Remarks

The modulus of the gradient argument can be adjusted to avoid saturation. EasyImage::ModulusImage sets a lookup-table image for use with function EasyImage::GradientScalar, ready to compute the modulus of the gradient in the source image, i.e. its amplitude (as defined by the Euclidian norm). The argument will be returned as a value in range 0..255 suitable for storage in an EImageBW8 or as a value in range 0..65535 suitable for storage in an EImageBW16. A gain coefficient can be adjusted to avoid saturation (gain = 1 saturates gradient amplitudes larger than 255 in the EBW8 case and 65535 in the EBW16 case; gain = 1/Sqrt(2) never saturates).

EasyImage::MorphoGradientBox

Computes the morphological gradient of an image using a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void MorphoGradientBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void MorphoGradientBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void MorphoGradientBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void MorphoGradientBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void MorphoGradientBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void MorphoGradientBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
void MorphoGradientBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element.
The kernel size is a pair of odd numbers; they must be halved before they are passed. For instance, a 3x5 kernel is passed as 1x2.

EasyImage::MorphoGradientDisk

Computes the morphological gradient of an image using a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void MorphoGradientDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void MorphoGradientDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void MorphoGradientDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void MorphoGradientDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void MorphoGradientDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void MorphoGradientDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void MorphoGradientDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

region

Region to apply the function on.

Remarks

The morphological gradient is the difference between the dilation and the erosion of the image, using the same structuring element.

EasyImage::Normalize

Normalizes an image, i.e. applies a linear transform to the gray levels so that their average and standard deviation are imposed.

Namespace: Euresys::Open_eVision

[C++]

```
void Normalize(
 const EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 float imposedAverage,
 float imposedStandardDeviation
 )
void Normalize(
 const EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float imposedAverage,
 float imposedStandardDeviation
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

imposedAverage

Imposed average.

imposedStandardDeviation

Imposed standard deviation.

EasyImage::Offset

Transforms an image, applying a gain and offset to all pixels.

Namespace: Euresys::Open_eVision

[C++]

```
void Offset(
 const EROIC24* sourceImage,
 EROIC24* destinationImage,
 EColor Offset
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Offset

Constant offset. By default (argument omitted) 0, i.e. no change.

Remarks

The gain should remain close to 1, and allows contrast adjustment of the image.

The offset can be positive or negative, and allows to adjust the image intensity. The resulting values are always saturated to range [0..255].

For color images, the separate gain and offset values are specified as triple of values stored in a EColor. The default values leave the image unchanged.

Internally, the computations are achieved through fixed-point arithmetic with 5 bits of precision for the fractional part. This can result in loss of precision with small gains.

EasyImage::OpenBox

Performs an opening on an image (erosion followed by dilation) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void OpenBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

```
void OpenBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void OpenBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void OpenBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void OpenBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void OpenBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void OpenBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half of the box width minus one, as shown on the picture below (by default, halfOfKernelWidth =1; 0 is allowed).

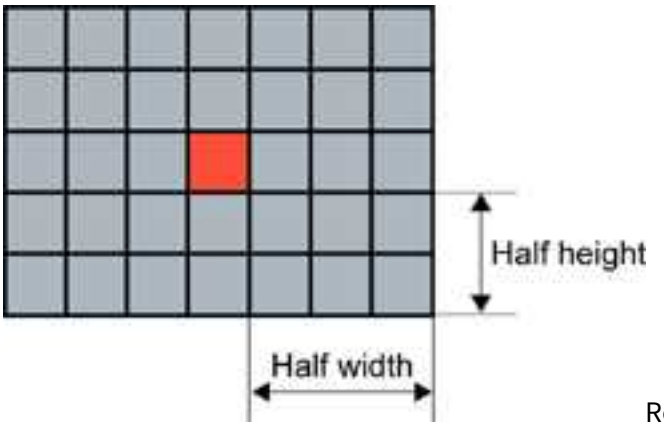
halfOfKernelHeight

Half of the box height minus one, as shown on the picture below (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

Remarks



Rectangular kernel of half width = 3 and half height = 2

EasyImage::OpenDisk

Performs an opening on an image (erosion followed by dilation) on a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void OpenDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void OpenDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void OpenDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void OpenDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void OpenDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void OpenDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void OpenDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. If NULL (default), this operation is destructive (i.e. applied to the source image).

halfOfKernelWidth

Half width of the kernel minus one, as shown on the picture below (by default, halfOfKernelWidth = 1; 0 is allowed).

region

Region to apply the function on.

EasyImage::Oper

Applies the desired arithmetic or logic pixel-wise operator between two images or constants.

Namespace: Euresys::Open_eVision

[C++]

```
void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EBW8 constant,
 EROIBW8* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EBW16 constant,
 EROIBW16* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EC24 constant,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EBW8 constant,
 const EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EBW16 constant,
 const EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EC24 constant,
 const EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage,
 EBW8 constant,
 EROIBW8* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW16* sourceImage,
 EBW16 constant,
 EROIBW16* destinationImage
 )
```

```
void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIC24* sourceImage,
 EC24 constant,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage0,
 const EROIBW8* sourceImage1,
 EROIBW8* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW16* sourceImage0,
 const EROIBW16* sourceImage1,
 EROIBW16* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIC24* sourceImage0,
 const EROIC24* sourceImage1,
 EROIC24* destinationImage
 )
```

```
void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage0,
 const EROIBW8* sourceImage1,
 EROIBW16* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW16* sourceImage0,
 const EROIBW8* sourceImage1,
 EROIBW16* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage0,
 const EROIBW8* sourceImage1,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW8* sourceImage0,
 const EROIC24* sourceImage1,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIC24* sourceImage0,
 const EROIBW8* sourceImage1,
 EROIC24* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 EBW1 constant,
 EROIBW1* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW1* sourceImage,
 EROIBW1* destinationImage
 )

void Oper(
 Euresys::Open_eVision::EArithmeticLogicOperation operation,
 const EROIBW1* sourceImage0,
 const EROIBW1* sourceImage1,
 EROIBW1* destinationImage
 )
```

Parameters

operation

Arithmetic or logic operator, as defined by EArithmeticLogicOperation.

constant

Gray-level or color constant.

destinationImage

Pointer to the destination image/ROI.

sourceImage

Pointer to the second source image/ROI (right operand).

sourceImage0

Pointer to the first source image/ROI (left operand).

sourceImage1

Pointer to the second source image/ROI (right operand).

Remarks

The source and destination images may be the same.
When the source operands are two color images/constants, the components are combined pair-wise. The result is a color image.
When the source operands are a color image and a gray-level image, each color component is combined with the gray-level component. The result is a color image.

EasyImage::Overlay

Overlays an image on the top of a color image, at a given position.

Namespace: Euresys::Open_eVision

[C++]

```
void Overlay(
 const EROIC24* sourceImage,
 EROIC16* destinationImage,
 float panX,
 float panY,
 EC24 referenceValue
 )
void Overlay(
 const EROIC24* sourceImage,
 EROIC15* destinationImage,
 float panX,
 float panY,
 EC24 referenceValue
 )
```

```
void Overlay(
 const EROIC24* sourceImage,
 EROIC24* destinationImage,
 float panX,
 float panY,
 EC24 referenceValue
 )
void Overlay(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EROIC15* destinationImage,
 float panX,
 float panY
 )
void Overlay(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EROIC16* destinationImage,
 float panX,
 float panY
 )
void Overlay(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EROIC24* destinationImage,
 float panX,
 float panY
 )
void Overlay(
 const EROIBW8* sourceImage,
 EROIC24* overlay,
 EROIC24* destinationImage,
 float panX,
 float panY,
 EC24 referenceValue
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

referenceValue

Reference color.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

overlay

When a BW8 source image is specified, pointer to the overlay image/ROI.

Remarks

If a color image is provided as the source image, all the pixels of this image are copied to the destination image, but the ones that equal the reference color.
If a BW8 image is provided as the source image, all the pixels of the overlay image are copied to the destination image, but the ones that equal the reference color, the latter being replaced by the content of the source image.

EasyImage::GetOverlayColor

EasyImage::SetOverlayColor

Gets/Sets the color of the overlay in the destination image when a BW8 Image is used as overlay source image in functions.

Namespace: Euresys::Open_eVision

[C++]

```
static EC24 GetOverlayColor()

static void SetOverlayColor(EC24 color)
```

Remarks

Note. When a C24 Image is used as overlay source image, the color of the overlay in destination image is the same as the one in the overlay source image, thus allowing multi colored overlays.

EasyImage::PathToImage

Given a path described by coordinates in a path vector, copies the pixel values from the path vector to the corresponding image pixels.

Namespace: Euresys::Open_eVision

[C++]

```
void PathToImage(
 EROIBW8* sourceImage,
 EBW8PathVector* path
 )
```

```
void PathToImage(
 EROIBW16* sourceImage,
 EBW16PathVector* path
 )

void PathToImage(
 EROIC24* sourceImage,
 EC24PathVector* path
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

path

Pointer to the destination vector.

EasyImage::PixelAverage

Computes the average pixel value in a gray-level or color image.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelAverage(
 const EROIBW8* sourceImage,
 float& average
 )

void PixelAverage(
 const EROIBW16* sourceImage,
 float& average
 )

void PixelAverage(
 const EROIC24* sourceImage,
 float& average0,
 float& average1,
 float& average2
 )

void PixelAverage(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& average
 )

void PixelAverage(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& average
 )
```

```
void PixelAverage(
 const EROIC24* sourceImage,
 const ERegion& region,
 float& average0,
 float& average1,
 float& average2
 )
void PixelAverage(
 const EROIBW8* sourceImage,
 const EROIBW8* inputMask,
 float& average
 )
void PixelAverage(
 const EROIBW16* sourceImage,
 const EROIBW8* inputMask,
 float& average
 )
void PixelAverage(
 const EROIC24* sourceImage,
 const EROIBW8* inputMask,
 float& average0,
 float& average1,
 float& average2
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

average
Reference to the average gray-level value.

average0
Reference to the average values for the first color channel.

average1
Reference to the average values for the second color channel.

average2
Reference to the average values for the third color channel.

region
Pointer to a region to apply the function only on a particular region in the image.

inputMask
Pointer to the mask, which allows functions to be applied on a particular region in the image.

EasyImage::PixelCompare

Counts the number of pixels differing between two images.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 PixelCompare(
 const EROIBW8* sourceImage0,
 const EROIBW8* sourceImage1
 )

OEV_UINT32 PixelCompare(
 const EROIBW16* sourceImage0,
 const EROIBW16* sourceImage1
 )

OEV_UINT32 PixelCompare(
 const EROIC24* sourceImage0,
 const EROIC24* sourceImage1
 )

OEV_UINT32 PixelCompare(
 const EROIBW8* sourceImage0,
 const ERegion& region,
 const EROIBW8* sourceImage1
 )

OEV_UINT32 PixelCompare(
 const EROIBW16* sourceImage0,
 const ERegion& region,
 const EROIBW16* sourceImage1
 )

OEV_UINT32 PixelCompare(
 const EROIC24* sourceImage0,
 const ERegion& region,
 const EROIC24* sourceImage1
 )

OEV_UINT32 PixelCompare(
 const EROIBW8* sourceImage0,
 const EROIBW8* sourceImage1,
 const EROIBW8* mask
 )

OEV_UINT32 PixelCompare(
 const EROIBW16* sourceImage0,
 const EROIBW16* sourceImage1,
 const EROIBW8* mask
 )

OEV_UINT32 PixelCompare(
 const EROIC24* sourceImage0,
 const EROIC24* sourceImage1,
 const EROIBW8* mask
 )

OEV_UINT32 PixelCompare(
 const EROIBW1* sourceImage0,
 const EROIBW1* sourceImage1
 )
```

Parameters

sourceImage0

Pointer to the first source image/ROI.

sourceImage1

Pointer to the second source image/ROI.

region

Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::PixelCount

Counts the pixels in the three value classes separated by two thresholds.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelCount(
 const EROIBW8* sourceImage,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 int& numberOfPixelsBelowThreshold,
 int& numberOfPixelsBetweenThresholds,
 int& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW16* sourceImage,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 int& numberOfPixelsBelowThreshold,
 int& numberOfPixelsBetweenThresholds,
 int& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 int& numberOfPixelsBelowThreshold,
 int& numberOfPixelsBetweenThresholds,
 int& numberOfPixelsAboveThreshold
 )
```

```
void PixelCount(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 int& numberOfPixelsBelowThreshold,
 int& numberOfPixelsBetweenThresholds,
 int& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 OEV_UINT64& numberOfPixelsBelowThreshold,
 OEV_UINT64& numberOfPixelsBetweenThresholds,
 OEV_UINT64& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 OEV_UINT64& numberOfPixelsBelowThreshold,
 OEV_UINT64& numberOfPixelsBetweenThresholds,
 OEV_UINT64& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW8* sourceImage,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 OEV_UINT64& numberOfPixelsBelowThreshold,
 OEV_UINT64& numberOfPixelsBetweenThresholds,
 OEV_UINT64& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW16* sourceImage,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 OEV_UINT64& numberOfPixelsBelowThreshold,
 OEV_UINT64& numberOfPixelsBetweenThresholds,
 OEV_UINT64& numberOfPixelsAboveThreshold
 )
```

```
void PixelCount(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 int& numberOfPixelsBelowThreshold,
 int& numberOfPixelsBetweenThresholds,
 int& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 int& numberOfPixelsBelowThreshold,
 int& numberOfPixelsBetweenThresholds,
 int& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8 lowThreshold,
 EBW8 highThreshold,
 OEV_UINT64& numberOfPixelsBelowThreshold,
 OEV_UINT64& numberOfPixelsBetweenThresholds,
 OEV_UINT64& numberOfPixelsAboveThreshold
 )

void PixelCount(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16 lowThreshold,
 EBW16 highThreshold,
 OEV_UINT64& numberOfPixelsBelowThreshold,
 OEV_UINT64& numberOfPixelsBetweenThresholds,
 OEV_UINT64& numberOfPixelsAboveThreshold
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

lowThreshold

Inferior threshold.

highThreshold

Superior threshold.

numberOfPixelsBelowThreshold

Reference to the count of pixels strictly below the inferior threshold.

numberOfPixelsBetweenThresholds

Reference to the count of pixels above or equal to the inferior threshold, and strictly below the superior threshold.

numberOfPixelsAboveThreshold

Reference to the count of pixels above or equal to the superior threshold.

region

Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::PixelMax

Computes the maximum gray-level value in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelMax(
 const EROIBW8* sourceImage,
 EBW8& maximumValue
 )

void PixelMax(
 const EROIBW16* sourceImage,
 EBW16& maximumValue
 )

void PixelMax(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8& maximumValue
 )

void PixelMax(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16& maximumValue
 )

void PixelMax(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8& maximumValue
 )

void PixelMax(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16& maximumValue
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumValue

Reference to the maximum value.

region

Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::PixelMaxBW16

Computes the maximum gray-level value in an image.

Namespace: Euresys::Open_eVision

```
[C++]
void PixelMaxBW16(
 const EROIBW16* sourceImage,
 EBW16& maximumValue
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumValue

Reference to the maximum value.

EasyImage::PixelMaxBW8

Computes the maximum gray-level value in an image.

Namespace: Euresys::Open_eVision

```
[C++]
void PixelMaxBW8(
 const EROIBW8* sourceImage,
 EBW8& maximumValue
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumValue

Reference to the maximum value.

EasyImage::PixelMin

Computes the minimum gray-level value in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelMin(
 const EROIBW8* sourceImage,
 EBW8& minimumValue
 )

void PixelMin(
 const EROIBW16* sourceImage,
 EBW16& minimumValue
 )

void PixelMin(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8& minimumValue
 )

void PixelMin(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16& minimumValue
 )

void PixelMin(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8& minimumValue
 )

void PixelMin(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16& minimumValue
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

minimumValue

Reference to the minimum value.

region

Region to apply the function on.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::PixelMinBW16

Computes the minimum gray-level value in an image.

Namespace: Euresys::Open_eVision

```
[C++]
void PixelMinBW16(
 const EROIBW16* sourceImage,
 EBW16& minimumValue
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

minimumValue
Reference to the minimum value.

EasyImage::PixelMinBW8

Computes the minimum gray-level value in an image.

Namespace: Euresys::Open_eVision

```
[C++]
void PixelMinBW8(
 const EROIBW8* sourceImage,
 EBW8& minimumValue
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

minimumValue
Reference to the minimum value.

EasyImage::PixelStat

Computes the minimum, maximum and average gray-level values in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelStat(
 const EROIBW8* sourceImage,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )

void PixelStat(
 const EROIBW16* sourceImage,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )

void PixelStat(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )

void PixelStat(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )

void PixelStat(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )

void PixelStat(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

minimumValue
Reference to the minimum value.

maximumValue
Reference to the maximum value.

average
Reference to the average value.

region
Pointer to a region to apply the function only on a particular region in the image.

mask
Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::PixelStatBW16

Computes the minimum, maximum and average gray-level values in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelStatBW16(
 const EROIBW16* sourceImage,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

minimumValue
Reference to the minimum value.

maximumValue
Reference to the maximum value.

average
Reference to the average value.

EasyImage::PixelStatBW8

Computes the minimum, maximum and average gray-level values in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelStatBW8(
 const EROIBW8* sourceImage,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

minimumValue
Reference to the minimum value.

maximumValue
Reference to the maximum value.

average
Reference to the average value.

EasyImage::PixelStdDev

Computes the average gray-level or color value in an image, the standard deviation of the color components, and the correlation between the color components (in the case of color images).

Namespace: Euresys::Open_eVision

[C++]

```
void PixelStdDev(
 const EROIBW8* sourceImage,
 float& standardDeviation,
 float& mean
 )

void PixelStdDev(
 const EROIBW16* sourceImage,
 float& standardDeviation,
 float& mean
 )
```

```
void PixelStdDev(
 const EROIC24* sourceImage,
 float& standardDeviation0,
 float& standardDeviation1,
 float& standardDeviation2,
 float& correlation01,
 float& correlation12,
 float& correlation20,
 float& mean0,
 float& mean1,
 float& mean2
 )

void PixelStdDev(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& standardDeviation,
 float& mean
 )

void PixelStdDev(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& standardDeviation,
 float& mean
 )

void PixelStdDev(
 const EROIC24* sourceImage,
 const ERegion& region,
 float& standardDeviation0,
 float& standardDeviation1,
 float& standardDeviation2,
 float& correlation01,
 float& correlation12,
 float& correlation20,
 float& mean0,
 float& mean1,
 float& mean2
 )

void PixelStdDev(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 float& standardDeviation,
 float& mean
 )

void PixelStdDev(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 float& standardDeviation,
 float& mean
 )
```

```
void PixelStdDev(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 float& standardDeviation0,
 float& standardDeviation1,
 float& standardDeviation2,
 float& correlation01,
 float& correlation12,
 float& correlation20,
 float& mean0,
 float& mean1,
 float& mean2
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

standardDeviation

Reference to a variable in which the standard deviation of the pixel values is to be stored (for gray-level images).

mean

Reference to a variable in which the average value of the pixels is to be stored (for gray-level images).

standardDeviation0

Reference to a variable in which the standard deviation of the values of the first color component is to be stored (for color images).

standardDeviation1

Reference to a variable in which the standard deviation of the values of the second color component is to be stored (for color images).

standardDeviation2

Reference to a variable in which the standard deviation of the values of the third color component is to be stored (for color images).

correlation01

Reference to a variable in which the correlation between the values of the first color component and the second color component is to be stored (for color images).

correlation12

Reference to a variable in which the correlation between the values of the second color component and the third color component is to be stored (for color images).

correlation20

-

mean0

Reference to a variable in which the average value of the first color component is to be stored (for color images).

mean1

Reference to a variable in which the average value of the second color component is to be stored (for color images).

mean2

Reference to a variable in which the average value of the third color component is to be stored (for color images).

region

Pointer to a region to apply the function only on a particular region in the image.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

The variance can be obtained from the standard deviation by squaring it.

EasyImage::PixelVariance

For a gray-level image, computes the mean and variance of the pixel values.

Namespace: Euresys::Open_eVision

[C++]

```
void PixelVariance(
 const EROIBW8* sourceImage,
 float& variance,
 float& mean
 )

void PixelVariance(
 const EROIBW16* sourceImage,
 float& variance,
 float& mean
 )

void PixelVariance(
 const EROIC24* sourceImage,
 float& variance0,
 float& variance1,
 float& variance2,
 float& covariance01,
 float& covariance12,
 float& covariance20,
 float& mean0,
 float& mean1,
 float& mean2
 )

void PixelVariance(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& variance,
 float& mean
 )
```

```
void PixelVariance(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& variance,
 float& mean
 )

void PixelVariance(
 const EROIC24* sourceImage,
 const ERegion& region,
 float& variance0,
 float& variance1,
 float& variance2,
 float& covariance01,
 float& covariance12,
 float& covariance20,
 float& mean0,
 float& mean1,
 float& mean2
 )

void PixelVariance(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 float& variance,
 float& mean
 )

void PixelVariance(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 float& variance,
 float& mean
 )

void PixelVariance(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 float& variance0,
 float& variance1,
 float& variance2,
 float& covariance01,
 float& covariance12,
 float& covariance20,
 float& mean0,
 float& mean1,
 float& mean2
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

variance
Reference to the covariances of the pairs of pixel component values.

mean
Reference to the mean pixel component values.

variance0
Reference to the covariances of the pairs of pixel component values.

variance1
Reference to the covariances of the pairs of pixel component values.

variance2
Reference to the covariances of the pairs of pixel component values.

covariance01
Reference to the covariances of the pairs of pixel component values.

covariance12
Reference to the covariances of the pairs of pixel component values.

covariance20
Reference to the covariances of the pairs of pixel component values.

mean0
Reference to the mean pixel component values.

mean1
Reference to the mean pixel component values.

mean2
Reference to the mean pixel component values.

region
Pointer to a region to apply the function only on a particular region in the image.

mask
Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

For a color image, computes the means of the three pixel color components, the variances of the components and the covariances between pairs of components.

EasyImage::ProfileDerivative

Computes the first derivative of a profile extracted from a gray-level image.

Namespace: Euresys::Open_eVision

[C++]

```
void ProfileDerivative(
 EBW8Vector* sourceVector,
 EBW8Vector* destinationVector
 )

void ProfileDerivative(
 EBW16Vector* sourceVector,
 EBW16Vector* destinationVector
 )
```

Parameters

sourceVector

Pointer to the source vector.

destinationVector

Pointer to the destination vector.

Remarks

Taking the derivative transforms transitions (edges) into peaks.

Note. Since the EBW8 data type only handles unsigned values, the derivative is shifted up by 128. Values under [above] 128 correspond to negative [positive] derivative (decreasing [increasing] slope).

EasyImage::ProjectOnAColumn

Projects an image horizontally onto a column.

Namespace: Euresys::Open_eVision

[C++]

```
void ProjectOnAColumn(
 const EROIBW8* sourceImage,
 EBW32Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW16* sourceImage,
 EBW32Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW8* sourceImage,
 EBW8Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW16* sourceImage,
 EBW16Vector* destinationVector
 )
```

```
void ProjectOnAColumn(
 const EROIC24* sourceImage,
 EC24Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW32Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW32Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW8* sourceImage,
 EROIBW8* mask,
 EBW8Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIBW16* sourceImage,
 EROIBW8* mask,
 EBW16Vector* destinationVector
 )

void ProjectOnAColumn(
 const EROIC24* sourceImage,
 EROIBW8* mask,
 EC24Vector* destinationVector
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationVector

Pointer to the destination vector.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

Pixel gray/color levels are added when projecting into an EBW32Vector. When projecting into an EBW8Vector/EBW16Vector/EC24Vector, pixel values are averaged, instead.

EasyImage::ProjectOnARow

Projects an image vertically onto a row.

Namespace: Euresys::Open_eVision

[C++]

```
void ProjectOnARow(
 const EROIBW8* sourceImage,
 EBW32Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW16* sourceImage,
 EBW32Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW8* sourceImage,
 EBW8Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW16* sourceImage,
 EBW16Vector* destinationVector
 )

void ProjectOnARow(
 const EROIC24* sourceImage,
 EC24Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW32Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW32Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 EBW8Vector* destinationVector
 )

void ProjectOnARow(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 EBW16Vector* destinationVector
 )

void ProjectOnARow(
 const EROIC24* sourceImage,
 const EROIBW8* mask,
 EC24Vector* destinationVector
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationVector

Pointer to the destination vector.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

Remarks

Pixel gray/color levels are added when projecting into an EBW32Vector. When projecting into an EBW8Vector/EBW16Vector/EC24Vector, pixel values are averaged, instead.

EasyImage::RealignFrame

Shifts one frame of the image horizontally.

Namespace: Euresys::Open_eVision

[C++]

```
void RealignFrame(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 int offset,
 OEV_UINT32 fixedRow
 )

void RealignFrame(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 int offset,
 OEV_UINT32 fixedRow
 )

void RealignFrame(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 int offset,
 OEV_UINT32 fixedRow
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

offset

Indicates the number of pixels by which to shift (positive to the right).

fixedRow

Specifies which frame remains unchanged (the frame made up of all lines of the same parity as fixedRow; by default, fixedRow = 0).

Remarks

The same image should be used as source and destination. If the destination image differs from the source image, only the shifted rows are copied. To use a different destination image, the source image must be copied first in the destination image object.
When an image is interlaced, the two frames (even and odd lines) are not recorded at the same time. If there is movement in the scene, a visible artifact can result (the edges of objects exhibit a "comb" effect).
When the movement is uniform and horizontal (objects on a conveyor belt), one cure to this problem is to shift one of the frames horizontally with respect to the other frame. The amplitude of the shift can be estimated automatically (using EasyImage::MatchFrames).

EasyImage::RebuildFrame

Rebuilds one frame of the image by interpolation between the lines of the other frame.

Namespace: Euresys::Open_eVision

[C++]

```
void RebuildFrame(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 fixedRow
 )

void RebuildFrame(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 fixedRow
 )

void RebuildFrame(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 fixedRow
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

fixedRow

Specifies which frame remains unchanged (the frame made up of all lines of the same parity as fixedRow; by default, fixedRow = 0).

Remarks

The same image should be used as source and destination. If the destination image differs from the source image, only the shifted rows are copied. To use a different destination image, the source image must be copied first in the destination image object. When an image is interlaced, the two frames (even and odd lines) are not recorded at the same time. If there is movement in the scene, a visible artifact can result (the edges of objects exhibit a "comb" effect). One cure to this problem is to replace one of the frames by linearly interpolating between the lines of the other frame.

EasyImage::RecursiveAverage

Applies stronger noise reduction to small variations and conversely.

Namespace: Euresys::Open_eVision

[C++]

```
void RecursiveAverage(
 const EROIBW8* sourceImage,
 const EROIBW16* store,
 EROIBW8* destinationImage,
 const EBW16Vector* lookupTable
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

store

Pointer to a 16-bit work image.

destinationImage

Pointer to the destination image/ROI.

lookupTable

Pointer to the LUT vector generated by a call to EasyImage::SetRecursiveAverageLUT.

Remarks

Recursive averaging is a well known process for noise reduction by temporal integration. The principle is to continuously update a noise-free image by blending it, using a linear combination, with the raw, noisy, live image stream.
Algorithmically, this amounts to apply the following recurrence: where a is a mixture coefficient. The value of this coefficient can be adjusted so that a prescribed noise reduction ratio is achieved. This procedure is effective when applied to still images, but generates a trailing effect on moving objects because of the transient behavior of the filter. The larger the noise reduction ratio, the heavier the trailing effect. To work around this, a non-linearity can be introduced in the blending process: small gray-level values variations between successive images are usually caused by noise, while large variations correspond to changes in the signal itself (camera displacement or object movements). Function EasyImage::RecursiveAverage uses this observation and applies stronger noise reduction to small variations and conversely. This way, noise is better reduced in still areas and trailing is avoided in moving areas.
For optimal performance, the non-linearity must be pre-computed once for all using function EasyImage::SetRecursiveAverageLUT.

Note. Before the first call to the EasyImage::RecursiveAverage method, the 16-bit work image *must* be cleared (all pixel values set to zero).

EasyImage::Register

Registers an image by realigning one, two or three pivot points to reference positions.

Namespace: Euresys::Open_eVision

[C++]

```
void Register(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 int interpolationBits
 )
```

```
void Register(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 int interpolationBits
 )

void Register(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 int interpolationBits
 )

void Register(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float sourceImagePivot1X,
 float sourceImagePivot1Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 float destinationImagePivot1X,
 float destinationImagePivot1Y,
 int interpolationBits,
 bool resize
 )

void Register(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float sourceImagePivot1X,
 float sourceImagePivot1Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 float destinationImagePivot1X,
 float destinationImagePivot1Y,
 int interpolationBits,
 bool resize
 )
```

```
void Register(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float sourceImagePivot1X,
 float sourceImagePivot1Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 float destinationImagePivot1X,
 float destinationImagePivot1Y,
 int interpolationBits,
 bool resize
 )

void Register(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float sourceImagePivot1X,
 float sourceImagePivot1Y,
 float sourceImagePivot2X,
 float sourceImagePivot2Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 float destinationImagePivot1X,
 float destinationImagePivot1Y,
 float destinationImagePivot2X,
 float destinationImagePivot2Y,
 int interpolationBits
 )

void Register(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float sourceImagePivot1X,
 float sourceImagePivot1Y,
 float sourceImagePivot2X,
 float sourceImagePivot2Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 float destinationImagePivot1X,
 float destinationImagePivot1Y,
 float destinationImagePivot2X,
 float destinationImagePivot2Y,
 int interpolationBits
 )
```

```
void Register(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 float sourceImagePivot0X,
 float sourceImagePivot0Y,
 float sourceImagePivot1X,
 float sourceImagePivot1Y,
 float sourceImagePivot2X,
 float sourceImagePivot2Y,
 float destinationImagePivot0X,
 float destinationImagePivot0Y,
 float destinationImagePivot1X,
 float destinationImagePivot1Y,
 float destinationImagePivot2X,
 float destinationImagePivot2Y,
 int interpolationBits
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. May not be the same as the source image.

sourceImagePivot0X

First pivot point abscissa in the source image.

sourceImagePivot0Y

First pivot point ordinate in the source image.

destinationImagePivot0X

First pivot point abscissa in the destination image.

destinationImagePivot0Y

First pivot point ordinate in the destination image.

interpolationBits

Number of bits of accuracy for interpolation. Allowed values are 0 (no interpolation, nearest neighbor), 4 (linear interpolation) or 8 (cubic interpolation).

sourceImagePivot1X

Second pivot point abscissa in the source image.

sourceImagePivot1Y

Second pivot point ordinate in the source image.

destinationImagePivot1X

Second pivot point abscissa in the destination image.

destinationImagePivot1Y

Second pivot point ordinate in the destination image.

resize

true if scaling is allowed.

sourceImagePivot2X

Third pivot point abscissa in the source image.

sourceImagePivot2Y

Third pivot point ordinate in the source image.

destinationImagePivot2X

Third pivot point abscissa in the destination image.

destinationImagePivot2Y

Third pivot point ordinate in the destination image.

Remarks

Out-of-image-bounds pixels are black.
Registration is the process of realigning two misaligned images so that point-to-point comparisons are possible. The simplest way to achieve this is to accurately locate features in both images (landmarks or pivots), using pattern matching, point measurement or whatever other technique, and realign one of the images so that the landmarks are superimposed.
* When a single pivot point is used, the registration transform is a simple translation. If interpolation bits are used, sub-pixel translation is achieved.
* When two pivot points are used, the registration is a combination of translation, rotation and optionally scaling. If scaling is not allowed, the second pivot point will not be matched exactly in general. Anyway, for most applications scaling should not be used unless it corresponds to a change of lens magnification or viewing distance.
* When three pivot points are used, the registration is a combination of translation, rotation, shearing correction and optionally scaling. The so-called shear effect can arise when acquiring images with a misaligned line-scan camera.
To achieve good accuracy, the pivot points should be chosen as far apart as possible.

EasyImage::RmsNoise

Computes the root-mean-square amplitude of noise, by comparing a given image to a reference image.

Namespace: Euresys::Open_eVision

[C++]

```
float RmsNoise(
 const EROIBW8* sourceImage,
 const EROIBW8* referenceImage,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float RmsNoise(
 const EROIBW16* sourceImage,
 const EROIBW16* referenceImage,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float RmsNoise(
 const EROIC24* sourceImage,
 const EROIC24* referenceImage,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )
```

```
float RmsNoise(
 const EROIBW8* sourceImage,
 const EROIBW8* referenceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )
float RmsNoise(
 const EROIBW16* sourceImage,
 const EROIBW16* referenceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )
float RmsNoise(
 const EROIC24* sourceImage,
 const EROIC24* referenceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

referenceImage

Pointer to the reference image/ROI.

referenceNoise

Specifies how the reference image is affected by noise, as defined by EReferenceNoise.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

count

-

Remarks

The reference image can be noiseless (obtained by suppressing the source of noise), or affected by a noise of the same distribution as the given image.

EasyImage::Rotate

Rotate an image by an increment of a quarter of a turn (right angle).

Namespace: Euresys::Open_eVision

[C++]

```
void Rotate(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 Euresys::Open_eVision::ERotationRightAngles rightAngle
 )
```

```
void Rotate(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 Euresys::Open_eVision::ERotationRightAngles rightAngle
 )
void Rotate(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 Euresys::Open_eVision::ERotationRightAngles rightAngle
 )
void Rotate(
 EROIBW8* sourceImage,
 Euresys::Open_eVision::ERotationRightAngles rightAngle
 )
void Rotate(
 EROIBW16* sourceImage,
 Euresys::Open_eVision::ERotationRightAngles rightAngle
 )
void Rotate(
 EROIC24* sourceImage,
 Euresys::Open_eVision::ERotationRightAngles rightAngle
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. May not be the same as the source image.

rightAngle

Right angle of rotation (90, 180 or 270 degrees).

Remarks

Destination image/roi size should be the compatible with the source image/roi size with the rotation.

EasyImage::ScaleRotate

Re-scales an image by an arbitrary factor and/or rotates it by an arbitrary angle.

Namespace: Euresys::Open_eVision

[C++]

```
void ScaleRotate(
 EROIBW8* sourceImage,
 float sourceImagePivotX,
 float sourceImagePivotY,
 float destinationImagePivotX,
 float destinationImagePivotY,
 float scaleX,
 float scaleY,
 float rotation,
 EROIBW8* destinationImage,
 int interpolationBits
 )

void ScaleRotate(
 EROIC24* sourceImage,
 float sourceImagePivotX,
 float sourceImagePivotY,
 float destinationImagePivotX,
 float destinationImagePivotY,
 float scaleX,
 float scaleY,
 float rotation,
 EROIC24* destinationImage,
 int interpolationBits
 )

void ScaleRotate(
 EROIBW16* sourceImage,
 float sourceImagePivotX,
 float sourceImagePivotY,
 float destinationImagePivotX,
 float destinationImagePivotY,
 float scaleX,
 float scaleY,
 float rotation,
 EROIBW16* destinationImage,
 int interpolationBits
 )

void ScaleRotate(
 EROIBW8* sourceImage,
 const ERegion& region,
 float sourceImagePivotX,
 float sourceImagePivotY,
 float destinationImagePivotX,
 float destinationImagePivotY,
 float scaleX,
 float scaleY,
 float rotation,
 EROIBW8* destinationImage,
 int interpolationBits
 )
```

```
void ScaleRotate(
 EROIC24* sourceImage,
 const ERegion& region,
 float sourceImagePivotX,
 float sourceImagePivotY,
 float destinationImagePivotX,
 float destinationImagePivotY,
 float scaleX,
 float scaleY,
 float rotation,
 EROIC24* destinationImage,
 int interpolationBits
 )
void ScaleRotate(
 EROIBW16* sourceImage,
 const ERegion& region,
 float sourceImagePivotX,
 float sourceImagePivotY,
 float destinationImagePivotX,
 float destinationImagePivotY,
 float scaleX,
 float scaleY,
 float rotation,
 EROIBW16* destinationImage,
 int interpolationBits
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

sourceImagePivotX
Pivot point abscissa in the source image.

sourceImagePivotY
Pivot point ordinate in the source image.

destinationImagePivotX
Pivot point abscissa in the destination image.

destinationImagePivotY
Pivot point ordinate in the destination image.

scaleX
Scale factor for the abscissas. Its value must be different than 0.0.

scaleY
Scale factor for the ordinates. Its value must be different than 0.0.

rotation
Anti-clockwise rotation angle, using the current angle unit.

destinationImage
Pointer to the destination image/ROI. May not be the same as the source image.

interpolationBits

Number of bits of accuracy for interpolation. Allowed values are 0 (no interpolation, nearest neighbor), 4 (linear interpolation) or 8 (cubic interpolation).

region

Region to apply the function on.

Remarks

For resampling, the nearest neighbor rule or bilinear interpolation with 4 or 8 bits of accuracy is used.
The pivot point is a given point in the source image which is mapped to a given point in the destination image. Rotation and scaling are done around the pivot point. The pivot point reference coordinates are based on the 'Pixel Coordinate System', meaning that the origin (0, 0) is the center of the top left pixel of the image.
Out-of-image-bounds pixels are black.
When using a region, only the pixels contained in this region will be taken into account.

EasyImage::SetCircleWarp

Prepares suitable warp images for use with function EasyImage::Warp to unwarp a circular ring-wedge shape into a straight rectangle. This is a cartesian to polar image conversion function.
See also EasyImage::SetInvCircleWarp.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCircleWarp(
 float centerX,
 float centerY,
 int numberOfRadialSampledPoints,
 float minimumRadius,
 float maximumRadius,
 int numberOfTangentSampledPoints,
 float minimumAngle,
 float maximumAngle,
 EImageBW16* warpImageX,
 EImageBW16* warpImageY
 )
```

Parameters

centerX

Abscissa of the ring-wedge center.

centerY

Ordinate of the ring-wedge center.

numberOfRadialSampledPoints

Number of points to be sampled in the radial direction (the height of the destination polar image).

minimumRadius

Starting radius of the ring-wedge shape.

maximumRadius

Ending radius of the ring-wedge shape.

numberOfTangentSampledPoints

Number of points to be sampled in the tangent direction (the width of the destination polar image).

minimumAngle

Starting angle of the ring-wedge shape.

maximumAngle

Ending angle of the ring-wedge shape.

warpImageX

Destination warp image for the abscissas.

warpImageY

Destination warp image for the ordinates.

Remarks

Typical use is unwarping of a text printed around a circle.

Note. A ring-wedge is delimited by two concentric circles and two straight lines passing through the center.

EasyImage::SetFrame

Replaces the frame of given parity in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFrame(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 bool odd
 )
```

```
void SetFrame(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 bool odd
 )
void SetFrame(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 bool odd
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

odd

Specifies which frame is replaced (the frame made up of all lines of the same parity as odd).

Remarks

The size of the destination image is determined as follows: DstImage_Width = SrcImage_ WidthDstImage_Height = (SrcImage_Height + 1 - odd) / 2

EasyImage::SetInvCircleWarp

Prepares suitable warp images for use with function EasyImage::Warp to unwarp a straight rectangle into a circular ring-wedge shape. This is a polar to cartesian image conversion function.
See also EasyImage::SetCircleWarp.

Namespace: Euresys::Open_eVision

[C++]

```
void SetInvCircleWarp(
 float centerX,
 float centerY,
 int numberOfRadialSampledPoints,
 float minimumRadius,
 float maximumRadius,
 int numberOfTangentSampledPoints,
 float minimumAngle,
 float maximumAngle,
 EImageBW16* warpImageX,
 EImageBW16* warpImageY,
 int warpImageWidth,
 int warpImageHeight
 )
```

Parameters

centerX

Abscissa of the ring-wedge center.

centerY

Ordinate of the ring-wedge center.

numberOfRadialSampledPoints

Number of points to be sampled in the radial direction (the height of the source polar image).

minimumRadius

Starting radius of the ring-wedge shape.

maximumRadius

Ending radius of the ring-wedge shape.

numberOfTangentSampledPoints

Number of points to be sampled in the tangent direction (the width of the source polar image).

minimumAngle

Starting angle of the ring-wedge shape.

maximumAngle

Ending angle of the ring-wedge shape.

warpImageX

Destination cartesian image for the abscissas.

warpImageY

Destination cartesian image for the ordinates.

warpImageWidth

The width of the destination cartesian images. Optional parameter, if omitted a best fit size is calculated.

warpImageHeight

The height of the destination cartesian images. Optional parameter, if omitted a best fit size is calculated.

EasyImage::SetRecursiveAverageLUT

Pre-compute the required non-linear transfer function for noise reduction by recursive temporal averaging.

Namespace: Euresys::Open_eVision

[C++]

```
void SetRecursiveAverageLUT(
 EBW16Vector* lookupTable,
 float reductionNoiseFactor,
 float reductionNoiseWidth
 )
```

Parameters

lookupTable

Pointer to the LUT vector holding the non-linear transfer function.

reductionNoiseFactor

Noise reduction factor. The larger the value, the more effectively noise will be reduced.

reductionNoiseWidth

Indicates the extent to which noise reduction applies to large variations in gray-level values. For variations small with respect to this parameter, noise will be reduced by a factor close to the reductionNoiseFactor value; for variations much larger than reductionNoiseWidth, no noise reduction will take place.

Remarks

This function is a companion to EasyImage::RecursiveAverage.

EasyImage::SetupEqualize

Prepares a lookup-table for image equalization, using an image histogram.

Namespace: Euresys::Open_eVision

```
[C++]
void SetupEqualize(
 EBWHistogramVector* histogram,
 EBW8Vector* lookupTable
 )
```

Parameters

histogram

Pointer to the source histogram vector.

lookupTable

Pointer to the destination lookup-table vector.

Remarks

This function, along with EasyImage::Histogram and EasyImage::Lut, is an alternative to using EasyImage::Equalize.

EasyImage::SetupInverseWarp

Prepares suitable inverse warp images for use with function EasyImage::Warp to unwarp an invertible LUT given by the warpImageX and warpImageY.

Namespace: Euresys::Open_eVision

[C++]

```
void SetupInverseWarp(
 EImageBW16* warpImageX,
 EImageBW16* warpImageY,
 EImageBW16* inverseWarpImageX,
 EImageBW16* inverseWarpImageY
 )
```

Parameters

warpImageX

Pointer to the X lookup image.

warpImageY

Pointer to the Y lookup image.

inverseWarpImageX

Pointer to the inverse X lookup image.

inverseWarpImageY

Pointer to the inverse Y lookup image.

Remarks

Typical use is warping back a text printed around a circle. The behavior when using non invertible LUT is undefined.

EasyImage::Shrink

Resizes an image to a smaller size. Pre-filtering is applied to avoid aliasing.

Namespace: Euresys::Open_eVision

[C++]

```
void Shrink(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Shrink(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void Shrink(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

EasyImage::SignalNoiseRatio

Computes the signal to noise ratio, in dB, by comparing a given image to a reference image.

Namespace: Euresys::Open_eVision

[C++]

```
float SignalNoiseRatio(
 const EROIBW8* sourceImage,
 const EROIBW8* referenceImage,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float SignalNoiseRatio(
 const EROIBW16* sourceImage,
 const EROIBW16* referenceImage,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float SignalNoiseRatio(
 const EROIC24* sourceImage,
 const EROIC24* referenceImage,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float SignalNoiseRatio(
 const EROIBW8* sourceImage,
 const EROIBW8* referenceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float SignalNoiseRatio(
 const EROIBW16* sourceImage,
 const EROIBW16* referenceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )

float SignalNoiseRatio(
 const EROIC24* sourceImage,
 const EROIC24* referenceImage,
 const EROIBW8* mask,
 Euresys::Open_eVision::EReferenceNoise referenceNoise
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

referenceImage

Pointer to the reference image/ROI.

referenceNoise

Specifies how the reference image is affected by noise, as defined by EReferenceNoise.

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

pSrcImage

-

pRefImage

-

un32Count

-

eReferenceNoise

-

Remarks

The reference image can be noiseless (obtained by suppressing the source of noise) or be affected by a noise of the same distribution as the given image.
The signal amplitude is defined as the sum of the squared pixel gray-level values while the noise amplitude is defined as the sum of the squared difference between the pixel gray-level values of the given image and the reference.

EasyImage::SwapFrames

Interchanges the even and odd rows of an image.

Namespace: Euresys::Open_eVision

[C++]

```
void SwapFrames(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void SwapFrames(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void SwapFrames(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

Remarks

This is helpful when acquisition of an interleaved image has confused even and odd frames.

EasyImage::Thick

Applies a thickening operation on an image, using a 3x3 kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void Thick(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EKernel* thickeningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )

void Thick(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 EKernel* thickeningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )

void Thick(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 EKernel* thickeningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )

int Thick(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 EKernel* thickeningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thickeningKernel

Pointer to the thickening kernel.

rotationMode

Rotation mode, as defined by EKernelRotation.

numberOfIterations

Number of iterations to apply. 0 indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either EKernelRotation_Clockwise or EKernelRotation_Anticlockwise, a pass comprises eight kernel rotations.

Remarks

The thickening kernel coefficients must be 0 (matching black pixel, value 0), 1 (matching non black pixel, value > 0) or -1 (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to 255.

EasyImage::Thin

Applies a thinning operation on an image, using a 3x3 kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void Thin(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EKernel* thinningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )

void Thin(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 EKernel* thinningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )

void Thin(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 EKernel* thinningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )
```

```
int Thin(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 EKernel* thinningKernel,
 Euresys::Open_eVision::EKernelRotation rotationMode,
 int& numberOfIterations
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as source image.

thinningKernel

Pointer to the thinning kernel.

rotationMode

Rotation mode, as defined by EKernelRotation.

numberOfIterations

Number of iterations to apply. 0 indicates stop when convergence is reached. Upon return, gives the number of passes actually performed. If the rotation mode is set to either EKernelRotation_Clockwise or EKernelRotation_Anticlockwise, a pass comprises eight kernel rotations.

Remarks

The thinning kernel coefficients must be 0 (matching black pixel, value 0), 1 (matching non black pixel, value > 0) or -1 (don't care). When a match is found between the kernel coefficients and the neighborhood of a pixel, the pixel value is set to 0.

EasyImage::ThreeLevelsMinResidueThreshold

Computes the two threshold values used to separate the pixels of an image in three classes.

Namespace: Euresys::Open_eVision

[C++]

```
float ThreeLevelsMinResidueThreshold(
 EBWHistogramVector* histogram,
 EBW8& firstGrayPixelValue,
 EBW8& firstWhitePixelValue,
 float& averageBlack,
 float& averageGray,
 float& averageWhite
 )
```

Parameters

histogram
Histogram of the image.

firstGrayPixelValue
Low threshold.

firstWhitePixelValue
High threshold.

averageBlack
Average value of the black pixels (pixels under the low threshold).

averageGray
Average value of the gray pixels (pixels between the low threshold and the high threshold).

averageWhite
Average value of the white pixels (pixels over the high threshold).

Remarks

These values are computed using the minimum residue criterion from the histogram of the image. The function returns the minimum residue as per the method. The residue is the Euclidian distance between the source image and the thresholded image.

EasyImage::Threshold

Binarize an image by setting pixels to two different possible values in a destination image, according to their value in a source image.

Namespace: Euresys::Open_eVision

[C++]

```
void Threshold(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 threshold,
 OEV_UINT8 lowValue,
 OEV_UINT8 highValue,
 float relativeThreshold
 )

void Threshold(
 const EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 threshold,
 OEV_UINT8 lowValue,
 OEV_UINT8 highValue,
 float relativeThreshold
 )

void Threshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )
```

```
void Threshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage
 )

void Threshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 threshold
 )

void Threshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 threshold
 )

void Threshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 threshold,
 EBW16 lowValue,
 EBW16 highValue
 )

void Threshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 threshold,
 EBW16 lowValue,
 EBW16 highValue
 )

void Threshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float relativeThreshold
 )

void Threshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 float relativeThreshold
 )

void Threshold(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 float relativeThreshold,
 EBW16 lowValue,
 EBW16 highValue
 )
```

```
void Threshold(
 const EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 float relativeThreshold,
 EBW16 lowValue,
 EBW16 highValue
 )

void Threshold(
 EROIC24* sourceImage,
 EROIBW8* destinationImage,
 EC24 minimum,
 EC24 maximum
 )

void Threshold(
 const EROIC24* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 EC24 minimum,
 EC24 maximum
 )

void Threshold(
 EROIC24* sourceImage,
 EROIBW8* destinationImage,
 EC24 minimum,
 EC24 maximum,
 EColorLookup* colorLookupTable,
 EBW8 rejectValue,
 EBW8 acceptValue
 )

void Threshold(
 const EROIC24* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 EC24 minimum,
 EC24 maximum,
 EColorLookup* colorLookupTable,
 EBW8 rejectValue,
 EBW8 acceptValue
 )

void Threshold(
 EROIC24* sourceImage,
 EROIBW8* destinationImage,
 EC24 minimum,
 EC24 maximum,
 EColorLookup* colorLookupTable
 )
```

```
void Threshold(
 const EROIC24* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 EC24 minimum,
 EC24 maximum,
 EColorLookup* colorLookupTable
 )
void Threshold(
 EROIBW8* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 threshold,
 float relativeThreshold
 )
void Threshold(
 EROIBW16* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 threshold,
 float relativeThreshold
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

threshold

The value to compare each pixel to

lowValue

Value for pixels below the threshold (by default, 0).

highValue

Value for pixels above the threshold (by default, it is set to 255 for BW8 destination images and 65535 for BW16 destination images).

relativeThreshold

Fraction of the image pixels that will be set below the threshold. Only used when the threshold value is EThresholdMode_Relative (by default, 0.5). This value must be greater than (or equal to) 0 and strictly less than 1.

region

Pointer to a region to apply the function only on a particular region in the image.

minimum

Three lower thresholds combined in a single color value.

maximum

Three upper thresholds combined in a single color value.

colorLookupTable

Pointer to the color lookup table to be applied before thresholding, if any.

rejectValue

Value for pixels falling outside the range (by default, 0).

acceptValue

Value for pixels falling inside the range (by default, 255).

Remarks

When the source image is gray-level, the pixel values are measured against a threshold. All pixels below this threshold will yield a low value in the destination image, and all pixels above or on the threshold will yield a high value.
When the destination image is binary (BW1 pixel type), then the values are set to 0 or 1, according to the criterion.
When the destination image is gray-level (BW8 or BW16), then the values are set to 0 or to the maximum pixel value for the image type (255 for BW8 and 65535 for BW16). In some overloads, these minimum and maximum destination values can be specified.
When the source image is gray-level, several modes are available: absolute (the threshold value is given), relative (the threshold value is computed to obtain a desired fraction of the image pixels), or automatic (using three different criteria). In the function overloads where this mode cannot be specified, it is assumed to be absolute.
If the source image is color, all pixels whose components are comprised in a range of values (minimum to maximum) will be set to a constant value (white by default), while other pixels will be set to another constant value (black by default). In this case, if a color lookup is specified, it is applied on the fly to the color image before thresholding.
The simpler color image overload does not support the use of an on-the-fly color lookup table nor rejectValue/acceptValue arguments. On the other hand, it has been MMX optimized, and will run significantly faster when the acceptance region is large.

EasyImage::Transpose

Transpose an image.

Namespace: Euresys::Open_eVision

[C++]

```
void Transpose(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )

void Transpose(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage
 )

void Transpose(
 EROIC24* sourceImage,
 EROIC24* destinationImage
 )

void Transpose(
 EROIBW8* sourceImage
 )

void Transpose(
 EROIBW16* sourceImage
 )
```

```
void Transpose(
 EROIC24* sourceImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. May not be the same as the source image.

Remarks

Destination image/roi width and height should be respectively equal to source image/roi height and width.

EasyImage::TwoLevelsMinResidueThreshold

Computes the threshold value used to separate the pixels of an image in two classes.

Namespace: Euresys::Open_eVision

[C++]

```
float TwoLevelsMinResidueThreshold(
 EBWHistogramVector* histogram,
 EBW8& firstWhitePixelValue,
 float& averageBlack,
 float& averageWhite
 )
```

Parameters

histogram

Histogram of the image.

firstWhitePixelValue

Threshold.

averageBlack

Average value of the black pixels (pixels under the threshold).

averageWhite

Average value of the white pixels (pixels over the threshold).

Remarks

This value is computed using the minimum residue criterion from the histogram of the image. The function returns the minimum residue as per the method. The residue is the Euclidian distance between the source image and the thresholded image.

EasyImage::Uniformize

Shading correction is the process of transforming the gray or color component values of an image, using one or two reference images or vectors.

Namespace: Euresys::Open_eVision

[C++]

```
void Uniformize(
 const EROIBW8* sourceImage,
 EBW8 pixelReference,
 const EROIBW8* imageReference,
 EROIBW8* destinationImage,
 bool multiplicative
 )

void Uniformize(
 const EROIBW16* sourceImage,
 EBW16 pixelReference,
 const EROIBW16* imageReference,
 EROIBW16* destinationImage,
 bool multiplicative
 )

void Uniformize(
 const EROIC24* sourceImage,
 EC24 pixelReference,
 const EROIC24* imageReference,
 EROIC24* destinationImage,
 bool multiplicative
 )

void Uniformize(
 const EROIBW8* sourceImage,
 EBW8 pixelReference,
 const EBW8Vector* vectorOfPixelReference,
 EROIBW8* destinationImage,
 bool multiplicative
 )

void Uniformize(
 const EROIBW16* sourceImage,
 EBW16 pixelReference,
 const EBW16Vector* vectorOfPixelReference,
 EROIBW16* destinationImage,
 bool multiplicative
 )
```

```
void Uniformize(
 const EROIC24* sourceImage,
 EC24 pixelReference,
 const EC24Vector* vectorOfPixelReference,
 EROIC24* destinationImage,
 bool multiplicative
 )

void Uniformize(
 const EROIBW8* sourceImage,
 EBW8 pixelLightReference,
 const EROIBW8* imageLightReference,
 EBW8 pixelDarkReference,
 const EROIBW8* imageDarkReference,
 EROIBW8* destinationImage
 )

void Uniformize(
 const EROIBW16* sourceImage,
 EBW16 pixelLightReference,
 const EROIBW16* imageLightReference,
 EBW16 pixelDarkReference,
 const EROIBW16* imageDarkReference,
 EROIBW16* destinationImage
 )

void Uniformize(
 const EROIC24* sourceImage,
 EC24 pixelLightReference,
 const EROIC24* imageLightReference,
 EC24 pixelDarkReference,
 const EROIC24* imageDarkReference,
 EROIC24* destinationImage
 )

void Uniformize(
 const EROIBW8* sourceImage,
 EBW8 pixelLightReference,
 const EBW8Vector* vectorOfPixelLightReference,
 EBW8 pixelDarkReference,
 const EBW8Vector* vectorOfPixelDarkReference,
 EROIBW8* destinationImage
 )

void Uniformize(
 const EROIBW16* sourceImage,
 EBW16 pixelLightReference,
 const EBW16Vector* vectorOfPixelLightReference,
 EBW16 pixelDarkReference,
 const EBW16Vector* vectorOfPixelDarkReference,
 EROIBW16* destinationImage
 )
```

```
void Uniformize(
 const EROIC24* sourceImage,
 EC24 pixelLightReference,
 const EC24Vector* vectorOfPixelLightReference,
 EC24 pixelDarkReference,
 const EC24Vector* vectorOfPixelDarkReference,
 EROIC24* destinationImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

pixelReference

Constant value to transform the reference image or vector into.

imageReference

Pointer to the reference source image/ROI or vector.

destinationImage

Pointer to the destination image/ROI.

multiplicative

true, if the transform is multiplicative (gain); false, if the transform is additive (offset) (by default, true).

vectorOfPixelReference

Constant value to transform the reference image or vector into.

pixelLightReference

Constant value to transform the light reference image or vector into.

imageLightReference

Pointer to the light reference source image/ROI or vector.

pixelDarkReference

Constant value to transform the dark reference image/ROI or vector into.

imageDarkReference

Pointer to the dark reference source image/ROI or vector.

vectorOfPixelLightReference

Constant value to transform the light reference image or vector into.

vectorOfPixelDarkReference

Constant value to transform the dark reference image/ROI or vector into.

Remarks

The intent is to compensate for non-uniform lighting or sensor response non-uniformity by providing images of the background with no foreground object present.
In the case of area-scan cameras, the illumination can change anywhere in the field of view, requiring 2D compensation. In the case of line-scan cameras imaging moving parts, illumination remains constant across image rows. Only 1D compensation is required. In this case, the reference illumination is specified as a vector, which is replicated across all image rows.
* When a single reference image is used, the transform is analog to an adaptive (space-variant) gain *or* offset (Gain * Intensity or Intensity + Offset); the transform lets the reference image(s) become a specified constant value, i.e. flat field illumination.
* When two reference images are used, the transform is analog to adaptive gain *and* offset (Gain * Intensity + Offset); the transform let both reference images become specified constants, i.e. flat field illumination with a correct black reference.

Note. The reference image(s) should be chosen such that they contain no saturated pixel values (remain in the linear domain) and little (filtered out) noise.

EasyImage::VerticalMirror

Mirrors an image vertically (the rows are swapped).

Namespace: Euresys::Open_eVision

[C++]

```
void VerticalMirror(
 EROIBW8* sourceImage
 )

void VerticalMirror(
 EROIBW16* sourceImage
 )

void VerticalMirror(
 EROIC24* sourceImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

EasyImage::Warp

Transforms an image so that each pixels are moved to the locations specified in the "warp" images used as look-up tables.

Namespace: Euresys::Open_eVision

[C++]

```
void Warp(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 EImageBW16* warpImageX,
 EImageBW16* warpImageY,
 int shiftX,
 int shiftY
 )

void Warp(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 EImageBW16* warpImageX,
 EImageBW16* warpImageY,
 int shiftX,
 int shiftY
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI.

warpImageX

Pointer to the X lookup image.

warpImageY

Pointer to the Y lookup image.

shiftX

Horizontal translation.

shiftY

Vertical translation.

Remarks

For example, pixel [10,20] moves to location [WarpXImage[10,20], WarpYImage[10,20]].

EasyImage::WeightedMoments

Computes the zero-th, first, second, third or fourth order weighted moments on the gray-level image. The weight of a pixel is its gray-level value.

Namespace: Euresys::Open_eVision

[C++]

```
void WeightedMoments(
 const EROIBW8* sourceImage,
 float& M,
 float& Mx,
 float& My
 )

void WeightedMoments(
 const EROIBW16* sourceImage,
 float& M,
 float& Mx,
 float& My
 )

void WeightedMoments(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My
 )

void WeightedMoments(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My
 )

void WeightedMoments(
 const EROIBW8* sourceImage,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )

void WeightedMoments(
 const EROIBW16* sourceImage,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
```

```
void WeightedMoments(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
void WeightedMoments(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
void WeightedMoments(
 const EROIBW8* sourceImage,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy
 )
void WeightedMoments(
 const EROIBW16* sourceImage,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy
 )
```

```
void WeightedMoments(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy
 )

void WeightedMoments(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy
 )

void WeightedMoments(
 const EROIBW8* sourceImage,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy,
 float& Mxxxx,
 float& Mxxxy,
 float& Mxxyy,
 float& Mxyyy,
 float& Myyyy
 )
```

```
void WeightedMoments(
 const EROIBW16* sourceImage,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy,
 float& Mxxxx,
 float& Mxxxy,
 float& Mxxyy,
 float& Mxyyy,
 float& Myyyy
 )
void WeightedMoments(
 const EROIBW8* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy,
 float& Mxxxx,
 float& Mxxxy,
 float& Mxxyy,
 float& Mxyyy,
 float& Myyyy
 )
```

```
void WeightedMoments(
 const EROIBW16* sourceImage,
 const ERegion& region,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy,
 float& Mxxxx,
 float& Mxxxy,
 float& Mxxyy,
 float& Mxyyy,
 float& Myyyy
 )

void WeightedMoments(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My
 )

void WeightedMoments(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My
 )

void WeightedMoments(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
```

```
void WeightedMoments(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy
 )
void WeightedMoments(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy
 )
void WeightedMoments(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy
 )
```

```
void WeightedMoments(
 const EROIBW8* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy,
 float& Mxxxx,
 float& Mxxxy,
 float& Mxxyy,
 float& Mxyyy,
 float& Myyyy
 )
void WeightedMoments(
 const EROIBW16* sourceImage,
 const EROIBW8* mask,
 float& M,
 float& Mx,
 float& My,
 float& Mxx,
 float& Mxy,
 float& Myy,
 float& Mxxx,
 float& Mxxy,
 float& Mxyy,
 float& Myyy,
 float& Mxxxx,
 float& Mxxxy,
 float& Mxxyy,
 float& Mxyyy,
 float& Myyyy
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

M

Reference to the zero-th order weighted moment (total gray value).

Mx

Reference to the first order moments (weighted sums of abscissas and ordinates).

My

Reference to the first order moments (weighted sums of abscissas and ordinates).

region

Pointer to a region to apply the function only on a particular region in the image.

Mxx

Reference to the second order uncentered moments (weighted sums of squared abscissas, cross-product and squared ordinates).

Mxy

Reference to the second order uncentered moments (weighted sums of squared abscissas, cross-product and squared ordinates).

Myy

Reference to the second order uncentered moments (weighted sums of squared abscissas, cross-product and squared ordinates).

Mxxx

Reference to the third order uncentered moments (weighted sums of third order products).

Mxxy

Reference to the third order uncentered moments (weighted sums of third order products).

Mxyy

Reference to the third order uncentered moments (weighted sums of third order products).

Myyy

Reference to the third order uncentered moments (weighted sums of third order products).

Mxxxx

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Mxxxy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Mxxyy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Mxyyy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

Myyyy

Reference to the fourth order uncentered moments (weighted sums of fourth order products).

mask

Pointer to a mask to apply the function only on a particular region in the image. Note: the mask must have the same size as the source image.

EasyImage::WhiteTopHatBox

Performs a top-hat filtering on an image (source image minus open image) on a rectangular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void WhiteTopHatBox(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void WhiteTopHatBox(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void WhiteTopHatBox(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void WhiteTopHatBox(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void WhiteTopHatBox(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void WhiteTopHatBox(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )

void WhiteTopHatBox(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth; 0 is allowed).

region

Region to apply the function on.

Remarks

This filter enhances the thin white features.

EasyImage::WhiteTopHatDisk

Performs a top-hat filtering on an image (source image minus open image) on a circular kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void WhiteTopHatDisk(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void WhiteTopHatDisk(
 EROIBW16* sourceImage,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void WhiteTopHatDisk(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )

void WhiteTopHatDisk(
 EROIBW8* sourceImage,
 const ERegion& region,
 EROIBW8* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

```
void WhiteTopHatDisk(
 EROIBW16* sourceImage,
 const ERegion& region,
 EROIBW16* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void WhiteTopHatDisk(
 EROIC24* sourceImage,
 const ERegion& region,
 EROIC24* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
void WhiteTopHatDisk(
 EROIBW1* sourceImage,
 EROIBW1* destinationImage,
 OEV_UINT32 halfOfKernelWidth
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination image/ROI. Must not be the same as the source image.

halfOfKernelWidth

Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

region

Region to apply the function on.

Remarks

This filter enhances the thin white features.

4.24. EasyObject Class

This class contains static properties and methods specific to the EasyObject library.

Namespace: Euresys::Open_eVision

Methods

ContourArea	Computes the area of an object from its contour.
ContourGravityCenter	Computes the area and gravity center of an object from its contour.
ContourInertia	Computes the inertia parameters of an object from its contour.
IsFloatFeature	Tests whether a given feature is associated with floating-point values.
IsIntegerFeature	Tests whether a given feature is associated with integer values.

IsUnsignedIntegerFeature	Tests whether a given feature is associated with unsigned integer values.

EasyObject::ContourArea

Computes the area of an object from its contour.

Namespace: Euresys::Open_eVision

```
[C++]
void ContourArea(
 EPathVector* pPathVector,
 int& n32Area
 )
```

Parameters

pPathVector

Pointer to the source vector.

n32Area

Reference to the area to compute.

EasyObject::ContourGravityCenter

Computes the area and gravity center of an object from its contour.

Namespace: Euresys::Open_eVision

```
[C++]
void ContourGravityCenter(
 EPathVector* pPathVector,
 int& n32Area,
 float& f32GravityCenterX,
 float& f32GravityCenterY
 )
```

Parameters

pPathVector

Pointer to the source vector.

n32Area

Reference to the area to compute.

f32GravityCenterX

Reference to the abscissa of the gravity center to compute.

f32GravityCenterY

Reference to the ordinate of the gravity center to compute.

EasyObject::ContourInertia

Computes the inertia parameters of an object from its contour.

Namespace: Euresys::Open_eVision

[C++]

```
void ContourInertia(
 EPathVector* pPathVector,
 int& n32Area,
 float& f32GravityCenterX,
 float& f32GravityCenterY,
 float& f32SigmaX,
 float& f32SigmaY,
 float& f32SigmaXY
 )
```

Parameters

pPathVector
Pointer to the source vector.

n32Area
Reference to the area to compute.

f32GravityCenterX
Reference to the abscissa of the gravity center to compute.

f32GravityCenterY
Reference to the ordinate of the gravity center to compute.

f32SigmaX
Centered cross moment of inertia.

f32SigmaY
Centered moment of inertia around Y.

f32SigmaXY
Centered cross moment of inertia.

EasyObject::IsFloatFeature

Tests whether a given feature is associated with floating-point values.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsFloatFeature(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature.

Remarks

Most features are floating-point. The exceptions are listed in these functions: EasyObject::IsUnsignedIntegerFeature and EasyObject::IsIntegerFeature.

EasyObject::IsIntegerFeature

Tests whether a given feature is associated with integer values.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsIntegerFeature(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature.

Remarks

The features associated with integer values are: EFeature_ContourX, EFeature_ContourY, EFeature_LeftLimit, EFeature_RightLimit, EFeature_TopLimit, and EFeature_BottomLimit.

EasyObject::IsUnsignedIntegerFeature

Tests whether a given feature is associated with unsigned integer values.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsUnsignedIntegerFeature(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature.

Remarks

The features associated with unsigned integer values are: EFeature_ElementIndex, EFeature_LayerIndex, EFeature_RunCount, EFeature_Area and EFeature_LargestRun.

4.25. EBarCode Class

This class is deprecated.

Manages a complete context for the reading or verification of bar codes in EasyBarCode. Deprecated, use EBarCodeReader instead

Namespace: Euresys::Open_eVision

Methods

Decode	Provides the decoded information (or a reading error code) corresponding to the specified symbology.
Detect	Processes the image, reads the possible contents of the bar code corresponding to all enabled symbologies and rates their likeliness.
Drag	Moves a handle to a new position and updates the position parameters of the symbol bounding box.
Draw	Draws the symbol bounding box.
DrawWithCurrentPen	Draws the symbol bounding box.
EBarCode	Creates an EBarCode object.
GetAdditionalSymbologies	Enabled symbologies belonging to the group of additional symbologies.
GetAngle	Orientation of the shape.
GetCenter	Center point of the EBarCode.
GetCenterX	Abscissa of the origin point of the EBarCode.
GetCenterY	Ordinate of the origin point of the EBarCode.
GetDecodedAngle	Allows retrieving the bar code reading angle, pertaining to the specified symbology, or, at least, to the most likely symbology.
GetDecodedDirection	Returns the encoding direction of the bar code, pertaining to the specified symbology, or, at least, to the most likely symbology.
GetDecodedRectangle	Allows retrieving the bar code reading rectangle, pertaining to the specified symbology, or, at least, to the most likely symbology.
GetDecodedSymbology	Returns the identifier of one of the symbologies that were successfully decoded.
GetKnownLocation	Flag indicating whether the symbol location is known or not.
GetKnownModule	Flag indicating whether the symbol module is known or not.
GetModule	Module value.
GetNumDecodedSymbologies	Number of symbologies (among the enabled ones) for which the decoding process was successful.
GetNumEnabledSymbologies	Number of enabled symbologies.

GetRectangleShape	Get the rectangle shape associated with the EBarCode.
GetRelativeReadingSizeX	Reading area width, relative to the symbol extent.
GetRelativeReadingSizeY	Reading area height, relative to the symbol extent.
GetRelativeReadingX	Reading area abscissa, relative to the symbol position.
GetRelativeReadingY	Reading area ordinate, relative to the symbol position.
GetSizeX	X size of the EBarCode
GetSizeY	Y size of the EBarCode
GetStandardSymbologies	Enabled symbologies belonging to the group of standard symbologies.
GetSymbologyName	Retrieves the name of given symbology as a string.
GetThicknessRatio	Bars thickness ratio.
GetVerifyChecksum	"VerifyChecksum" mode.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Load	Loads an EBarCode.The given EBarCode must have been created for reading.
Read	Processes the image, reads the possible contents of the bar code and returns the single possible decoding (mono-symbology) or the most likely one (multi-symbology) or a reading error code.
Save	Loads an EBarCode.The given EBarCode must have been created for writing.
SetAdditionalSymbologies	Enabled symbologies belonging to the group of additional symbologies.
SetAngle	Orientation of the shape.
SetCenter	Center point of the EBarCode.
SetCenterXY	Sets the center coordinates of a EBarCode object.
SetKnownLocation	Flag indicating whether the symbol location is known or not.
SetKnownModule	Flag indicating whether the symbol module is known or not.
SetModule	Module value.
SetReadingCenter	Sets the reading area center coordinates, relative to the symbol position and extent.
SetReadingSize	Sets the reading area size, relative to the symbol extent.
SetRectangle	Sets the nominal position (center coordinates), size and rotation angle of the symbol bounding box according to a known rectangle.
SetSize	Sets the size of a EBarCode object.
SetStandardSymbologies	Enabled symbologies belonging to the group of standard symbologies.
SetThicknessRatio	Bars thickness ratio.

SetVerifyChecksum	"VerifyChecksum" mode.

EBarCode::GetAdditionalSymbologies

EBarCode::SetAdditionalSymbologies

This property is deprecated.

Enabled symbologies belonging to the group of additional symbologies.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetAdditionalSymbologies() const
void SetAdditionalSymbologies(OEV_UINT32 un32AdditionalSymbologies)
```

Remarks

Due to the large number of supported symbologies, they have been gathered in two groups. For more information about these groups, see ESymbologies.

EBarCode::GetAngle

EBarCode::SetAngle

This property is deprecated.

Orientation of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAngle() const
void SetAngle(float f32Angle)
```

EBarCode::GetCenter

EBarCode::SetCenter

This property is deprecated.

Center point of the EBarCode.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

EBarCode::GetCenterX

This property is deprecated.

Abscissa of the origin point of the EBarCode.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

EBarCode::GetCenterY

This property is deprecated.

Ordinate of the origin point of the EBarCode.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

EBarCode::Decode

This method is deprecated.

Provides the decoded information (or a reading error code) corresponding to the specified symbology.

Namespace: Euresys::Open_eVision

[C++]

```
std::string Decode(
 Euresys::Open_eVision::ESymbologies symbology
 )
```

Parameters

symbology

Specified symbology, as defined by ESymbologies this symbology must have been enabled).

Remarks

Before calling EBarCode::Decode, an EBarCode::Detect operation must have been performed. In case of the mono-symbology mode, or if only the most likely decoding matters, the EBarCode::Read method should be used.

EBarCode::Detect

This method is deprecated.

Processes the image, reads the possible contents of the bar code corresponding to all enabled symbologies and rates their likeliness.

Namespace: Euresys::Open_eVision

[C++]

```
void Detect(
 EROIBW8* sourceImage
 )
```

Parameters

sourceImage

Pointer to the image containing the bar code.

Remarks

Processing the image means finding the symbol (in case of automatic location mode) and retrieving its descriptive parameters. The decoded information corresponding to a specific symbology is provided by the decode function. The symbologies that were successfully decoded are ranked by decreasing likeliness (range 0 to NumDecodedSymbologies-1). In case of the mono-symbology mode or if only the most likely decoding matters, the Read function should be used.

EBarCode::Drag

This method is deprecated.

Moves a handle to a new position and updates the position parameters of the symbol bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int cursorX,
 int cursorY
 )
```

Parameters

cursorX

Cursor current coordinates.

cursorY

Cursor current coordinates.

EBarCode::Draw

This method is deprecated.

Draws the symbol bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the symbol bounding box must be displayed, as defined by EDrawingMode.

daughters

true if the shapes attached to the symbol bounding box are to be displayed as well.

color

The color in which to draw the overlay.

Remarks

The bounding box corresponds to the nominal position of the bar code (EDrawingMode_Nominal), in case this information has been explicitly provided, and to the actual position (EDrawingMode_Actual) if it has been determined by image analysis. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBarCode::DrawWithCurrentPen

This method is deprecated.

Draws the symbol bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the symbol bounding box must be displayed, as defined by EDrawingMode.

daughters

true if the shapes attached to the symbol bounding box are to be displayed as well.

Remarks

The bounding box corresponds to the nominal position of the bar code (EDrawingMode_Nominal), in case this information has been explicitly provided, and to the actual position (EDrawingMode_Actual) if it has been determined by image analysis. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBarCode::EBarCode

This method is deprecated.

Creates an EBarCode object.

Namespace: Euresys::Open_eVision

```
[C++]
void EBarCode(
 )
```

EBarCode::GetDecodedAngle

This method is deprecated.

Allows retrieving the bar code reading angle, pertaining to the specified symbology, or, at least, to the most likely symbology.

Namespace: Euresys::Open_eVision

```
[C++]
void GetDecodedAngle(
 float& decodedAngle
 )
void GetDecodedAngle(
 float& decodedAngle,
 float cutAngle
 )
void GetDecodedAngle(
 Euresys::Open_eVision::ESymbologies symbology,
 float& decodedAngle
 )
void GetDecodedAngle(
 Euresys::Open_eVision::ESymbologies symbology,
 float& decodedAngle,
 float cutAngle
 )
```

Parameters

decodedAngle

Returned bar code reading angle value.

cutAngle

Cut angle value (degrees) defining the allowed range for the bar code reading angle ([cutAngle, cutAngle + 360]). By default, the cut angle equals -45.

symbology

Specified symbology, as defined by ESymbologies (this symbology must have been enabled).

EBarCode::GetDecodedDirection

This method is deprecated.

Returns the encoding direction of the bar code, pertaining to the specified symbology, or, at least, to the most likely symbology.

Namespace: Euresys::Open_eVision

```
[C++]
void GetDecodedDirection(
 bool& directEncoding
 )
void GetDecodedDirection(
 Euresys::Open_eVision::ESymbologies symbology,
 bool& directEncoding
 )
```

Parameters

directEncoding

Boolean holding the encoding direction status.

symbology

Specified symbology, as defined by ESymbologies (this symbology must have been enabled).

Remarks

The encoding direction of the bar code is "Direct" or "Inverse". The encoding direction is said to be "Direct" when the bar code longitudinal axis falls in the range [-45 degrees, 135]. Conversely, when the bar code longitudinal axis doesn't fall in the previous range, the encoding direction is said to be "Inverse".

EBarCode::GetDecodedRectangle

This method is deprecated.

Allows retrieving the bar code reading rectangle, pertaining to the specified symbology, or, at least, to the most likely symbology.

Namespace: Euresys::Open_eVision

[C++]

```
void GetDecodedRectangle(
 ERectangle& rect
 )

void GetDecodedRectangle(
 Euresys::Open_eVision::ESymbologies symbology,
 ERectangle& rect
 )
```

Parameters

rect

Returned bar code reading rectangle.

symbology

Specified symbology, as defined by ESymbologies (this symbology must have been enabled).

EBarCode::GetDecodedSymbology

This method is deprecated.

Returns the identifier of one of the symbologies that were successfully decoded.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ESymbologies GetDecodedSymbology(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the specified symbology (range 0 to NumDecodedSymbologies-1).

Remarks

The desired symbology is specified by its ranking index (range 0 to NumDecodedSymbologies-1). The symbologies that were successfully decoded are ranked by decreasing likeliness.

EBarCode::GetSymbologyName

This method is deprecated.

Retrieves the name of given symbology as a string.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetSymbologyName(
 Euresys::Open_eVision::ESymbologies symbology
 )
```

Parameters

symbology

Symbology.

EBarCode::HitTest

This method is deprecated.

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 bool daughters
 )
```

Parameters

daughters

true if the handles of the shapes attached to the symbol bounding box have to be considered as well.

EBarCode::GetKnownLocation

EBarCode::SetKnownLocation

This property is deprecated.

Flag indicating whether the symbol location is known or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetKnownLocation() const
void SetKnownLocation(bool bKnownLocation)
```

Remarks

In case of known location, use EBarCode::Rectangle to adjust a rectangle around the symbol.

EBarCode::GetKnownModule

EBarCode::SetKnownModule

This property is deprecated.

Flag indicating whether the symbol module is known or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetKnownModule() const
void SetKnownModule(bool bKnownModule)
```

Remarks

If true, it is also necessary to get the EBarCode::Module and EBarCode::ThicknessRatio properties in order to specify the requested module and thickness ratio.

EBarCode::Load

This method is deprecated.

Loads an EBarCode.The given EBarCode must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path,
 bool daughters
 )

void Load(
 ESerializer* serializer,
 bool daughters
 )
```

Parameters

path

The file path.

daughters

Indicates if the load must be done on the whole hierarchy or just this object.

serializer

Pointer to the ESerializer created for reading.

EBarCode::GetModule

EBarCode::SetModule

This property is deprecated.

Module value.

Namespace: Euresys::Open_eVision

[C++]

```
float GetModule() const

void SetModule(float f32Module)
```

Remarks

The module value is a descriptive parameter participating in the encoding; it corresponds to the thinner bar width. Symbols whose bars thickness can take two values, the module and V times the module (where V runs from 1.5 to 3), are called *binary* bar codes. When the bars thickness are small integer multiples (1 to 4 or 5) of a module, the symbols are called *modular* bar codes.

EBarCode::GetNumDecodedSymbologies

This property is deprecated.

Number of symbologies (among the enabled ones) for which the decoding process was successful.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumDecodedSymbologies() const
```

EBarCode::GetNumEnabledSymbologies

This property is deprecated.

Number of enabled symbologies.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumEnabledSymbologies() const
```

EBarCode::Read

This method is deprecated.

Processes the image, reads the possible contents of the bar code and returns the single possible decoding (mono-symbology) or the most likely one (multi-symbology) or a reading error code.

Namespace: Euresys::Open_eVision

[C++]

```
std::string Read(
 EROIBW8* sourceImage
 )
```

Parameters

sourceImage

The image containing the bar code.

Remarks

Processing the image means finding the symbol (in case of automatic location mode) and retrieving its descriptive parameters. When decoding other than the most likely one are also required, a call to the EBarCode::Detect function followed by a EBarCode::Decode should be used.

EBarCode::SetRectangle

This property is deprecated.

Sets the nominal position (center coordinates), size and rotation angle of the symbol bounding box according to a known rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void SetRectangle(const ERectangle& rectangle)
```

Remarks

An ERectangle object is characterized by its center coordinates, its size and its rotation angle.

EBarCode::GetRectangleShape

This property is deprecated.

Get the rectangle shape associated with the EBarCode.

Namespace: Euresys::Open_eVision

[C++]

```
const ERectangleShape& GetRectangleShape() const
```

EBarCode::GetRelativeReadingSizeX

This property is deprecated.

Reading area width, relative to the symbol extent.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeReadingSizeX() const
```

EBarCode::GetRelativeReadingSizeY

This property is deprecated.

Reading area height, relative to the symbol extent.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeReadingSizeY() const
```

EBarCode::GetRelativeReadingX

This property is deprecated.

Reading area abscissa, relative to the symbol position.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeReadingX() const
```

EBarCode::GetRelativeReadingY

This property is deprecated.

Reading area ordinate, relative to the symbol position.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeReadingY() const
```

EBarCode::Save

This method is deprecated.

Loads an EBarCode.The given EBarCode must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path,
 bool daughters
 )

void Save(
 ESerializer* serializer,
 bool daughters
 )
```

Parameters

path
The file path.

daughters
Indicates if the save must be done on the whole hierarchy or just this object.

serializer
Pointer to the ESerializer created for writing.

EBarCode::SetCenterXY

This method is deprecated.

Sets the center coordinates of a EBarCode object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the EBarCode object.

centerY

Center coordinates of the EBarCode object.

EBarCode::SetReadingCenter

This method is deprecated.

Sets the reading area center coordinates, relative to the symbol position and extent.

Namespace: Euresys::Open_eVision

```
[C++]
void SetReadingCenter(
 float relativeX,
 float relativeY
 )
```

Parameters

relativeX

Reading area center abscissa, relative to the symbol position and extent.

relativeY

Reading area center ordinate, relative to the symbol position and extent.

EBarCode::SetReadingSize

This method is deprecated.

Sets the reading area size, relative to the symbol extent.

Namespace: Euresys::Open_eVision

```
[C++]
void SetReadingSize(
 float relativeSizeX,
 float relativeSizeY
 )
```

Parameters

relativeSizeX

Reading area width, relative to the symbol extent.

relativeSizeY

Reading area height, relative to the symbol extent.

EBarCode::SetSize

This method is deprecated.

Sets the size of a EBarCode object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSize(
 float sizeX,
 float sizeY
 )
```

Parameters

sizeX

Nominal size X of the EBarCode object. Default values is 100.

sizeY

Nominal size Y of the EBarCode object. Default values is 100.

Remarks

A EBarCode object is fully defined knowing its nominal position (given by the coordinates of its center), its nominal size, its rotation angle and its outline tolerance. By default, the width and height values are 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EBarCode::GetSizeX

This property is deprecated.

X size of the EBarCode

Namespace: Euresys::Open_eVision

[C++]

```
float GetSizeX() const
```

EBarCode::GetSizeY

This property is deprecated.

Y size of the EBarCode

Namespace: Euresys::Open_eVision

```
[C++]
float GetSizeY() const
```

EBarCode::GetStandardSymbologies

EBarCode::SetStandardSymbologies

This property is deprecated.

Enabled symbologies belonging to the group of standard symbologies.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetStandardSymbologies() const
void SetStandardSymbologies(OEV_UINT32 un32StandardSymbologies)
```

Remarks

Due to the large number of supported symbologies, they have been gathered in two groups. For more information about these groups, see ESymbologies.

EBarCode::GetThicknessRatio

EBarCode::SetThicknessRatio

This property is deprecated.

Bars thickness ratio.

Namespace: Euresys::Open_eVision

```
[C++]
float GetThicknessRatio() const
void SetThicknessRatio(float f32ThicknessRatio)
```

Remarks

This property is relevant in case of binary codes only. It corresponds to the ratio of a thin bar width over a thick bar width, and should range from 1.5 to 3.

EBarCode::GetVerifyChecksum

EBarCode::SetVerifyChecksum

This property is deprecated.

"VerifyChecksum" mode.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetVerifyChecksum() const
void SetVerifyChecksum(bool bVerifyChecksum)
```

Remarks

The "VerifyChecksum" mode enables or disables verification of the checksum character. That verification mode is set in the same way for all enabled symbologies. It is worth noting that checksum may be present or not in the bar code, and the user may verify or not checksum validity. These two circumstances are independent, and give rise to four modes of operation. The verification process will return an "invalid checksum" error in case of bad checksum character. This error can also be generated if there is no checksum in the bar code. In the other case, when checksum are not verified, no error will occur, and the process will continue silently. When the "VerifyChecksum" mode is enabled, the returned decoded string does not contain the checksum character(s). Conversely, when the verification process is disabled, the checksum character(s) are concatenated to the encoded information.

4.26. EBarCode Class

Represents a 1D Bar Code.

Namespace: Euresys::Open_eVision::EasyBarCode2

Methods

DrawPosition	Draws the BarCode Position.
DrawPositionWithCurrentPen	Draws the BarCode Position using the pen currently set in the graphical context.
EBarCode	Constructs a EBarCode context.
GetChecksumOK	Checksum validity status.
GetDecodedString	Decoded bar code string.
GetGradingParameters	Grading Parameters according to ISO15416.
GetPosition	Position of the barcode.
GetSymbologies	Returns all compatible symbologies, in order of likelyhood.
GetSymbology	Returns the most likely compatible symbology.

HasGradingParameters	Returns whether the EBarCode contains Grading Parameters.
operator=	Copies all the data from another EBarCode object into the current EBarCode object.

EBarCode::DrawPosition

Draws the BarCode Position.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
void DrawPosition(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawPosition(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

Handle of the graphics context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBarCode::DrawPositionWithCurrentPen

This method is deprecated.

Draws the BarCode Position using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
void DrawPositionWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBarCode::EBarCode

Constructs a EBarCode context.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
void EBarCode(
 const EBarCode& other
 )
```

Parameters

other

Another EBarCode object to be copied in the new EBarCode object.

Remarks

The default constructor constructs an uninitialized EBarCode object. All properties are initialized to their respective default values. The copy constructor constructs a EBarCode context based on a pre-existing EBarCode object. All properties and internal data are copied.

EBarCode::GetChecksumOK

Checksum validity status.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
bool GetChecksumOK(
 )
bool GetChecksumOK(
 Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies symbology
 )
```

Parameters

symbology

Chosen symbology

Remarks

If the symbology parameter is omitted, the property returns the value pertaining to the barcode's most likely symbology.

EBarCode::GetDecodedString

Decoded bar code string.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
std::string GetDecodedString(
 bool includeChecksum
 )
std::string GetDecodedString(
 Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies symbology,
 bool includeChecksum
 )
```

Parameters

includeChecksum

Indicates if the returned string should include the check character (default: true)

symbology

Chosen symbology

Remarks

If the symbology parameter is omitted, the property returns the value pertaining to the barcode's most likely symbology. If you choose to exclude the check character and you use a symbology where it is optional, be sure it is included as the character corresponding to the checksum position will be removed in all cases. When the symbology is EBarCodeSymbologies_Gs1_128, no checksum is returned no matter the value of includeChecksum as it would break the parsing of the machine readable Gs1 code.

EBarCode::GetGradingParameters

Grading Parameters according to ISO15416.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
EBarCodeGradingParameters GetGradingParameters() const
```

Remarks

If the most likely symbology does not support grading parameters or EBarCodeReader::ComputeGrading is false, an exception is thrown.

EBarCode::HasGradingParameters

Returns whether the EBarCode contains Grading Parameters.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool HasGradingParameters(
 )
```

EBarCode::operator=

Copies all the data from another EBarCode object into the current EBarCode object.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
EBarCode& operator=(
 const EBarCode& other
 )
```

Parameters

other

EBarCode object to be copied

EBarCode::GetPosition

Position of the barcode.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
EQuadrangle GetPosition() const
```

EBarCode::GetSymbologies

Returns all compatible symbologies, in order of likelyhood.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies> GetSymbologies()
const
```

EBarCode::GetSymbology

Returns the most likely compatible symbology.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies GetSymbology() const
```

4.27. EBarCodeGrid Class

-

Namespace: Euresys::Open_eVision::EasyBarCode2

Methods

EBarCodeGrid	Creates an EBarCodeGrid object.
GetCellEnabled	Returns true if Cell is enabled and false otherwise
GetNumCols	Returns the number of columns in the grid
GetNumRows	Returns the number of rows in the grid
GetResults	Returns the detected Barcodes
operator=	Assignment operator
SetEnableAll	Enable/Disable all cells
SetEnableCell	Enable/Disable Cell
SetEnableColumn	Enable/Disable Column
SetEnableRow	Enable/Disable Row

EBarCodeGrid::EBarCodeGrid

Creates an EBarCodeGrid object.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
void EBarCodeGrid(
 )
void EBarCodeGrid(
 const EBarCodeGrid& other
 )
void EBarCodeGrid(
 OEV_UINT32 numCols,
 OEV_UINT32 numRows
 )
```

Parameters

other
The reference EBarCodeGrid instance to copy this one from.

numCols
The number of columns in the grid.

numRows
The number of rows in the grid.

EBarCodeGrid::SetEnableAll

Enable/Disable all cells

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void SetEnableAll(bool enable)
```

Remarks

By default, all grid cells are enabled.

EBarCodeGrid::GetCellEnabled

Returns true if Cell is enabled and false otherwise

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetCellEnabled(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
```

Parameters

column

-

row

-

Remarks

By default, all grid cells are enabled.

EBarCodeGrid::GetResults

Returns the detected Barcodes

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCode> GetResults(
 )

std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCode> GetResults(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
```

Parameters

column

The column of a cell.

row

The row of a cell.

EBarCodeGrid::GetNumCols

Returns the number of columns in the grid

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT32 GetNumCols() const
```

EBarCodeGrid::GetNumRows

Returns the number of rows in the grid

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT32 GetNumRows() const
```

EBarCodeGrid::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
EBarCodeGrid& operator=(
 const EBarCodeGrid& other
 )
```

Parameters

other

The EBarCodeGrid instance to assign.

EBarCodeGrid::SetEnableCell

Enable/Disable Cell

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void SetEnableCell(
 OEV_UINT32 column,
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

column

-

row

-

enable

-

Remarks

By default, all grid cells are enabled.

EBarCodeGrid::SetEnableColumn

Enable/Disable Column

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void SetEnableColumn(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

EBarCodeGrid::SetEnableRow

Enable/Disable Row

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void SetEnableRow(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

4.28. EBarCodeReader Class

Represents a 1D Bar Code Reading Context.

Namespace: Euresys::Open_eVision::EasyBarCode2

Methods

DisableAllSymbologies	Disables all symbologies.
EBarCodeReader	Constructs a EBarCodeReader context.
EnableAllSymbologies	Enables all supported symbologies, with the exception of EBarCodeSymbologies_CodeStk, EBarCodeSymbologies_PharmacodeOneTrack and EBarCodeSymbologies_BinaryCode because they might interfere with other symbologies.
EnableDefaultSymbologies	Enables default symbologies.
EnableSymbologies	Enables a set of symbologies.
EnableSymbology	Enables a single symbology.
GetComputeGrading	Enables or disables the computation of the grades of the read barcodes according to ISO 15416. At the moment, grading is only supported for EBarCodeSymbologies_Code128, EBarCodeSymbologies_Gs1_128 and EBarCodeSymbologies_Ean13.
GetEnabledSymbologies	Enabled symbologies.
GetEnablePermissiveDecoding	Enables or disables the decoding of codes containing partially incorrect information. In practice, when this option is set, the reader assumes a character might be badly printed and uses the checksum to correct that character. In rare cases, this option may lead to reading false positives codes. It has no effect for symbologies in which a checksum is not mandatory.
GetLearningPerformed	Returns whether a learning has been performed on the EBarCodeReader.
GetMaxNumCodes	Maximum number of barcodes to find in a single Image/ROI.

GetMinModuleSize	Sets the minimal module size. This is the size in pixels of the smallest module of the barcode in the image. Setting this parameter is helpful when trying to detect small barcodes in large images. In most cases, the min module size is to be set to an integer value, either 1, 2 or 3. Nevertheless, we allow intermediary floating-point values, like 2.5, because the processing time of the EBarCodeReader::Read method is strongly dependent of the minModuleSize.
GetReadingOrientation	Some symbologies do not specify start and stop patterns, namely: EBarCodeSymbologies_CodeStk, EBarCodeSymbologies_PharmacodeOneTrack and EBarCodeSymbologies_BinaryCode. In these cases, we do not know if we should decode the barcode from left to right or from right to left. EBarCodeReader::ReadingOrientation solves this problem.
GetTimeOut	Time-out for the EBarCodeReader::Read method.
GetUseMinModuleSize	Enables or disables the usage of the minimal module size, see EBarCodeReader::MinModuleSize to set it.
GetValidateBarCode	Enables/Disables all barcode validations. Deprecated, use EBarCodeReader::ValidateMandatoryChecksum, EBarCodeReader::ValidateOptionalChecksum and EBarCodeReader instead.
GetValidateMandatoryChecksum	Enables/Disables checksum validation for all symbologies for which a checksum is mandatory, for example EBarCodeSymbologies_Code128. See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm
GetValidateOptionalChecksum	Enables/Disables checksum validation for all symbologies for which a checksum is optional, for example EBarCodeSymbologies_Code39. See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm
GetValidatePharmacode	Enables/Disables validation for the EBarCodeSymbologies_PharmacodeOneTrack symbology.
GetValidateWithChecksum	Enables/Disables all barcode validations. Deprecated, use EBarCodeReader::ValidateMandatoryChecksum, EBarCodeReader::ValidateOptionalChecksum and EBarCodeReader instead.
Learn	Learns the best options to use to read the given Images/ROIs. The EBarCodeReader::MinModuleSize is modified as well as the scales at which we detect barcodes. The other options (symbologies, checksum verification, max number of codes, ...) must be set by the user so that the codes are read correctly.
Load	Load the reader configuration.
operator=	Copies all the data from another EBarCodeReader object into the current EBarCodeReader object.
Read	Finds and reads all the barcodes in the provided Image/ROI.

ResetLearning	Forgets the learned parameter settings and resets them to their default values.
Save	Save the reader configuration.
SetComputeGrading	Enables or disables the computation of the grades of the read barcodes according to ISO 15416. At the moment, grading is only supported for EBarCodeSymbologies_Code128, EBarCodeSymbologies_Gs1_128 and EBarCodeSymbologies_Ean13.
SetEnablePermissiveDecoding	Enables or disables the decoding of codes containing partially incorrect information. In practice, when this option is set, the reader assumes a character might be badly printed and uses the checksum to correct that character. In rare cases, this option may lead to reading false positives codes. It has no effect for symbologies in which a checksum is not mandatory.
SetMaxNumCodes	Maximum number of barcodes to find in a single Image/ROI.
SetMinModuleSize	Sets the minimal module size. This is the size in pixels of the smallest module of the barcode in the image. Setting this parameter is helpful when trying to detect small barcodes in large images. In most cases, the min module size is to be set to an integer value, either 1, 2 or 3. Nevertheless, we allow intermediary floating-point values, like 2.5, because the processing time of the EBarCodeReader::Read method is strongly dependent of the minModuleSize.
SetReadingOrientation	Some symbologies do not specify start and stop patterns, namely: EBarCodeSymbologies_CodeStk, EBarCodeSymbologies_PharmacodeOneTrack and EBarCodeSymbologies_BinaryCode. In these cases, we do not know if we should decode the barcode from left to right or from right to left. EBarCodeReader::ReadingOrientation solves this problem.
SetTimeOut	Time-out for the EBarCodeReader::Read method.
SetUseMinModuleSize	Enables or disables the usage of the minimal module size, see EBarCodeReader::MinModuleSize to set it.
SetValidateBarCode	Enables/Disables all barcode validations. Deprecated, use EBarCodeReader::ValidateMandatoryChecksum, EBarCodeReader::ValidateOptionalChecksum and EBarCodeReader instead.
SetValidateMandatoryChecksum	Enables/Disables checksum validation for all symbologies for which a checksum is mandatory, for example EBarCodeSymbologies_Code128. See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm
SetValidateOptionalChecksum	Enables/Disables checksum validation for all symbologies for which a checksum is optional, for example EBarCodeSymbologies_Code39. See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm

SetValidatePharmacode	Enables/Disables validation for the EBarCodeSymbologies_PharmacodeOneTrack symbology.
SetValidateWithChecksum	Enables/Disables all barcode validations. Deprecated, use EBarCodeReader::ValidateMandatoryChecksum, EBarCodeReader::ValidateOptionalChecksum and EBarCodeReader instead.

EBarCodeReader::GetComputeGrading

EBarCodeReader::SetComputeGrading

Enables or disables the computation of the grades of the read barcodes according to ISO 15416.
At the moment, grading is only supported for Code128, Gs1_128 and Ean13.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetComputeGrading() const
void SetComputeGrading(bool enable)
```

Remarks

ComputeGrading is disabled by default.

EBarCodeReader::DisableAllSymbologies

Disables all symbologies.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void DisableAllSymbologies(
 )
```

EBarCodeReader::EBarCodeReader

Constructs a EBarCodeReader context.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void EBarCodeReader(
 )
```

```
void EBarCodeReader(
 const EBarCodeReader& other
 )
```

Parameters

other

Another EBarCodeReader object to be copied in the new EBarCodeReader object.

Remarks

The default constructor constructs an uninitialized EBarCodeReader object. All properties are initialized to their respective default values. The copy constructor constructs a EBarCodeReader context based on a pre-existing EBarCodeReader object. All properties and internal data are copied.

EBarCodeReader::EnableAllSymbologies

Enables all supported symbologies, with the exception of CodeStk, PharmacodeOneTrack and BinaryCode because they might interfere with other symbologies.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void EnableAllSymbologies(
 )
```

EBarCodeReader::EnableDefaultSymbologies

Enables default symbologies.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void EnableDefaultSymbologies(
 )
```

EBarCodeReader::GetEnabledSymbologies

Enabled symbologies.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies>
GetEnabledSymbologies() const
```

EBarCodeReader::GetEnablePermissiveDecoding

EBarCodeReader::SetEnablePermissiveDecoding

Enables or disables the decoding of codes containing partially incorrect information.
In practice, when this option is set, the reader assumes a character might be badly printed and uses the checksum to correct that character. In rare cases, this option may lead to reading false positives codes. It has no effect for symbologies in which a checksum is not mandatory.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetEnablePermissiveDecoding() const
void SetEnablePermissiveDecoding(bool enable)
```

Remarks

Permissive decoding is enabled by default.

EBarCodeReader::EnableSymbologies

Enables a set of symbologies.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void EnableSymbologies(
 const std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies>&
symbologies
 )
```

Parameters

symbologies

-

EBarCodeReader::EnableSymbology

Enables a single symbology.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void EnableSymbology(
 Euresys::Open_eVision::EasyBarCode2::EBarCodeSymbologies symbology
 )
```

Parameters

symbology

-

EBarCodeReader::Learn

Learns the best options to use to read the given Images/ROIs. The EBarCodeReader::MinModuleSize is modified as well as the scales at which we detect barcodes. The other options (symbologies, checksum verification, max number of codes, ...) must be set by the user so that the codes are read correctly.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
float Learn(
 const std::vector<Euresys::Open_eVision::EROIBW8>& rois,
 bool keepDefaultScales,
 bool addAllScales
 )
float Learn(
 const std::vector<Euresys::Open_eVision::EImageBW8>& images,
 bool keepDefaultScales,
 bool addAllScales
 )
```

Parameters

rois

ROIs in which to find the barcodes.

keepDefaultScales

if true, Learning does not remove the scales used by default even if they are useless for the given images. Default: true

addAllScales

if true, Learning adds all of the scales at which a code is detected. Otherwise, the smallest number of scales required to read all of the images is added. Default: true

images

Images in which to find the barcodes.

Remarks

Adding more scales does not necessarily makes the reading slower, as processing stops as soon as we find the required number of codes. On the other hand, having too few scales can make the reading fail. The is why the default scales are kept and all the interesting scales are added by default.

EBarCodeReader::GetLearningPerformed

Returns whether a learning has been performed on the EBarCodeReader.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetLearningPerformed() const
```

Remarks

Note that after a failed call to EBarCodeReader::Learn (the value of 0 was returned), calling this method will returns false.

EBarCodeReader::Load

Load the reader configuration.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void Load(
 const std::string& file
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

file

File path.

serializer

Serializer. Must be in read mode.

EBarCodeReader::GetMaxNumCodes

EBarCodeReader::SetMaxNumCodes

Maximum number of barcodes to find in a single Image/ROI.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT32 GetMaxNumCodes() const

void SetMaxNumCodes(OEV_UINT32 max)
```

Remarks

By default, this parameter is set to 1.

EBarCodeReader::GetMinModuleSize

EBarCodeReader::SetMinModuleSize

Sets the minimal module size. This is the size in pixels of the smallest module of the barcode in the image. Setting this parameter is helpful when trying to detect small barcodes in large images.
In most cases, the min module size is to be set to an integer value, either 1, 2 or 3. Nevertheless, we allow intermediary floating-point values, like 2.5, because the processing time of the EBarCodeReader::Read method is strongly dependent of the minModuleSize.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
float GetMinModuleSize() const

void SetMinModuleSize(float minModuleSize)
```

Remarks

Setting the minModuleSize automatically enables its use, see EBarCodeReader::UseMinModuleSize. If the min module size is used but never set, it is considered to be 1 pixel.

EBarCodeReader::operator=

Copies all the data from another EBarCodeReader object into the current EBarCodeReader object.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
EBarCodeReader& operator=(
 const EBarCodeReader& other
 )
```

Parameters

other

EBarCodeReader object to be copied.

EBarCodeReader::Read

Finds and reads all the barcodes in the provided Image/ROI.

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCode> Read(
 const EROIBW8& img
 )
std::vector<Euresys::Open_eVision::EasyBarCode2::EBarCode> Read(
 const EROIBW8& img,
 const ERegion& region
 )
EBarCodeGrid Read(
 const EROIBW8& field,
 const ERectangleRegion& area,
 int numCellsX,
 int numCellsY,
 float extension
 )
EBarCodeGrid Read(
 const EROIBW8& field,
 const ERectangleRegion& area,
 const EBarCodeGrid& grid,
 float extension
 )
```

Parameters

img

Image/ROI in which to find the barcodes.

region

Optional region used to reduce the search domain.

field

-

area

Rectangular Region used as the full grid area

numCellsX

Number of grid cells in the X direction

numCellsY

Number of grid cells in the Y direction

extension

Extension of the grid cells to allow cell overlap. For instance, 0.0f means no extension and 0.1f means a 10% cell size extension. default: 0.0f

grid

Grid with cell disabling capabilities

Remarks

The grid overload allows you to disable some cells of the grid if those cells are not supposed to contain barcodes. See the EBarCodeGrid class documentation for more information.

EBarCodeReader::GetReadingOrientation

EBarCodeReader::SetReadingOrientation

Some symbologies do not specify start and stop patterns, namely: CodeStk, PharmacodeOneTrack and BinaryCode. In these cases, we do not know if we should decode the barcode from left to right or from right to left. EBarCodeReader::ReadingOrientation solves this problem.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
Euresys::Open_eVision::EasyBarCode2::EReadingOrientation GetReadingOrientation() const

void SetReadingOrientation(Euresys::Open_eVision::EasyBarCode2::EReadingOrientation
orientation)
```

Remarks

The default orientation is EReadingOrientation_LeftToRight.

EBarCodeReader::ResetLearning

Forgets the learned parameter settings and resets them to their default values.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void ResetLearning(
 )
```

EBarCodeReader::Save

Save the reader configuration.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
void Save(
 const std::string& file
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

file
File path.

serializer
Serializer. Must be in write mode.

EBarCodeReader::GetTimeOut

EBarCodeReader::SetTimeOut

Time-out for the EBarCodeReader::Read method.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT64 GetTimeOut() const
void SetTimeOut(OEV_UINT64 value)
```

Remarks

If the processing time of one of these functions becomes longer than the set time-out, the processing is stopped and an exception is thrown.
In that case, the error code of the exception is EError_TimeoutReached.
The time-out is set in microseconds.
This time-out is not a real time-out.
The processing is stopped as soon as possible after the time-out has been reached.
This means that the time elapsed effectively in the method can be greater than the time-out itself.

EBarCodeReader::GetUseMinModuleSize

EBarCodeReader::SetUseMinModuleSize

Enables or disables the usage of the minimal module size, see EBarCodeReader::MinModuleSize to set it.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetUseMinModuleSize() const
void SetUseMinModuleSize(bool useMinModuleSize)
```

Remarks

The minimal module size is not used by default.

EBarCodeReader::GetValidateBarCode

EBarCodeReader::SetValidateBarCode

Enables/Disables all barcode validations.
Deprecated, use EBarCodeReader::ValidateMandatoryChecksum, EBarCodeReader::ValidateOptionalChecksum and EBarCodeReader instead.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetValidateBarCode() const

void SetValidateBarCode(bool validate)
```

Remarks

If this validation is enabled, barcodes with an erroneous checksum will not be returned.
For the PharmaCodeOneTrack symbology, no checksum is computed. We instead validate that the barcode is coherent along its height and that the relative width of the bars/space are coherent with the symbology's specification. This allow to reduce the number of false positives significantly.

Some symbologies have an optional checksum, for example EBarCodeSymbologies_Code39, for these symbologies, enforcing checksum validation comes at the risk of removing good barcodes that do not have a checksum. See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm.

By default, validation is enforced when checksum is mandatory for the symbology and for Pharmacodes.

EBarCodeReader::GetValidateMandatoryChecksum

EBarCodeReader::SetValidateMandatoryChecksum

Enables/Disables checksum validation for all symbologies for which a checksum is mandatory, for example Code128.
See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetValidateMandatoryChecksum() const

void SetValidateMandatoryChecksum(bool validate)
```

Remarks

If validation is enabled for symbologies with a mandatory checksum, barcodes of these symbologies with an erroneous checksum will not be returned.
By default, validation is enforced when checksum is mandatory.

EBarCodeReader::GetValidateOptionalChecksum

EBarCodeReader::SetValidateOptionalChecksum

Enables/Disables checksum validation for all symbologies for which a checksum is optional, for example Code39.
See complete list at https://documentation.euresys.com/Products/OPEN_EVISION/OPEN_EVISION/en-us/Content/00_Home/List_of_Supported_Codes.htm

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetValidateOptionalChecksum() const

void SetValidateOptionalChecksum(bool validate)
```

Remarks

If validation is enabled for symbologies with an optional checksum, barcodes of these symbologies with an erroneous checksum will not be returned.
By default, validation is not enforced when checksum is optional to avoid removing correct codes.

EBarCodeReader::GetValidatePharmacode

EBarCodeReader::SetValidatePharmacode

Enables/Disables validation for the PharmacodeOneTrack symbology.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetValidatePharmacode() const

void SetValidatePharmacode(bool validate)
```

Remarks

Pharmacodes are validated by checking that the barcode is coherent along its height and that the relative width of the bars/space are coherent with the symbology's specification. This allows to reduce the number of false positives significantly.
By default, validation is enforced for Pharmacodes.

EBarCodeReader::GetValidateWithChecksum

EBarCodeReader::SetValidateWithChecksum

This property is deprecated.

Enables/Disables all barcode validations.
Deprecated, use EBarCodeReader::ValidateMandatoryChecksum, EBarCodeReader::ValidateOptionalChecksum and EBarCodeReader instead.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
bool GetValidateWithChecksum() const

void SetValidateWithChecksum(bool validate)
```

4.29. EBaseROI Class

This represents the abstract base class for all ROI and image classes.

Derived Class (es):
EROIBW1
EROIBW16EROIBW32EROIBW32fEROIBW8EROIC15EROIC16EROIC24EROIC24AEROIC48

Namespace: Euresys::Open_eVision

Methods

Attach	This method attaches the ROI to another ROI or image. Only ROIs can be attached. An image can never be attached to another image or ROI. Optionally, the ROI can be moved and resized after attachment by supplying additional parameters.
CopyTo	Copies all the data of the current EBaseROI object into another EBaseROI object and returns it.
CropToImage	This method reduces the ROI size so that it is completely contained within its topmost parent image bounds (if such an image exists at the top of the hierarchy). This means that, after calling this method, either the ROI has a zero size, or all of its pixels map to valid pixels of the underlying image buffer. If the ROI is already contained within its parent image bounds, then this method does nothing.
Drag	Moves the specified handle to a new position and updates all placement parameters of the ROI.
Draw	Draws an ROI/image in a device context.
DrawFrame	Draws a rectangular frame around an image or ROI.

DrawFrameWithCurrentPen	Draws a rectangular frame around an image or ROI.
GetAuthor	Gets or sets the Author attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
GetBaseTopParent	Returns the image at the top of the hierarchy, or NULL if there is no image on top.
GetBitsPerPixel	Gets the number of storage bits per pixel.
GetColorSystem	Gets or sets the color system used by this image, as defined by the EColorSystem enumeration.
GetColPitch	Gets the pitch of a column (number of bytes between two horizontally adjacent pixels). For BW1 (one bit per pixel) images, this value is undefined.
GetComment	Gets or sets the Comment attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
GetDate	Gets or sets the Date attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
GetHeight	Gets or sets the height of the ROI.
GetImagePtr	Returns a pointer to the pixel at given coordinates within the image/ROI.
GetOrgX	Gets or sets the x-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.
GetOrgY	Gets or sets the y-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.
GetParent	Returns the hierarchical parent of this object.
GetPlanesPerPixel	Gets the number of color components in each pixel of the ROI/image.
GetRowPitch	Gets the pitch of a row (the number of bytes between two vertically adjacent pixels). This is also valid for BW1 images (one bit per pixel).
GetSubBaseROIs	Returns all the children, and possibly recursively all their children, too.
GetTitle	Gets or sets the Title attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
GetTotalHeight	Gets the height, in pixels, of the ROI topmost parent.
GetTotalOrgX	Gets the x-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.
GetTotalOrgY	Gets the y-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.
GetTotalWidth	Gets the width, in pixels, of the ROI topmost parent.

GetType	Gets the ROI/image pixel type, as defined by the EImageType enumeration.
GetWidth	Gets or sets the width of the ROI.
HasSubROI	Tests whether this object has a given attached ROI.
HitTest	Detects if the cursor is placed over one of the dragging handles.
IsAnROI	Tests whether this object is an ROI or an image.
IsVoid	Tests whether if the topmost parent image of this hierarchy has a zero size.
Load	Restores an image stored in the given file.
Save	Saves the EBaseROI object to the given file.
SaveJpeg	Saves the EBaseROI object to the given file, in JPEG format.
SaveJpeg2K	Saves the EBaseROI object to the given file, in JPEG 2000 format.
SavePng	Saves the EBaseROI object to the given file, in PNG format.
SetAuthor	Gets or sets the Author attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
SetColorSystem	Gets or sets the color system used by this image, as defined by the EColorSystem enumeration.
SetComment	Gets or sets the Comment attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
SetDate	Gets or sets the Date attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
SetHeight	Gets or sets the height of the ROI.
SetImagePtr	Sets the pointer to an externally allocated image buffer.
SetOrgX	Gets or sets the x-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.
SetOrgY	Gets or sets the y-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.
SetPlacement	Sets the placement of an ROI, relative to its parent ROI/image.
SetSize	Sets the width and height of an ROI/image.
SetTitle	Gets or sets the Title attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.
SetWidth	Gets or sets the width of the ROI.

EBaseROI::Attach

This method attaches the ROI to another ROI or image.
Only ROIs can be attached. An image can never be attached to another image or ROI.
Optionally, the ROI can be moved and resized after attachment by supplying additional parameters.

Namespace: Euresys::Open_eVision

[C++]

```
void Attach(
 EBaseROI* parent
 )

void Attach(
 EBaseROI* parent,
 int orgX,
 int orgY,
 int width,
 int height
 )
```

Parameters

parent
ROI or image on which to attache the ROI.

orgX
When specified, sets the new x-coordinate of the ROI top-left corner.

orgY
When specified, sets the new y-coordinate of the ROI top-left corner.

width
When specified, sets the new width of the ROI.

height
When specified, sets the new height of the ROI.

EBaseROI::GetAuthor

EBaseROI::SetAuthor

Gets or sets the Author attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetAuthor() const

void SetAuthor(const std::string& author)
```

EBaseROI::GetBaseTopParent

Returns the image at the top of the hierarchy, or NULL if there is no image on top.

Namespace: Euresys::Open_eVision

[C++]

```
EBaseROI* GetBaseTopParent()
```

EBaseROI::GetBitsPerPixel

Gets the number of storage bits per pixel.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetBitsPerPixel() const
```

EBaseROI::GetColorSystem

EBaseROI::SetColorSystem

Gets or sets the color system used by this image, as defined by the EColorSystem enumeration.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EColorSystem GetColorSystem() const
void SetColorSystem(Euresys::Open_eVision::EColorSystem colorSystem)
```

Remarks

Upon object creation, a default color system is set, compatible with the ROI/image type (EColorSystem_GrayLevel for gray-level types and EColorSystem_Rgb for color types).
The color system associated to an image is mainly relevant when working on color images.
See EasyColor (FG) for more information.

EBaseROI::GetColPitch

Gets the pitch of a column (number of bytes between two horizontally adjacent pixels). For BW1 (one bit per pixel) images, this value is undefined.

Namespace: Euresys::Open_eVision

[C++]

```
int GetColPitch() const
```

EBaseROI::GetComment

EBaseROI::SetComment

Gets or sets the Comment attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetComment() const

void SetComment(const std::string& comment)
```

EBaseROI::CopyTo

Copies all the data of the current EBaseROI object into another EBaseROI object and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EBaseROI& dest
 )

void CopyTo(
 EBaseROI* dest
 )
```

Parameters

dest

An EBaseROI object in which the current EBaseROI object data have to be copied.

Remarks

This method copies all the object date to the destination object. The attached ROIs are copied recursively and attached to the destination object. They will be deleted automatically when the the destination object is deleted.
When the buffer of the source image has been provided by a call to SetImagePtr, the pointer will be copied into the destination image. Both images will thus refer the same external buffer. Deprecation notice: The overload taking a pointer is deprecated.

EBaseROI::CropToImage

This method reduces the ROI size so that it is completely contained within its topmost parent image bounds (if such an image exists at the top of the hierarchy). This means that, after calling this method, either the ROI has a zero size, or all of its pixels map to valid pixels of the underlying image buffer. If the ROI is already contained within its parent image bounds, then this method does nothing.

Namespace: Euresys::Open_eVision

[C++]

```
void CropToImage(
 )
```

EBaseROI::GetDate

EBaseROI::SetDate

Gets or sets the Date attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetDate() const

void SetDate(const std::string& date)
```

EBaseROI::Drag

Moves the specified handle to a new position and updates all placement parameters of the ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 Euresys::Open_eVision::EDragHandle dragHandle,
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

dragHandle

Handle identifier, as defined by EDragHandle. The value returned by EBaseROI::HitTest should be used.

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

If zooming and/or panning were used when drawing the ROI, the same values must be used with EBaseROI::HitTest and EBaseROI::Drag.

EBaseROI::Draw

Draws an ROI/image in a device context.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

c24Vector

When supplied, this parameter allows using a LUT that maps from BW8 to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from BW8 to BW8 when drawing.

Remarks

An ROI/image can be drawn (its pixels rendered) using a device context.
The horizontal and vertical zooming factors can be different and must be contained in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBaseROI::DrawFrame

Draws a rectangular frame around an image or ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawFrame(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EFramePosition framePosition,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawFrame(
 EDrawAdapter* graphicContext,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawFrame(
 HDC graphicContext,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawFrame(
 HDC graphicContext,
 Euresys::Open_eVision::EFramePosition framePosition,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawFrame(
 HDC graphicContext,
 const ERGBColor& color,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

framePosition

Positioning of the frame relative to the ROI.

handles

true if handles are to be drawn.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

Color in which to draw the frame.

Remarks

A frame can be drawn (using a device context) around an image or ROI, possibly with 9 sizing handles.
A suitable default pen is used (see EBaseROI::DrawFrameWithCurrentPen if you wish to use the pen currently selected into the device context).
Zooming and panning are possible. Please note that panning is applied *before* zooming.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBaseROI::DrawFrameWithCurrentPen

This method is deprecated.

Draws a rectangular frame around an image or ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawFrameWithCurrentPen(
 HDC graphicContext,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawFrameWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EFramePosition framePosition,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

handles

true if handles are to be drawn.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

framePosition

Positioning of the frame relative to the ROI.

Remarks

A frame can be drawn (using a device context) around an image or ROI, possibly with 9 sizing handles.
The current device context pen is used. Zooming and panning are possible. Please note that panning is applied *before* zooming.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBaseROI::GetImagePtr

Returns a pointer to the pixel at given coordinates within the image/ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetImagePtr(
 int x,
 int y
 )

const void* GetImagePtr(
 int x,
 int y
 )

void* GetImagePtr(
 )

const void* GetImagePtr(
 )
```

Parameters

x
The pixel x-coordinate.

y
The pixel y-coordinate.

Remarks

This methods returns the memory address of the byte that contains the pixel (or address that contains the first byte of the pixel if it is bigger than one byte).
If the pixel coordinates are not specified, the method returns the address of the top-left pixel of the ROI/image.

EBaseROI::GetSubBaseROIs

Returns all the children, and possibly recursively all their children, too.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EBaseROI*> GetSubBaseROIs(
 bool recursive
 )
std::vector<const Euresys::Open_eVision::EBaseROI*> GetSubBaseROIs(
 bool recursive
 )
```

Parameters

recursive

true to retrieve all sub-ROIs recursively. false otherwise.

EBaseROI::HasSubROI

Tests whether this object has a given attached ROI.

Namespace: Euresys::Open_eVision

[C++]

```
bool HasSubROI(
 const EBaseROI* subROI
 )
```

Parameters

subROI

Sub ROI to find.

EBaseROI::GetHeight

EBaseROI::SetHeight

Gets or sets the height of the ROI.

Namespace: Euresys::Open_eVision

[C++]

```
int GetHeight() const
void SetHeight(int h)
```

Remarks

The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI::HitTest

Detects if the cursor is placed over one of the dragging handles.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EDragHandle HitTest(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
x-coordinate of the mouse cursor.

y
y-coordinate of the mouse cursor.

zoomX
Horizontal zoom factor. By default, true scale is used.

zoomY
Vertical zoom factor. By default, true scale is used.

panX
Horizontal panning value expressed in pixels. By default, no panning occurs.

panY
Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Returns a handle identifier, as defined by EDragHandle.
If zooming and/or panning were used when drawing the ROI, the same values must be used with EBaseROI::HitTest and EBaseROI::Drag.

EBaseROI::IsAnROI

Tests whether this object is an ROI or an image.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsAnROI(
 )
```

EBaseROI::GetIsVoid

Tests whether if the topmost parent image of this hierarchy has a zero size.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsVoid() const
```

Remarks

For an image, this method returns true if the image size is zero.
For an ROI, this method returns true if the topmost parent image size is zero or if there is no topmost image.

EBaseROI::Load

Restores an image stored in the given file.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path of the file.

serializer

The ESerializer file-like object that is read from.

Remarks

When loading, an image is resized if need be. On the opposite, an ROI cannot be resized, and the sizes *must* match.
The image contents around the ROI remains unchanged.
If a serializer is used, then the Euresys proprietary file format is expected. This format preserves attributes and sub-ROIs.
See Supported Image File Types for details about supported files.
See Image File Access - Save, Load - for details about file format and compatibility. When loading a color image file into a gray-level image, the conversion equation will depend of the current ERgbStandard set.

EBaseROI::GetOrgX

EBaseROI::SetOrgX

Gets or sets the x-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.

Namespace: Euresys::Open_eVision

[C++]

```
int GetOrgX() const
void SetOrgX(int x)
```

Remarks

The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI::GetOrgY

EBaseROI::SetOrgY

Gets or sets the y-coordinate of the upper left corner of the ROI, relative to its parent ROI/image.

Namespace: Euresys::Open_eVision

[C++]

```
int GetOrgY() const
void SetOrgY(int y)
```

Remarks

The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EBaseROI* GetParent()
```

EBaseROI::GetPlanesPerPixel

Gets the number of color components in each pixel of the ROI/image.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetPlanesPerPixel() const
```

EBaseROI::GetRowPitch

Gets the pitch of a row (the number of bytes between two vertically adjacent pixels). This is also valid for BW1 images (one bit per pixel).

Namespace: Euresys::Open_eVision

[C++]

```
int GetRowPitch() const
```

EBaseROI::Save

Saves the EBaseROI object to the given file.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path of the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

serializer

The ESerializer file-like object that is written to.

Remarks

By default (if no format is specified), the file format is determined from the file extension.
If a serializer is used, then the Euresys proprietary file format is used. This format preserves attributes and sub-ROIs.
See Supported Image File Types for details about supported files.
See Image File Access - Save, Load - for details about file format and compatibility.

EBaseROI::SaveJpeg

Saves the EBaseROI object to the given file, in JPEG format.

Namespace: Euresys::Open_eVision

[C++]

```
void SaveJpeg(
 const std::string& path,
 int quality
 )
```

Parameters

path

The full path of the destination file.

quality

JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

Remarks

See Image File Access - Save, Load - for details about file format and quality.

EBaseROI::SaveJpeg2K

Saves the EBaseROI object to the given file, in JPEG 2000 format.

Namespace: Euresys::Open_eVision

[C++]

```
void SaveJpeg2K(
 const std::string& path,
 int quality
 )
```

Parameters

path

The full path of the destination file.

quality

JPEG 2000 quality, between 1 and 512. The default value is 16.

Remarks

See Image File Access - Save, Load - for details about file format and quality.

EBaseROI::SavePng

Saves the EBaseROI object to the given file, in PNG format.

Namespace: Euresys::Open_eVision

[C++]

```
void SavePng(
 const std::string& path,
 int compression
 )
```

Parameters

path

The full path of the destination file.

compression

PNG compression, between 1 and 9 (1 is the fastest and 9 is the best compression). The default value is 1.

Remarks

See Image File Access - Save, Load - for details about file format and quality.

EBaseROI::SetImagePtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision

[C++]

```
void SetImagePtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EBaseROI::SetImagePtr.

Remarks

This call is only valid on an image. An ROI gets its buffer from its parent while an image normally allocates a pixel buffer automatically. The pointer to this buffer refers to the top left pixel of the image. The next pixels are stored contiguously, row by row, from top to bottom and from left to right.
Padding at the end of a row may be used, but it must lead to rows that are multiple of 4 bytes. This method overrides the internally allocated image buffer of the EBaseROI.
As long as the image accesses this buffer, it must not be deleted.

EBaseROI::SetPlacement

Sets the placement of an ROI, relative to its parent ROI/image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPlacement(
 int originX,
 int originY,
 int width,
 int height
 )
```

Parameters

originX

New x-coordinate of the top-left pixel of this ROI.

originY

New y-coordinate of the top-left pixel of this ROI.

width

New ROI width.

height

New ROI height.

Remarks

This method can only be called on ROIs.
The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

EBaseROI::SetSize

Sets the width and height of an ROI/image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSize(
 int width,
 int height
 )

void SetSize(
 const EBaseROI* other
 )
```

Parameters

width

The new requested ROI/image width.

height

The new requested ROI/image height.

other

The other ROI/image whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an image* is specified as a number of columns (width) and rows (height).
The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries.
The *placement of an ROI* is given by the x and y coordinates of its upper left pixel relative to its parent image, and also by its width and its height.

EBaseROI::GetTitle

EBaseROI::SetTitle

Gets or sets the Title attribute of the ROI/image. This is a convenience property that allows storing some information in the ROI/image. Please note that not all file formats preserve this information.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetTitle() const
void SetTitle(const std::string& title)
```

EBaseROI::GetTotalHeight

Gets the height, in pixels, of the ROI topmost parent.

Namespace: Euresys::Open_eVision

[C++]

```
int GetTotalHeight() const
```

Remarks

The *total size* of an ROI is the size of its *topmost* parent.

EBaseROI::GetTotalOrgX

Gets the x-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.

Namespace: Euresys::Open_eVision

[C++]

```
int GetTotalOrgX() const
```

Remarks

The *total origin coordinates* of an ROI indicate the position of its upper left pixel relative to the *topmost* parent image.
The total origin coordinates (top-left pixel) of a topmost parent are always (0,0).

EBaseROI::GetTotalOrgY

Gets the y-coordinate of the upper left pixel of the ROI, relative to its topmost parent upper left pixel.

Namespace: Euresys::Open_eVision

[C++]

```
int GetTotalOrgY() const
```

Remarks

The *total origin coordinates* of an ROI indicate the position of its upper left pixel relative to the *topmost* parent.
The total origin coordinates (top-left pixel) of a topmost parent are always (0,0).

EBaseROI::GetTotalWidth

Gets the width, in pixels, of the ROI topmost parent.

Namespace: Euresys::Open_eVision

[C++]

```
int GetTotalWidth() const
```

Remarks

The *total size* of an ROI is the size of its *topmost* parent.

EBaseROI::GetType

Gets the ROI/image pixel type, as defined by the EImageType enumeration.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EImageType GetType() const
```

EBaseROI::GetWidth

EBaseROI::SetWidth

Gets or sets the width of the ROI.

Namespace: Euresys::Open_eVision

[C++]

```
int GetWidth() const
void SetWidth(int w)
```

Remarks

The *placement of an ROI* is given by the coordinates its upper left corner (origin point), relative to its parent ROI/image and by its size (width and height).

4.30. EBinaryImageSegmenter Class

Segments a binary image.

Remarks

This segmenter is applicable to EROIBW1 grayscale images.
It produces coded images with two layers: The Black layer (usually, with index 0) contains the unmasked pixels having a binary value equal to zero; and the White layer (usually, with index 1) contains the remaining unmasked pixels, i.e. unmasked pixels having a binary value equal to one.

Base Class:ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

EBinaryImageSegmenter	-
operator==	Comparison operator.

EBinaryImageSegmenter::EBinaryImageSegmenter

-

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void EBinaryImageSegmenter(
 const ETwoLayersImageSegmenter& other
 )
```

Parameters

other

-

EBinaryImageSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool operator==(
 const EBinaryImageSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to.

4.31. EBW16PathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EBW16PathVector member, and then add elements one at time at the tail by calling the EBW16PathVector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EBW16PathVector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.

Draw	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).
DrawClosedContour	Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes EContourMode_ClockwiseAlwaysClosed and EContourMode_AnticlockwiseAlwaysClosed.
DrawWithCurrentPen	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).
EBW16PathVector	Constructs a vector.
GetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW16PathVector object into the current EBW16PathVector object
SetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
SetElement	Modifies the vector element at the given index by the given value.

EBW16PathVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

[C++]

```
void AddElement(
 EBW16Path element
 )
```

Parameters

element

The element to be added.

EBW16PathVector::GetClosed

EBW16PathVector::SetClosed

Flag indicating whether the shape built with EasyImage::Contour must be closed or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetClosed() const
void SetClosed(bool bClosed)
```

EBW16PathVector::Draw

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPathVectorDrawOption option,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

option

-

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW16PathVector::DrawClosedContour

Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes ClockwiseAlwaysClosed and AnticlockwiseAlwaysClosed.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawClosedContour(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EContourMode contourMode,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

contourMode

Contour mode used to get this path vector used to draw the external boundary of the contour.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

EBW16PathVector::DrawWithCurrentPen

This method is deprecated.

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW16PathVector::EBW16PathVector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW16PathVector(
 )

void EBW16PathVector(
 OEV_UINT32 maxNumberOfElements
 )

void EBW16PathVector(
 const EBW16PathVector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW16PathVector object to be copied

EBW16PathVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW16Path GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EBW16PathVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW16PathVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW16Path& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EBW16PathVector (excluded) of the element to be accessed.

EBW16PathVector::operator=

Copies all the data from another EBW16PathVector object into the current EBW16PathVector object

Namespace: Euresys::Open_eVision

[C++]

```
EBW16PathVector& operator=(
 const EBW16PathVector& other
 )
```

Parameters

other

EBW16PathVector object to be copied

EBW16PathVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

```
[C++]
void* GetRawDataPtr() const
```

EBW16PathVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EBW16Path value
 )
```

Parameters

index

Index, between 0 and EBW16PathVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.32. EBW16PixelAccessor Class

Manages a BW16 pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EBW16PixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EBW16PixelAccessor::EBW16PixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EBW16PixelAccessor(
 EROIBW16& roi
 )
```

Parameters

roi

Pixel source.

EBW16PixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EBW16PixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 OEV_UINT16 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

Pixel value.

x

Pixel X coordinate.

y

Pixel Y coordinate.

4.33. EBW16Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EBW16Vector member, and then add elements one at time at the tail by calling the EBW16Vector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EBW16Vector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws a plot of the vector element values.
DrawWithCurrentPen	Draws a plot of the vector element values.
EBW16Vector	Constructs a vector.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW16Vector object into the current EBW16Vector object
SetElement	Modifies the vector element at the given index by the given value.
WeightedMoment	Returns the first order geometric moment (weighted gravity center).

EBW16Vector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EBW16 element
 )
```

Parameters

element

The element to be added.

EBW16Vector::Draw

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

width
Outermost horizontal size, in pixels.

height
Outermost vertical size, in pixels.

originX
Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY
Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color
The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW16Vector::DrawWithCurrentPen

This method is deprecated.

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

width
Outermost horizontal size, in pixels.

height
Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW16Vector::EBW16Vector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW16Vector(
 )

void EBW16Vector(
 OEV_UINT32 maxNumberOfElements
 )

void EBW16Vector(
 const EBW16Vector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW16Vector object to be copied

EBW16Vector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW16 GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EBW16Vector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW16Vector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW16& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EBW16Vector (excluded) of the element to be accessed.

EBW16Vector::operator=

Copies all the data from another EBW16Vector object into the current EBW16Vector object

Namespace: Euresys::Open_eVision

[C++]

```
EBW16Vector& operator=(
 const EBW16Vector& other
 )
```

Parameters

other

EBW16Vector object to be copied

EBW16Vector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EBW16Vector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EBW16 value
 )
```

Parameters

index

Index, between 0 and EBW16Vector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW16Vector::WeightedMoment

Returns the first order geometric moment (weighted gravity center).

Namespace: Euresys::Open_eVision

```
[C++]
float WeightedMoment(
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

from

First element of the vector portion for which the weighted moment will be calculated. By default, this is the first element of the vector.

to

Last element of the vector portion for which the weighted moment will be calculated. By default, this is the last element of the vector.

4.34. EBW32PixelAccessor Class

Manages a BW32 pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EBW32PixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EBW32PixelAccessor::EBW32PixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EBW32PixelAccessor(
 EROIBW32& roi
 )
```

Parameters

roi

Pixel source.

EBW32PixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EBW32PixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixel(
 OEV_UINT32 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

Pixel value.

x

Pixel X coordinate.

y

Pixel Y coordinate.

4.35. EBW32Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EBW32Vector member, and then add elements one at time at the tail by calling the EBW32Vector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EBW32Vector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws a plot of the vector element values.
DrawWithCurrentPen	Draws a plot of the vector element values.
EBW32Vector	Constructs a vector.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW32Vector object into the current EBW32Vector object
SetElement	Modifies the vector element at the given index by the given value.
WeightedMoment	Returns the first order geometric moment (weighted gravity center).

EBW32Vector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EBW32 element
 )
```

Parameters

element

The element to be added.

EBW32Vector::Draw

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

width
Outermost horizontal size, in pixels.

height
Outermost vertical size, in pixels.

originX
Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY
Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color
The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW32Vector::DrawWithCurrentPen

This method is deprecated.

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

width
Outermost horizontal size, in pixels.

height
Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW32Vector::EBW32Vector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW32Vector(
 )
void EBW32Vector(
 OEV_UINT32 maxNumberOfElements
 )
void EBW32Vector(
 const EBW32Vector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW32Vector object to be copied

EBW32Vector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW32 GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EBW32Vector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW32Vector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW32& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EBW32Vector (excluded) of the element to be accessed.

EBW32Vector::operator=

Copies all the data from another EBW32Vector object into the current EBW32Vector object

Namespace: Euresys::Open_eVision

[C++]

```
EBW32Vector& operator=(
 const EBW32Vector& other
 )
```

Parameters

other

EBW32Vector object to be copied

EBW32Vector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EBW32Vector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EBW32 value
 )
```

Parameters

index

Index, between 0 and EBW32Vector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW32Vector::WeightedMoment

Returns the first order geometric moment (weighted gravity center).

Namespace: Euresys::Open_eVision

```
[C++]
float WeightedMoment(
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

from

First element of the vector portion for which the weighted moment will be calculated. By default, this is the first element of the vector.

to

Last element of the vector portion for which the weighted moment will be calculated. By default, this is the last element of the vector.

4.36. EBW8PathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EBW8PathVector member, and then add elements one at time at the tail by calling the EBW8PathVector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EBW8PathVector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).
DrawClosedContour	Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes EContourMode_ClockwiseAlwaysClosed and EContourMode_AnticlockwiseAlwaysClosed.
DrawWithCurrentPen	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).
EBW8PathVector	Constructs a vector.
GetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW8PathVector object into the current EBW8PathVector object
SetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
SetElement	Modifies the vector element at the given index by the given value.

EBW8PathVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EBW8Path element
 )
```

Parameters

element

The element to be added.

EBW8PathVector::GetClosed

EBW8PathVector::SetClosed

Flag indicating whether the shape built with EasyImage::Contour must be closed or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetClosed() const

void SetClosed(bool bClosed)
```

EBW8PathVector::Draw

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPathVectorDrawOption option,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

option

-

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW8PathVector::DrawClosedContour

Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes ClockwiseAlwaysClosed and AnticlockwiseAlwaysClosed.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawClosedContour(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EContourMode contourMode,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

contourMode

Contour mode used to get this path vector used to draw the external boundary of the contour.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

EBW8PathVector::DrawWithCurrentPen

This method is deprecated.

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW8PathVector::EBW8PathVector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW8PathVector(
 )

void EBW8PathVector(
 OEV_UINT32 maxNumberOfElements
 )

void EBW8PathVector(
 const EBW8PathVector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW8PathVector object to be copied

EBW8PathVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
EBW8Path GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EBW8PathVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW8PathVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
EBW8Path& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EBW8PathVector (excluded) of the element to be accessed.

EBW8PathVector::operator=

Copies all the data from another EBW8PathVector object into the current EBW8PathVector object

Namespace: Euresys::Open_eVision

[C++]

```
EBW8PathVector& operator=(
 const EBW8PathVector& other
 )
```

Parameters

other

EBW8PathVector object to be copied

EBW8PathVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EBW8PathVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

[C++]

```
void SetElement(
 int index,
 EBW8Path value
 )
```

Parameters

index

Index, between 0 and EBW8PathVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.37. EBW8PixelAccessor Class

Manages a BW8 pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EBW8PixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EBW8PixelAccessor::EBW8PixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EBW8PixelAccessor(
 EROIBW8& roi
 )
```

Parameters

roi

Pixel source.

EBW8PixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT8 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EBW8PixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 OEV_UINT8 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

Pixel value.

x

Pixel X coordinate.

y

Pixel Y coordinate.

4.38. EBW8Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EBW8Vector member, and then add elements one at time at the tail by calling the EBW8Vector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EBW8Vector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws a plot of the vector element values.
DrawWithCurrentPen	Draws a plot of the vector element values.
EBW8Vector	Constructs a vector.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBW8Vector object into the current EBW8Vector object
SetElement	Modifies the vector element at the given index by the given value.
WeightedMoment	Returns the first order geometric moment (weighted gravity center).

EBW8Vector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EBW8 element
 )
```

Parameters

element

The element to be added.

EBW8Vector::Draw

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

width
Outermost horizontal size, in pixels.

height
Outermost vertical size, in pixels.

originX
Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY
Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color
The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW8Vector::DrawWithCurrentPen

This method is deprecated.

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

width
Outermost horizontal size, in pixels.

height
Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBW8Vector::EBW8Vector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW8Vector(
 )

void EBW8Vector(
 OEV_UINT32 maxNumberOfElements
 )

void EBW8Vector(
 const EBW8Vector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EBW8Vector object to be copied

EBW8Vector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW8 GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EBW8Vector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW8Vector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EBW8& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EBW8Vector (excluded) of the element to be accessed.

EBW8Vector::operator=

Copies all the data from another EBW8Vector object into the current EBW8Vector object

Namespace: Euresys::Open_eVision

[C++]

```
EBW8Vector& operator=(
 const EBW8Vector& other
 )
```

Parameters

other

EBW8Vector object to be copied

EBW8Vector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EBW8Vector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EBW8 value
 )
```

Parameters

index

Index, between 0 and EBW8Vector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBW8Vector::WeightedMoment

Returns the first order geometric moment (weighted gravity center).

Namespace: Euresys::Open_eVision

```
[C++]
float WeightedMoment(
 OEV_UINT32 from,
 OEV_UINT32 to
 )
```

Parameters

from

First element of the vector portion for which the weighted moment will be calculated. By default, this is the first element of the vector.

to

Last element of the vector portion for which the weighted moment will be calculated. By default, this is the last element of the vector.

4.39. EBWHistogramVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EBWHistogramVector member, and then add elements one at time at the tail by calling the EBWHistogramVector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EBWHistogramVector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws a plot of the vector element values.
DrawWithCurrentPen	Draws a plot of the vector element values.
EBWHistogramVector	Constructs a vector.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EBWHistogramVector object into the current EBWHistogramVector object
SetElement	Modifies the vector element at the given index by the given value.

EBWHistogramVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 OEV_UINT32 element
 )
```

Parameters

element

The element to be added.

EBWHistogramVector::Draw

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )
void Draw(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color

The color in which to draw the overlay.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBWHistogramVector::DrawWithCurrentPen

This method is deprecated.

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EBWHistogramVector::EBWHistogramVector

Constructs a vector.

Namespace: Euresys::Open_eVision

```
[C++]
void EBWHistogramVector(
 )
```

```
void EBWHistogramVector(
 const EBWHistogramVector& other
 )
void EBWHistogramVector(
 OEV_UINT32 maxNumberOfElements
 )
```

Parameters

other

EBWHistogramVector object to be copied

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

EBWHistogramVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EBWHistogramVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EBWHistogramVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EBWHistogramVector (excluded) of the element to be accessed.

EBWHistogramVector::operator=

Copies all the data from another EBWHistogramVector object into the current EBWHistogramVector object

Namespace: Euresys::Open_eVision

```
[C++]
EBWHistogramVector& operator=(
 const EBWHistogramVector& other
 )
```

Parameters

other

EBWHistogramVector object to be copied

EBWHistogramVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

```
[C++]
void* GetRawDataPtr() const
```

EBWHistogramVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 OEV_UINT32 value
 )
```

Parameters

index

Index, between 0 and EBWHistogramVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.40. EC15PixelAccessor Class

Manages a C15 pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EC15PixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EC15PixelAccessor::EC15PixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EC15PixelAccessor(
 EROIC15& roi
 )
```

Parameters

roi

Pixel source.

EC15PixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
EC15 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EC15PixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC15 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value
Pixel value.

x
Pixel X coordinate.

y
Pixel Y coordinate.

4.41. EC16PixelAccessor Class

Manages a C16 pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EC16PixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EC16PixelAccessor::EC16PixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EC16PixelAccessor(
 EROIC16& roi
 )
```

Parameters

roi

Pixel source.

EC16PixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
EC16 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EC16PixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC16 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

Pixel value.

x

Pixel X coordinate.

y

Pixel Y coordinate.

4.42. EC24APixelAccessor Class

Manages a C24A pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EC24APixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EC24APixelAccessor::EC24APixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EC24APixelAccessor(
 EROIC24A& roi
 )
```

Parameters

roi

Pixel source.

EC24APixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
EC24A GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EC24APixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC24A value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value

Pixel value.

x

Pixel X coordinate.

y

Pixel Y coordinate.

4.43. EC24PathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EC24PathVector member, and then add elements one at time at the tail by calling the EC24PathVector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EC24PathVector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

DrawClosedContour	Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes EContourMode_ClockwiseAlwaysClosed and EContourMode_AnticlockwiseAlwaysClosed.
DrawWithCurrentPen	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).
EC24PathVector	Constructs a vector.
GetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EC24PathVector object into the current EC24PathVector object
SetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
SetElement	Modifies the vector element at the given index by the given value.

EC24PathVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

[C++]

```
void AddElement(
 EC24Path element
 )
```

Parameters

element

The element to be added.

EC24PathVector::GetClosed

EC24PathVector::SetClosed

Flag indicating whether the shape built with EasyImage::Contour must be closed or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetClosed() const

void SetClosed(bool bClosed)
```

EC24PathVector::Draw

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPathVectorDrawOption option,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

option

-

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EC24PathVector::DrawClosedContour

Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes ClockwiseAlwaysClosed and AnticlockwiseAlwaysClosed.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawClosedContour(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EContourMode contourMode,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

contourMode

Contour mode used to get this path vector used to draw the external boundary of the contour.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

EC24PathVector::DrawWithCurrentPen

This method is deprecated.

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EC24PathVector::EC24PathVector

Constructs a vector.

Namespace: Euresys::Open_eVision

```
[C++]
void EC24PathVector(
 )
void EC24PathVector(
 const EC24PathVector& other
 )
void EC24PathVector(
 OEV_UINT32 maxNumberOfElements
 )
```

Parameters

other

EC24PathVector object to be copied

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

EC24PathVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
EC24Path GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EC24PathVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EC24PathVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EC24Path& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EC24PathVector (excluded) of the element to be accessed.

EC24PathVector::operator=

Copies all the data from another EC24PathVector object into the current EC24PathVector object

Namespace: Euresys::Open_eVision

[C++]

```
EC24PathVector& operator=(
 const EC24PathVector& other
 )
```

Parameters

other

EC24PathVector object to be copied

EC24PathVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EC24PathVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

[C++]

```
void SetElement(
 int index,
 EC24Path value
 )
```

Parameters

index

Index, between 0 and EC24PathVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.44. EC24PixelAccessor Class

Manages a C24 pixel accessor context.

Namespace: Euresys::Open_eVision

Methods

EC24PixelAccessor	Constructor.
GetPixel	Gets the Pixel at the given coordinates.
SetPixel	Sets the Pixel at the given coordinates.

EC24PixelAccessor::EC24PixelAccessor

Constructor.

Namespace: Euresys::Open_eVision

```
[C++]
void EC24PixelAccessor(
 EROIC24& roi
 )
```

Parameters

roi

Pixel source.

EC24PixelAccessor::GetPixel

Gets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
EC24 GetPixel(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

Pixel X coordinate.

y

Pixel Y coordinate.

EC24PixelAccessor::SetPixel

Sets the Pixel at the given coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC24 value,
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

value
Pixel value.

x
Pixel X coordinate.

y
Pixel Y coordinate.

4.45. EC24Vector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EC24Vector member, and then add elements one at time at the tail by calling the EC24Vector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EC24Vector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws a plot of the vector element values.
EC24Vector	Constructs a vector.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EC24Vector object into the current EC24Vector object
SetElement	Modifies the vector element at the given index by the given value.

EC24Vector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EC24 element
 )
```

Parameters

element

The element to be added.

EC24Vector::Draw

Draws a plot of the vector element values.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height
 )

void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 float originX,
 float originY,
 const ERGBColor& color0,
 const ERGBColor& color1,
 const ERGBColor& color2
 )

void Draw(
 EDrawAdapter* graphicContext,
 float width,
 float height,
 const ERGBColor& color0,
 const ERGBColor& color1,
 const ERGBColor& color2
 )
```

```
void Draw(
 HDC graphicContext,
 float width,
 float height
 )

void Draw(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float width,
 float height,
 float originX,
 float originY,
 const ERGBColor& color0,
 const ERGBColor& color1,
 const ERGBColor& color2
 )

void Draw(
 HDC graphicContext,
 float width,
 float height,
 const ERGBColor& color0,
 const ERGBColor& color1,
 const ERGBColor& color2
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

width

Outermost horizontal size, in pixels.

height

Outermost vertical size, in pixels.

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

color0

The color to be used when drawing the curve of the first color component of the vector, as an RGB color. By default, the current pen is used to draw the curve.

color1

The color to be used when drawing the curve of the second color component of the vector, as an RGB color. By default, the current pen is used to draw the curve.

color2

The color to be used when drawing the curve of the third color component of the vector, as an RGB color. By default, the current pen is used to draw the curve.

Remarks

A vector is able to draw itself in a window. The vector plots the element values as a function of the element indices. The drawing appears on a neutral background. Drawing is done in the device context associated to the desired window. To draw the curves, the current pen is used. To draw the annotations, a black pen is used instead. In the special case of the EC24Vector, three curves are drawn instead of one, each corresponding to a color component. Three pen objects must be provided to draw the curves with appropriate attributes. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EC24Vector::EC24Vector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EC24Vector(
 )

void EC24Vector(
 OEV_UINT32 maxNumberOfElements
 )

void EC24Vector(
 const EC24Vector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EC24Vector object to be copied

EC24Vector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EC24 GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EC24Vector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EC24Vector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EC24& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EC24Vector (excluded) of the element to be accessed.

EC24Vector::operator=

Copies all the data from another EC24Vector object into the current EC24Vector object

Namespace: Euresys::Open_eVision

[C++]

```
EC24Vector& operator=(
 const EC24Vector& other
 )
```

Parameters

other

EC24Vector object to be copied

EC24Vector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

```
[C++]
void* GetRawDataPtr() const
```

EC24Vector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EC24 value
 )
```

Parameters

index

Index, between 0 and EC24Vector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.46. ECalibrationGenerator Class

Represents a 3D calibration model generator, a class made to compute calibration models.

Derived Class(es): EObjectBasedCalibrationGenerator

Namespace: Euresys::Open_eVision::Easy3D

4.47. ECalibrationModel Class

Represents a 3D calibration model.

Derived Class (es): EExplicitGeometricCalibrationModel EObjectBasedCalibrationModel EScaleCalibrationModel

Namespace: Euresys::Open_eVision::Easy3D

Methods

Apply	Applies this model to convert an uncalibrated point to a world position. This method returns a world position.
Create	Factory method from a serializer stream: allocates and reads the calibration model from the given serializer. Returns the corresponding calibration model, must be released by caller.
GetType	Returns the type of calibration model, see ECalibrationType.
Save	Saves the calibration model. The given ESerializer must have been created for writing.

ECalibrationModel::Apply

Applies this model to convert an uncalibrated point to a world position.
This method returns a world position.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint Apply(
 E3DPoint uvwPoint
 )
```

Parameters

uvwPoint

The position of a depth map pixel.

ECalibrationModel::Create

Factory method from a serializer stream: allocates and reads the calibration model from the given serializer.
Returns the corresponding calibration model, must be released by caller.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::unique_ptr<ECalibrationModel> Create(
 const std::string& path
 )

std::unique_ptr<ECalibrationModel> Create(
 ESerializer* file
 )
```

Parameters

path
The file path.

file
A serializer created for reading.

ECalibrationModel::Save

Saves the calibration model. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ECalibrationModel::GetType

Returns the type of calibration model, see ECalibrationType.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::ECalibrationType GetType() const
```

4.48. ECannyEdgeDetector Class

Manages a complete context for the Canny edge detector.

Remarks

The Canny edge detector operates on a grayscale BW8 image and delivers a black-and-white BW8 image where pixels have only 2 possible values: 0 and 255. Pixels corresponding to edges in the source image are set to value 255 in the output image; The other pixels are set to value 0.

Namespace: Euresys::Open_eVision

Methods

Apply	Apply the Canny edge detector on an image/ROI.
ECannyEdgeDetector	Constructs a ECannyEdgeDetector object initialized to its default values.
GetHighThreshold	Sets the high hysteresis threshold for a pixel to be considered as an edge.
GetLowThreshold	Sets the low hysteresis threshold for a pixel to be considered as an edge.
GetSmoothingScale	The scale of the features of interest.
GetThresholdingMode	Sets the mode of the hysteresis thresholding.
ResetSmoothingScale	Prevents the smoothing of the source image by a Gaussian filter.
SetHighThreshold	Sets the high hysteresis threshold for a pixel to be considered as an edge.
SetLowThreshold	Sets the low hysteresis threshold for a pixel to be considered as an edge.
SetSmoothingScale	The scale of the features of interest.
SetThresholdingMode	Sets the mode of the hysteresis thresholding.

ECannyEdgeDetector::Apply

Apply the Canny edge detector on an image/ROI.

Namespace: Euresys::Open_eVision

```
[C++]
void Apply(
 const EROIBW8& source,
 EROIBW8& result
 )
```

Parameters

source

The source image/ROI.

result

The output image/ROI.

Remarks

The output ROI must have the same size than the input ROI.

ECannyEdgeDetector::ECannyEdgeDetector

Constructs a ECannyEdgeDetector object initialized to its default values.

Namespace: Euresys::Open_eVision

```
[C++]
void ECannyEdgeDetector(
 )
```

ECannyEdgeDetector::GetHighThreshold

ECannyEdgeDetector::SetHighThreshold

Sets the high hysteresis threshold for a pixel to be considered as an edge.

Namespace: Euresys::Open_eVision

```
[C++]
float GetHighThreshold() const
void SetHighThreshold(float highThreshold)
```

ECannyEdgeDetector::GetLowThreshold

ECannyEdgeDetector::SetLowThreshold

Sets the low hysteresis threshold for a pixel to be considered as an edge.

Namespace: Euresys::Open_eVision

```
[C++]
float GetLowThreshold() const
void SetLowThreshold(float lowThreshold)
```

ECannyEdgeDetector::ResetSmoothingScale

Prevents the smoothing of the source image by a Gaussian filter.

Namespace: Euresys::Open_eVision

[C++]

```
void ResetSmoothingScale(
 )
```

Remarks

Calling this method is equivalent to set ECannyEdgeDetector::SmoothingScale to zero. It disables the use of the Gaussian filter.

ECannyEdgeDetector::GetSmoothingScale

ECannyEdgeDetector::SetSmoothingScale

The scale of the features of interest.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSmoothingScale() const
void SetSmoothingScale(float scale)
```

Remarks

This scale corresponds to the standard deviation of the Gaussian filter that is used to smooth the source image before the computation of the gradient, hereby selecting the scale of the features of interest.
If this scale is set to zero, no smoothing is achieved: The gradient is computed directly on the raw source image, speeding up the detector, but making the process much less reliable.

ECannyEdgeDetector::GetThresholdingMode

ECannyEdgeDetector::SetThresholdingMode

Sets the mode of the hysteresis thresholding.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ECannyThresholdingMode GetThresholdingMode() const
void SetThresholdingMode(Euresys::Open_eVision::ECannyThresholdingMode mode)
```

Remarks

If the threshold mode is set to ECannyThresholdingMode_Absolute, the threshold values are interpreted as absolute thresholds. In this case, the thresholds must be strictly positive real values.

If the threshold mode is set to ECannyThresholdingMode_Relative, the thresholds are expressed as a fraction ranging from 0 to 1 of the maximum value of the gradient of the source image.

In either case, the low threshold must be less than the high threshold.

4.49. EChecker Class

This class is deprecated.

Manages a complete context for the inspection tool based on image comparison in EasyOCV.

Namespace: Euresys::Open_eVision

Methods

AddPathName	Adds a single file pathname.
Attach	Associates a source image to a checker context.
BatchLearn	Performs the learning sequence using the specified list of image files.
Drag	Moves the relevant ROI by means of its handle.
Draw	Draws one of the geometric items that define the EChecker tool.
DrawWithCurrentPen	Draws one of the geometric items that define the EChecker tool.
EChecker	Constructs an uninitialized checker context.
EmptyPathNames	Clears the list of file pathnames.
GetAverage	Global intensity of the mother image.
GetDarkGray	Gray level in the dark areas of the mother image.
GetDegreesOfFreedom	Boolean combination of EDegreesOfFreedom members, that indicates which degrees of freedom are to be considered.
GetDeviation	Global contrast of the mother image.
GetHigh	High threshold image for the adaptive segmentation.
GetHitHandle	Handle currently hit.
GetHitRoi	ROI currently hit.
GetLightGray	Gray level in the light areas of the mother image.
GetLow	Low threshold image for the adaptive segmentation.
GetNormalize	Current normalization mode.
GetNumAverageSamples	Number of samples that were accumulated in the "average" phase of the training.

GetNumDeviationSamples	Number of samples that were accumulated in the "deviation" phase of the training.
GetPanX	Current horizontal panning value, expressed in pixels, for use in display operations.
GetPanY	Current vertical panning value, expressed in pixels, for use in display operations.
GetRegistered	Represents the source image, after it has been aligned with the reference image.
GetRelativeTolerance	Current tolerance factor to be used for threshold image setup.
GetToleranceX	Current horizontal search tolerance, in pixels.
GetToleranceY	Current vertical search tolerance, in pixels.
GetZoomX	Current horizontal zooming factor for use in display operations.
GetZoomY	Current vertical zooming factor for use in display operations.
HitTest	Returns true if the cursor is over one of the dragging handles.
Learn	Accumulates a reference image, following a sequence of operations.
Load	Loads the EChecker. The given ESerializer must have been created for reading.
Register	Realigns and normalizes the source image.
Save	Saves the EChecker. The given ESerializer must have been created for writing.
SetDarkGray	Gray level in the dark areas of the mother image.
SetDegreesOfFreedom	Boolean combination of EDegreesOfFreedom members, that indicates which degrees of freedom are to be considered.
SetLightGray	Gray level in the light areas of the mother image.
SetNormalize	Current normalization mode.
SetPan	Sets the panning value, expressed in pixels, for use in display operations.
SetRelativeTolerance	Current tolerance factor to be used for threshold image setup.
SetTolerance	Sets the search margin, i.e. the width and height of the search area surrounding the alignment pattern(s).
SetZoom	Sets the zooming factor for use in display operations.

EChecker::AddPathName

This method is deprecated.

Adds a single file pathname.

Namespace: Euresys::Open_eVision

[C++]

```
void AddPathName(
 const std::string& pathName
 )
```

Parameters

pathName

NULL terminated text string containing the file pathname.

EChecker::Attach

This method is deprecated.

Associates a source image to a checker context.

Namespace: Euresys::Open_eVision

[C++]

```
void Attach(
 EROIBW8* source
 )
```

Parameters

source

Pointer to the source image.

Remarks

The source image is used in all consecutive learning/inspection operations.

EChecker::GetAverage

This property is deprecated.

Global intensity of the mother image.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAverage()
```

Remarks

Valid in mode ENormalizationMode_Moments only.

EChecker::BatchLearn

This method is deprecated.

Performs the learning sequence using the specified list of image files.

Namespace: Euresys::Open_eVision

```
[C++]
void BatchLearn(
 Euresys::Open_eVision::ELearningMode mode
 )
```

Parameters

mode

ELearningMode_RmsDeviation or ELearningMode_AbsDeviation, depending on the preferred method of computing the deviations.

EChecker::GetDarkGray

EChecker::SetDarkGray

This property is deprecated.

Gray level in the dark areas of the mother image.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDarkGray()
void SetDarkGray(float f32DarkGray)
```

Remarks

Valid in mode ENormalizationMode_Threshold only.

EChecker::GetDegreesOfFreedom

EChecker::SetDegreesOfFreedom

This property is deprecated.

Boolean combination of EDegreesOfFreedom members, that indicates which degrees of freedom are to be considered.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetDegreesOfFreedom()
void SetDegreesOfFreedom(OEV_UINT32 un32DegreesOfFreedom)
```

EChecker::GetDeviation

This property is deprecated.

Global contrast of the mother image.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDeviation()
```

Remarks

Valid in mode ENormalizationMode_Moments only.

EChecker::Drag

This method is deprecated.

Moves the relevant ROI by means of its handle.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int x,
 int y
 )
```

Parameters

x

New horizontal cursor position.

y

New vertical cursor position.

EChecker::Draw

This method is deprecated.

Draws one of the geometric items that define the EChecker tool.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Device context of the drawing window.

drawingMode

ROI to be drawn, as defined by EDrawingMode.

handles

true if the dragging handles must be displayed.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EChecker::DrawWithCurrentPen

This method is deprecated.

Draws one of the geometric items that define the EChecker tool.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Device context of the drawing window.

drawingMode

ROI to be drawn, as defined by EDrawingMode.

handles

true if the dragging handles must be displayed.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EChecker::EChecker

This method is deprecated.

Constructs an uninitialized checker context.

Namespace: Euresys::Open_eVision

```
[C++]
void EChecker(
 )
```

EChecker::EmptyPathNames

This method is deprecated.

Clears the list of file pathnames.

Namespace: Euresys::Open_eVision

```
[C++]
void EmptyPathNames(
 )
```

EChecker::GetHigh

This property is deprecated.

High threshold image for the adaptive segmentation.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW8* GetHigh()
```

EChecker::GetHitHandle

This property is deprecated.

Handle currently hit.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EDragHandle GetHitHandle()
```

EChecker::GetHitRoi

This property is deprecated.

ROI currently hit.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ERoiHit GetHitRoi()
```

EChecker::HitTest

This method is deprecated.

Returns true if the cursor is over one of the dragging handles.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 int x,
 int y
 )
```

Parameters

x

Current horizontal cursor position.

y

Current vertical cursor position.

Remarks

In this case, EChecker::HitRoi returns the name of the ROI that has been hit, and EChecker::HitHandle returns the name of the corresponding handle.

EChecker::Learn

This method is deprecated.

Accumulates a reference image, following a sequence of operations.

Namespace: Euresys::Open_eVision

[C++]

```
void Learn(
 Euresys::Open_eVision::ELearningMode mode
 )
```

Parameters

mode

Current mode of operation in the learning sequence, as defined by ELearningMode.

Remarks

First the model is reset; then the matching patterns are shown; next a series of images is presented to estimate the average gray levels; then a second series of images is presented to estimate the gray-level variations; finally, the threshold images are generated. A typical sequence with three reference images goes as follows: For standard deviation estimationEChecker.Learn(ELearningMode_Reset); initializes. EChecker.Register(); realigns and normalizes 1st source image. EChecker.Learn(ELearningMode_RmsDeviation); processes 1st image for deviation info. EChecker.Register(); realigns and normalizes 2nd source image. EChecker.Learn(ELearningMode_RmsDeviation); processes 2nd image for deviation info. EChecker.Register(); realigns and normalizes 3rd source image. EChecker.Learn (ELearningMode_RmsDeviation); processes 3rd image for deviation info. For robust deviation estimationEChecker.Learn(ELearningMode_Reset); initializes. EChecker.Register(); realigns and normalizes 1st source image. EChecker.Learn(ELearningMode_Average); processes 1st image for average info. EChecker.Register(); realigns and normalizes 2nd source image. EChecker.Learn(ELearningMode_Average); processes 2nd image for average info. EChecker.Register(); realigns and normalizes 3rd source image. EChecker.Learn (ELearningMode_Average); processes 3rd image for average info. EChecker.Register(); realigns and normalizes 1st source image. EChecker.Learn(ELearningMode_RmsDeviation); processes 1st image for deviation info. EChecker.Register(); realigns and normalizes 2nd source image. EChecker.Learn(ELearningMode_RmsDeviation); processes 2nd image for deviation info. EChecker.Register(); realigns and normalizes 3rd source image. EChecker.Learn (ELearningMode_RmsDeviation); processes 3rd image for deviation info. EChecker.Learn (ELearningMode_Ready); computes the threshold images.

EChecker::GetLightGray

EChecker::SetLightGray

This property is deprecated.

Gray level in the light areas of the mother image.

Namespace: Euresys::Open_eVision

[C++]

```
float GetLightGray()
void SetLightGray(float f32LightGray)
```

Remarks

Valid in mode ENormalizationMode_Threshold only.

EChecker::Load

This method is deprecated.

Loads the EChecker. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 ESerializer* serializer
 )
void Load(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

The file path.

EChecker::GetLow

This property is deprecated.

Low threshold image for the adaptive segmentation.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW8* GetLow()
```

EChecker::GetNormalize

EChecker::SetNormalize

This property is deprecated.

Current normalization mode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ENormalizationMode GetNormalize()
void SetNormalize(Euresys::Open_eVision::ENormalizationMode eNormalize)
```

EChecker::GetNumAverageSamples

This property is deprecated.

Number of samples that were accumulated in the "average" phase of the training.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumAverageSamples()
```

EChecker::GetNumDeviationSamples

This property is deprecated.

Number of samples that were accumulated in the "deviation" phase of the training.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumDeviationSamples()
```

EChecker::GetPanX

This property is deprecated.

Current horizontal panning value, expressed in pixels, for use in display operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPanX()
```

EChecker::GetPanY

This property is deprecated.

Current vertical panning value, expressed in pixels, for use in display operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPanY()
```

EChecker::Register

This method is deprecated.

Realigns and normalizes the source image.

Namespace: Euresys::Open_eVision

[C++]

```
void Register(
 )
```

Remarks

Only the inspected ROI is processed. The first time this function is called, the current pattern ROI are used to define the search patterns. After registration, public member EChecker::Registered contains the realigned, normalized contents of the inspected ROI.

EChecker::GetRegistered

This property is deprecated.

Represents the source image, after it has been aligned with the reference image.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW8* GetRegistered()
```

EChecker::GetRelativeTolerance

EChecker::SetRelativeTolerance

This property is deprecated.

Current tolerance factor to be used for threshold image setup.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeTolerance()
void SetRelativeTolerance(float f32RelativeTolerance)
```

EChecker::Save

This method is deprecated.

Saves the EChecker. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 ESerializer* serializer
 )
void Save(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

The file path.

EChecker::SetPan

This method is deprecated.

Sets the panning value, expressed in pixels, for use in display operations.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPan(
 float panX,
 float panY
 )
```

Parameters

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

EChecker::SetTolerance

This method is deprecated.

Sets the search margin, i.e. the width and height of the search area surrounding the alignment pattern(s).

Namespace: Euresys::Open_eVision

[C++]

```
void SetTolerance(
 OEV_UINT32 toleranceX,
 OEV_UINT32 toleranceY
 )
```

Parameters

toleranceX

Horizontal search tolerance, in pixels.

toleranceY

Vertical search tolerance, in pixels.

EChecker::SetZoom

This method is deprecated.

Sets the zooming factor for use in display operations.

Namespace: Euresys::Open_eVision

[C++]

```
void SetZoom(
 float zoom
 )

void SetZoom(
 float zoomX,
 float zoomY
 )
```

Parameters

zoom

Magnification factor for zooming in or out in the horizontal and vertical directions (isotropic scaling).

zoomX

Magnification factor for zooming in or out in the horizontal direction.

zoomY

Magnification factor for zooming in or out in the vertical direction.

EChecker::GetToleranceX

This property is deprecated.

Current horizontal search tolerance, in pixels.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetToleranceX()
```

EChecker::GetToleranceY

This property is deprecated.

Current vertical search tolerance, in pixels.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetToleranceY()
```

EChecker::GetZoomX

This property is deprecated.

Current horizontal zooming factor for use in display operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetZoomX()
```

EChecker::GetZoomY

This property is deprecated.

Current vertical zooming factor for use in display operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetZoomY()
```

4.50. EChecker2 Class

Manages a complete context for the EChecker2 golden template inspection tool.

Namespace: Euresys::Open_eVision

Methods

AddTrainingImageFile	Add Training Image File.

ClearTrainingImageFiles	Clear Training Image Files.
DrawInspectionField	Draws the inspection field overlay.
DrawReferenceInspectionField	Draws the reference inspection field overlay.
DrawReferenceSearchFields	Draws the reference search fields overlay.
EChecker2	Constructs an uninitialized golden template inspection context.
GetFiducialHorizontalTolerance	Horizontal positionning tolerance for fiducials. The fiducials in the training and inspection images will be searched around the position of the fiducial given during the initialization process, with a tolerance in pixels set by this parameter. The default value is 30 pixels.
GetFiducialMatchingMode	Fiducial matching mode. This setting allow you to select which type of finder is used to locate the fiducials in the training and inspection images. It can either be geometric (as per EasyFind) or area-based (as per EasyMatch). Default: Geometric finder.
GetFiducialVerticalTolerance	Vertical positionning tolerance for fiducials. The fiducials in the training and inspection images will be searched around the position of the fiducial given during the initialization process, with a tolerance in pixels set by this parameter. The default value is 30 pixels.
GetHighThresholdImage	High threshold image.
GetInspectionTolerance	Inspection Tolerance factor. If you need to be more tolerant towards noise, texture or illumination irregularities, raise this value. If you have clean images, low texture and some defects are missed, lower this value. Default value: 4.0
GetIsInitialized	Application state. Has the checker been initialized.
GetIsTrained	Application state. Has the checker been trained.
GetLastRegisteredImage	Represents the last source image, after it has been aligned with the reference image and normalized.
GetLowThresholdImage	Low threshold image.
GetNormalizationMode	Normalization mode. This setting allows you to specify if you want to enable normalization, and if yes, which type. Normalization allows the training and inspection processes to automatically correct global illumination differences between the images. Default: Moments based normalization.
GetTrainingMode	Training mode. The training mode determines which process will be used to build the threshold images used for inspection. EChecker2 has two training modes: A Quick mode, efficent for crude defects and stable images, and a Precise mode, designed for more subtle defects and handling more variability in the images. Default: Precise training mode.
Initialize	Initialize the training process.
Inspect	Inspect.

Load	Load an inspection model
ResetTraining	Reset training. Use this if you want to reset the state of the object without calling EChecker2::Initialize again.
Save	Save the inspection model
SetFiducialHorizontalTolerance	Horizontal positionning tolerance for fiducials. The fiducials in the training and inspection images will be searched around the position of the fiducial given during the initialization process, with a tolerance in pixels set by this parameter. The default value is 30 pixels.
SetFiducialMatchingMode	Fiducial matching mode. This setting allow you to select which type of finder is used to locate the fiducials in the training and inspection images. It can either be geometric (as per EasyFind) or area-based (as per EasyMatch). Default: Geometric finder.
SetFiducialVerticalTolerance	Vertical positionning tolerance for fiducials. The fiducials in the training and inspection images will be searched around the position of the fiducial given during the initialization process, with a tolerance in pixels set by this parameter. The default value is 30 pixels.
SetInspectionTolerance	Inspection Tolerance factor. If you need to be more tolerant towards noise, texture or illumination irregularities, raise this value. If you have clean images, low texture and some defects are missed, lower this value. Default value: 4.0
SetNormalizationMode	Normalization mode. This setting allows you to specify if you want to enable normalization, and if yes, which type. Normalization allows the training and inspection processes to automatically correct global illumination differences between the images. Default: Moments based normalization.
SetTrainingMode	Training mode. The training mode determines which process will be used to build the threshold images used for inspection. EChecker2 has two training modes: A Quick mode, efficent for crude defects and stable images, and a Precise mode, designed for more subtle defects and handling more variability in the images. Default: Precise training mode.
Train	Train.
TrainFromImageFiles	Train From Image Files.

EChecker2::AddTrainingImageFile

Add Training Image File.

Namespace: Euresys::Open_eVision

[C++]

```
void AddTrainingImageFile(
 const std::string& path
 )
```

Parameters

path

Path to File.

EChecker2::ClearTrainingImageFiles

Clear Training Image Files.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearTrainingImageFiles(
 )
```

EChecker2::DrawInspectionField

Draws the inspection field overlay.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawInspectionField(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawInspectionField(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawInspectionField(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Device context of the drawing window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

Color in which to draw the overlay

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EChecker2::DrawReferenceInspectionField

Draws the reference inspection field overlay.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawReferenceInspectionField(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawReferenceInspectionField(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawReferenceInspectionField(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Device context of the drawing window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

Color in which to draw the overlay

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EChecker2::DrawReferenceSearchFields

Draws the reference search fields overlay.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawReferenceSearchFields(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawReferenceSearchFields(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawReferenceSearchFields(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Device context of the drawing window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning factor. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

color

Color in which to draw the overlay

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EChecker2::EChecker2

Constructs an uninitialized golden template inspection context.

Namespace: Euresys::Open_eVision

```
[C++]
void EChecker2(
 )
```

EChecker2::GetFiducialHorizontalTolerance

EChecker2::SetFiducialHorizontalTolerance

Horizontal positionning tolerance for fiducials. The fiducials in the training and inspection images will be searched around the position of the fiducial given during the initialization process, with a tolerance in pixels set by this parameter. The default value is 30 pixels.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFiducialHorizontalTolerance() const
void SetFiducialHorizontalTolerance(int tolerance)
```

EChecker2::GetFiducialMatchingMode

EChecker2::SetFiducialMatchingMode

Fiducial matching mode. This setting allow you to select which type of finder is used to locate the fiducials in the training and inspection images. It can either be geometric (as per EasyFind) or area-based (as per EasyMatch). Default: Geometric finder.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EFiducialMatchingMode GetFiducialMatchingMode() const
void SetFiducialMatchingMode(Euresys::Open_eVision::EFiducialMatchingMode mode)
```

EChecker2::GetFiducialVerticalTolerance

EChecker2::SetFiducialVerticalTolerance

Vertical positionning tolerance for fiducials. The fiducials in the training and inspection images will be searched around the position of the fiducial given during the initialization process, with a tolerance in pixels set by this parameter. The default value is 30 pixels.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFiducialVerticalTolerance() const
void SetFiducialVerticalTolerance(int tolerance)
```

EChecker2::GetHighThresholdImage

High threshold image.

Namespace: Euresys::Open_eVision

[C++]

```
const EImageBW8& GetHighThresholdImage() const
```

EChecker2::Initialize

Initialize the training process.

Namespace: Euresys::Open_eVision

[C++]

```
void Initialize(
 const EROIBW8& referenceImage,
 const std::vector<Euresys::Open_eVision::ERegion*>& fiducialRegions,
 const ERegion& inspectionRegion
 )
```

Parameters

referenceImage

ROI or Image used as the reference for the training process.

fiducialRegions

Regions of the fiducials used for the registration process.

inspectionRegion

Region used for the inspection process.

EChecker2::Inspect

Inspect.

Namespace: Euresys::Open_eVision

[C++]

```
void Inspect(
 const EROIBW8& roi,
 ECodedImage2& defects
 )
```

Parameters

roi

ROI or Image to inspect.

defects

List of defects returned by the inspection.

EChecker2::GetInspectionTolerance

EChecker2::SetInspectionTolerance

Inspection Tolerance factor. If you need to be more tolerant towards noise, texture or illumination irregularities, raise this value. If you have clean images, low texture and some defects are missed, lower this value. Default value: 4.0

Namespace: Euresys::Open_eVision

[C++]

```
float GetInspectionTolerance() const
void SetInspectionTolerance(float tolerance)
```

EChecker2::GetIsInitialized

Application state. Has the checker been initialized.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetIsInitialized()
```

EChecker2::GetIsTrained

Application state. Has the checker been trained.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetIsTrained()
```

EChecker2::GetLastRegisteredImage

Represents the last source image, after it has been aligned with the reference image and normalized.

Namespace: Euresys::Open_eVision

[C++]

```
const EImageBW8& GetLastRegisteredImage() const
```

EChecker2::Load

Load an inspection model

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& file
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

file

File path

serializer

Serializer. Must be in read mode

EChecker2::GetLowThresholdImage

Low threshold image.

Namespace: Euresys::Open_eVision

[C++]

```
const EImageBW8& GetLowThresholdImage() const
```

EChecker2::GetNormalizationMode

EChecker2::SetNormalizationMode

Normalization mode. This setting allows you to specify if you want to enable normalization, and if yes, which type. Normalization allows the training and inspection processes to automatically correct global illumination differences between the images. Default: Moments based normalization.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ENormalizationMode GetNormalizationMode() const
void SetNormalizationMode(Euresys::Open_eVision::ENormalizationMode mode)
```

Remarks

If you have consistent illumination between your images, using normalization can prove detrimental, as in that case the defect themselves might be enough to shift the global contrast.

EChecker2::ResetTraining

Reset training. Use this if you want to reset the state of the object without calling EChecker2::Initialize again.

Namespace: Euresys::Open_eVision

[C++]

```
void ResetTraining(
 )
```

EChecker2::Save

Save the inspection model

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 ESerializer* serializer
 )
void Save(
 const std::string& file
 )
```

Parameters

serializer

Serializer. Must be in write mode

file

File path

EChecker2::Train

Train.

Namespace: Euresys::Open_eVision

[C++]

```
void Train(
 const std::vector<const Euresys::Open_eVision::EROIBW8*>& rois
 )
```

Parameters

rois

ROI or Images to train on.

EChecker2::TrainFromImageFiles

Train From Image Files.

Namespace: Euresys::Open_eVision

[C++]

```
void TrainFromImageFiles(
 )
```

EChecker2::GetTrainingMode

EChecker2::SetTrainingMode

Training mode. The training mode determines which process will be used to build the threshold images used for inspection. EChecker2 has two training modes: A Quick mode, efficent for crude defects and stable images, and a Precise mode, designed for more subtle defects and handling more variability in the images. Default: Precise training mode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETrainingMode GetTrainingMode() const
void SetTrainingMode(Euresys::Open_eVision::ETrainingMode mode)
```

4.51. ECircle Class

Represents a model of a circle (or arc) in EasyGauge.

Base Class:EFrame

Namespace: Euresys::Open_eVision

Methods

CopyTo	Copies all the data of the current ECircle object into another ECircle object and returns it.
Distance	Returns the smallest distance between this ECircle object an another ECircle.
ECircle	Constructs a ECircle object.
GetAmplitude	Angular amplitude of the ECircle object.
GetApex	Apex point coordinates of a ECircle object.
GetApexAngle	Angular position at the apex of a ECircle object.
GetArcLength	Circle arc length of a ECircle object.
GetDiameter	Diameter of a ECircle object.
GetDirect	Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.
GetDistanceBetweenLineAndCircle	Computes the distance between a line and a circle.
GetDistanceBetweenPointAndCircle	Computes the distance between a point and a circle.
GetEnd	End point coordinates of a ECircle object.
GetEndAngle	Angular position of the end of a ECircle object.
GetFull	Flag indicating whether the ECircle object is a full circle or not.
GetIntersectionOfCircles	Computes the intersections between two circles. Returns the number of intersections and stores the found intersections in the provided point parameters.
GetIntersectionOfLineAndCircle	Computes the intersections between a line and circle. Returns the number of intersections and stores the found intersections in the provided point parameters.
GetOrg	Origin point coordinates of a ECircle object.
GetOrgAngle	Angular position from where the ECircle object extents.
GetPoint	Returns the coordinates of a particular point specified by its location along the circle arc.
GetProjectionOfPointOnCircle	Computes the projection of a point on a circle.
GetRadius	Radius of a ECircle object.
operator=	Copies all the data from another ECircle object into the current ECircle object
SetAmplitude	Angular amplitude of the ECircle object.
SetDiameter	Diameter of a ECircle object.
SetFromCenterAndOrigin	Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

SetFromOriginMiddleEnd	Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.
SetRadius	Radius of a ECircle object.

ECircle::GetAmplitude

ECircle::SetAmplitude

Angular amplitude of the ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAmplitude() const

void SetAmplitude(float amplitude)
```

Remarks

The default value is 360. A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle::GetApex

Apex point coordinates of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetApex() const
```

ECircle::GetApexAngle

Angular position at the apex of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetApexAngle() const
```

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle::GetArcLength

Circle arc length of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetArcLength() const
```

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance.

ECircle::CopyTo

Copies all the data of the current ECircle object into another ECircle object and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ECircle& other
 )

ECircle* CopyTo(
 ECircle* other
 )
```

Parameters

other

Pointer to the ECircle object in which the current ECircle object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ECircle object will be created and returned.

ECircle::GetDiameter

ECircle::SetDiameter

Diameter of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetDiameter() const

void SetDiameter(float f32Diameter)
```

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the diameter is 100, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

ECircle::GetDirect

Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetDirect() const
```

Remarks

true (default) means that angles increase anticlockwisely in a direct coordinate system, and clockwisely in an inverse coordinate system. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards.

ECircle::Distance

Returns the smallest distance between this ECircle object an another ECircle.

Namespace: Euresys::Open_eVision

[C++]

```
float Distance(
 const ECircle& circle
 )
```

Parameters

circle

The other circle

ECircle::ECircle

Constructs a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
void ECircle(
 )

void ECircle(
 const EPoint& center,
 float diameter,
 float originAngle,
 bool direct
 )

void ECircle(
 const EPoint& center,
 const EPoint& origin,
 bool direct
 )

void ECircle(
 const EPoint& center,
 float diameter,
 float originAngle,
 float amplitude
 )

void ECircle(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 bool fullCircle
 )

void ECircle(
 const ECircle& other
 )
```

Parameters

center
Center coordinates of the circle at its nominal position. The default value is (0,0).

diameter
Nominal diameter of the circle. The default value is 100.

originAngle
Nominal angular origin of the circle. The default value is 0.

direct
true (default) means that angles increase anticlockwisely in a direct coordinate system.

origin
Origin point coordinates of the circle.

amplitude
Nominal angular amplitude of the circle. The default value is 360.

middle
Middle point coordinates of the circle.

end
End point coordinates of the circle.

fullCircle
true (default) in case of a full turn circle. If fullCircle is false, origin and end give the circle's amplitude.

other
Another ECircle object to be copied in the new ECircle object.

ECircle::GetEnd

End point coordinates of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetEnd() const
```

ECircle::GetEndAngle

Angular position of the end of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEndAngle() const
```

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle::GetFull

Flag indicating whether the ECircle object is a full circle or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFull() const
```

Remarks

By default (true), the ECircle object is a full circle.

ECircle::GetDistanceBetweenLineAndCircle

Computes the distance between a line and a circle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetDistanceBetweenLineAndCircle(
 const ELine& line,
 const ECircle& circle,
 bool limited
 )
```

Parameters

line

The line.

circle

The circle.

limited

Indicates if the line and circle parameters should be considered as infinite lines and full circles or as a segments and arcs.

ECircle::GetDistanceBetweenPointAndCircle

Computes the distance between a point and a circle.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDistanceBetweenPointAndCircle(
 const EPoint& pt,
 const ECircle& circle,
 bool limited
 )
```

Parameters

pt

The point.

circle

The circle.

limited

Indicates if the circle parameter should be considered as a full circle or as an arc.

ECircle::GetIntersectionOfCircles

Computes the intersections between two circles. Returns the number of intersections and stores the found intersections in the provided point parameters.

Namespace: Euresys::Open_eVision

```
[C++]
int GetIntersectionOfCircles(
 const ECircle& circle1,
 const ECircle& circle2,
 EPoint& intersection1,
 EPoint& intersection2,
 bool limited
 )
```

Parameters

circle1
The first circle

circle2
The second circle

intersection1
The first intersection

intersection2
The second intersection

limited
Indicates if the circle parameters should be considered as full circles or as arcs.

Remarks

The function returns the number of intersections found. It will return -1 if the two circles are overlapping.

ECircle::GetIntersectionOfLineAndCircle

Computes the intersections between a line and circle. Returns the number of intersections and stores the found intersections in the provided point parameters.

Namespace: Euresys::Open_eVision

[C++]

```
int GetIntersectionOfLineAndCircle(
 const ELine& line,
 const ECircle& circle,
 EPoint& intersection1,
 EPoint& intersection2,
 bool limited
 )
```

Parameters

line
The line

circle
The circle

intersection1
The first intersection

intersection2
The second intersection

limited
Indicates if the line and circle parameters should be considered as infinite lines and full circles or as a segments and arcs.

Remarks

The function returns the number of intersections found.

ECircle::GetPoint

Returns the coordinates of a particular point specified by its location along the circle arc.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range [-1, +1]).

ECircle::GetProjectionOfPointOnCircle

Computes the projection of a point on a circle.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetProjectionOfPointOnCircle(
 const EPoint& pt,
 const ECircle& circle
 )
```

Parameters

pt

The point.

circle

The circle.

ECircle::operator=

Copies all the data from another ECircle object into the current ECircle object

Namespace: Euresys::Open_eVision

[C++]

```
ECircle& operator=(
 const ECircle& other
 )
```

Parameters

other

ECircle object to be copied

ECircle::GetOrg

Origin point coordinates of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetOrg() const
```

ECircle::GetOrgAngle

Angular position from where the ECircle object extents.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOrgAngle() const
```

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircle::GetRadius

ECircle::SetRadius

Radius of a ECircle object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRadius() const

void SetRadius(float f32Radius)
```

Remarks

A ECircle object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the radius is 50, which means 50 pixels when the field of view is not calibrated, and 50 physical units in case of a calibrated field of view.

ECircle::SetFromCenterAndOrigin

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromCenterAndOrigin(
 const EPoint& center,
 const EPoint& origin,
 bool direct
 )
```

Parameters

center

Center coordinates of the circle at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the circle.

direct

true (default) means that angles increase anticlockwisely in a direct coordinate system.

ECircle::SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircle object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 bool fullCircle
 )
```

Parameters

origin

Origin of the circle Arc.

middle

Middle of the circle Arc.

end

End of the circle Arc.

fullCircle

true (default) in case of a full turn circle. If fullCircle is false, origin and end give the circle's amplitude.

Remarks

For example, for a calibrated circle centered on the origin with a radius of 1, whose arc length is 180° and origin angle 0°:
Its origin is (1, 0), its middle (0, 1) and its end (-1, 0).

4.52. ECircleGauge Class

Manages a circle fitting gauge.

Base Class:ECircleShape

Namespace: Euresys::Open_eVision

Methods

AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.
CopyTo	Copies all the data of the current ECircleGauge object into another ECircleGauge object, and returns it.
DisableInnerFiltering	Disables inner sampled point filtering.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
ECircleGauge	Constructs a circle measurement context.
GetAverageDistance	Average distance between the sampled points and the fitted model.
GetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
GetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

GetInnerFilteringEnabled	Getter method for the GetInnerFilteringEnabled property. This property is the flag indicating if the inner sampled point filtering is enabled (true), or not.
GetInnerFilteringThreshold	Sampled point inner filtering threshold.
GetMeasuredCircle	Information pertaining to the fitted circle.
GetMeasuredPeak	Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.
GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.
GetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
GetMinArea	Minimum area value.
GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetNumFilteringPasses	Number of filtering passes for a model fitting operation.
GetNumMeasuredPoints	Number of edge-crossing points along the sample path of the gauge that was inspected with the last call to ECircleGauge::MeasureSample.
GetNumSamples	Number of sampled points during the model fitting operation.
GetNumSkipRanges	Number of skip ranges in the gauge after a call to ECircleGauge::AddSkipRange.
GetNumValidSamples	Number of valid sample points remaining after a model fitting operation.
GetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
GetSample	Allows to retrieve the sample points found along the circle.
GetSamplingStep	Approximate distance between sampled points during a model fitting operation.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ECircleGauge::AddSkipRange method.
GetSmoothing	Number of pixels used for the low-pass filtering operation.
GetThickness	Number of parallel segments used to extract the data profile.
GetThreshold	Threshold level used to delimit significant peaks in the data profile.
GetTolerance	Searching area half thickness of the circle fitting gauge.
GetTransitionChoice	Transition choice.
GetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
GetTransitionType	Transition type.
GetType	Shape type.
GetValid	Flag indicating if at least one valid transition has been found.

HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without circle fitting operation.
operator=	Copies all the data from another ECircleGauge object into the current ECircleGauge object
Plot	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
PlotWithCurrentPen	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.
RemoveAllSkipRanges	Removes all the skip ranges previously created by a call to ECircleGauge::AddSkipRange.
RemoveSkipRange	After a call to ECircleGauge::AddSkipRange, removes the skip range with the given index.
SetActive	Sets the flag indicating whether the gauge is active or not.
SetCircle	Sets the nominal position (center coordinates), diameter, angular origin and amplitude of the circle fitting gauge according to a known circle.
SetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
SetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
SetInnerFilteringThreshold	Sampled point inner filtering threshold.
SetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
SetMinArea	Minimum area value.
SetMinNumFitSamples	Sets the minimum number of samples required for fitting on each side of the shape.
SetNumFilteringPasses	Number of filtering passes for a model fitting operation.
SetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
SetSamplingStep	Approximate distance between sampled points during a model fitting operation.
SetSmoothing	Number of pixels used for the low-pass filtering operation.
SetThickness	Number of parallel segments used to extract the data profile.
SetThreshold	Threshold level used to delimit significant peaks in the data profile.

SetTolerance	Searching area half thickness of the circle fitting gauge.
SetTransitionChoice	Transition choice.
SetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
SetTransitionType	Transition type.

ECircleGauge::SetActive

Sets the flag indicating whether the gauge is active or not.

Namespace: Euresys::Open_eVision

```
[C++]
void SetActive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying ECircleGauge::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (true).

ECircleGauge::AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 AddSkipRange(
 const OEV_UINT32 start,
 const OEV_UINT32 end
 )
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process.
The ECircleGauge::AddSkipRange method allows to define skip ranges in an ECircleGauge. This means that, at measure time, samples belonging to these ranges will not be taken into account.
A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another.
The range is allowed to be reversed (i.e. end is not required to be greater than start).
Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for ECircleGauge::NumSamples).

ECircleGauge::GetAverageDistance

Average distance between the sampled points and the fitted model.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAverageDistance()
```

Remarks

Irrelevant in case of a point location operation.

ECircleGauge::SetCircle

Sets the nominal position (center coordinates), diameter, angular origin and amplitude of the circle fitting gauge according to a known circle.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCircle(const ECircle& circle)
```

ECircleGauge::CopyTo

Copies all the data of the current ECircleGauge object into another ECircleGauge object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ECircleGauge& other,
 bool recursive
 )

ECircleGauge* CopyTo(
 ECircleGauge* other,
 bool recursive
 )
```

Parameters

other

Pointer to the ECircleGauge object in which the current ECircleGauge object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ECircleGauge object will be created and returned.

ECircleGauge::DisableInnerFiltering

Disables inner sampled point filtering.

Namespace: Euresys::Open_eVision

[C++]

```
void DisableInnerFiltering(
 )
```

Remarks

The inner sampled point filtering is activated as soon as the corresponding ECircleGauge::InnerFilteringThreshold is set.

ECircleGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y
 )
```

Parameters

x

Cursor current X coordinate.

y

Cursor current Y coordinate.

ECircleGauge::Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECircleGauge::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECircleGauge::ECircleGauge

Constructs a circle measurement context.

Namespace: Euresys::Open_eVision

[C++]

```
void ECircleGauge(
 )

void ECircleGauge(
 const ECircleGauge& other
 )
```

Parameters

other

Another ECircleGauge object to be copied in the new ECircleGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the circle measurement context is based on a pre-existing ECircleGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the ECircleGauge::CopyTo method.

ECircleGauge::GetFilteringThreshold

ECircleGauge::SetFilteringThreshold

Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFilteringThreshold()

void SetFilteringThreshold(float f32FilteringThreshold)
```

Remarks

Irrelevant in case of a point location operation.
During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).

ECircleGauge::GetMeasuredPeak

Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
EPeak GetMeasuredPeak(
 OEV_UINT32 index
 )
```

Parameters

index

This argument must be left unchanged from its default value, i.e. ~0 (= 0xFFFFFFFF).

Remarks

ECircleGauge::GetMeasuredPeak returns the information about the derivative peak that meets the following two requirements: 1. It lies on the sample path inspected with the last call to ECircleGauge::MeasureSample, and 1. It is associated with the edge-crossing point along the latter sample path that is selected by the transition choice parameter (cf. ECircleGauge::TransitionChoice).

Note. For this method to succeed, it is necessary to previously call ECircleGauge::MeasureSample.

ECircleGauge::GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetMeasuredPoint(
 OEV_UINT32 index
 )
```

Parameters

index

This argument must be left unchanged from its default value, i.e. ~0 (= 0xFFFFFFFF).

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current ECircleGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry. ECircleGauge::GetMeasuredPoint returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to ECircleGauge::MeasureSample, and 1. Among all the sample points along the latter sample path, it is the one selected by the ECircleGauge::TransitionChoice property.

Note. For this method to succeed, it is necessary to previously call ECircleGauge::MeasureSample.

ECircleGauge::GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void GetMinNumFitSamples(
 int& side0,
 int& side1,
 int& side2,
 int& side3
 )
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

ECircleGauge::GetSample

Allows to retrieve the sample points found along the circle.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSample(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSample(
 ESamplePoint& pt,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure to receive the position of the sample point.

index

The sample index

Remarks

The method provides the sample point corresponding to the supplied index. The returned value corresponds to the sample validity.

ECircleGauge::GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ECircleGauge::AddSkipRange method.

Namespace: Euresys::Open_eVision

```
[C++]
void GetSkipRange(
 const OEV_UINT32 index,
 OEV_UINT32& start,
 OEV_UINT32& end
 )
```

Parameters

index

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

ECircleGauge::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool daughters
 )
```

Parameters

daughters

true if the daughters gauges handles have to be considered as well.

ECircleGauge::GetHVConstraint

ECircleGauge::SetHVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetHVConstraint()
void SetHVConstraint(bool bHVConstraint)
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

ECircleGauge::GetInnerFilteringEnabled

Getter method for the GetInnerFilteringEnabled property. This property is the flag indicating if the inner sampled point filtering is enabled (true), or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetInnerFilteringEnabled()
```

Remarks

The inner sampled point filtering is activated as soon as the corresponding ECircleGauge::InnerFilteringThreshold is set. To disable inner filtering, use the ECircleGauge::DisableInnerFiltering method.

ECircleGauge::GetInnerFilteringThreshold

ECircleGauge::SetInnerFilteringThreshold

Sampled point inner filtering threshold.

Namespace: Euresys::Open_eVision

```
[C++]
float GetInnerFilteringThreshold()
void SetInnerFilteringThreshold(float f32InnerFilteringThreshold)
```

Remarks

If inner filtering is enabled, the sampled points that have been found inside the measured circle are filtered in regard of their distance to it. If this distance is greater than the threshold, the sampled point is set as invalid, and removed from the measure. This distance is in physical units. The inner sampled point filtering is activated as soon as the corresponding threshold is set. To disable inner filtering, use the ECircleGauge::DisableInnerFiltering method.

ECircleGauge::Measure

Triggers the point location or the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
void Measure(
 EROIBW8* sourceImage
 )
void Measure(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Pointer to the source image.

region

Region to use with the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ECircleGauge::GetMeasuredCircle

Information pertaining to the fitted circle.

Namespace: Euresys::Open_eVision

```
[C++]
ECircle GetMeasuredCircle()
```

Remarks

Use method EShape::GetFound to get the status of the measurement. ECircleGauge::MeasuredCircle returns a successful fitted circle if EShape::GetFound is true, otherwise it returns the orginal (nominal) circle.

ECircleGauge::MeasureSample

Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureSample(
 EROIBW8* sourceImage,
 OEV_UINT32 pathIndex
 )

void MeasureSample(
 EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 pathIndex
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

region

Region on which to measure.

Remarks

This method stores its results into a temporary variable inside the ECircleGauge object.

ECircleGauge::MeasureWithoutFitting

Triggers the point location without circle fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureWithoutFitting(
 EROIBW8* sourceImage
 )

void MeasureWithoutFitting(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Source image.

region

Region on which to measure.

Remarks

This method performs the actual measurement for each transition, but does not perform the circle fitting. This means that individual samples will be available through the ECircleGauge::GetSample method, but the gauge position will not be changed. Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

ECircleGauge::GetMinAmplitude

ECircleGauge::SetMinAmplitude

Offset added to the Threshold when a peak is to be detected.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinAmplitude()

void SetMinAmplitude(OEV_UINT32 un32MinAmplitude)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value. To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected. When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

ECircleGauge::GetMinArea

ECircleGauge::SetMinArea

Minimum area value.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinArea()

void SetMinArea(OEV_UINT32 un32MinArea)
```

Remarks

A transition is detected if its derivative peak reaches Threshold + MinAmplitude value, and then declared valid if the area between the peak curve and the horizontal at level Threshold reaches the MinArea value.

ECircleGauge::GetNumFilteringPasses

ECircleGauge::SetNumFilteringPasses

Number of filtering passes for a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumFilteringPasses()

void SetNumFilteringPasses(OEV_UINT32 un32NumFilteringPasses)
```

Remarks

Irrelevant in case of a point location operation.
During a filtering pass, the points that are too distant from the model are discarded.
During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).
By default (the number of filtering passes is 0), the outliers rejection process is disabled.

ECircleGauge::GetNumMeasuredPoints

Number of edge-crossing points along the sample path of the gauge that was inspected with the last call to ECircleGauge::MeasureSample.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumMeasuredPoints()
```

Remarks

Note. For this method to succeed, it is necessary to previously call ECircleGauge::MeasureSample.

ECircleGauge::GetNumSamples

Number of sampled points during the model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamples() const
```

Remarks

Irrelevant in case of a point location operation.
After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

ECircleGauge::GetNumSkipRanges

Number of skip ranges in the gauge after a call to ECircleGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSkipRanges() const
```

ECircleGauge::GetNumValidSamples

Number of valid sample points remaining after a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumValidSamples()
```

Remarks

Irrelevant in case of a point location operation.
After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

ECircleGauge::operator=

Copies all the data from another ECircleGauge object into the current ECircleGauge object

Namespace: Euresys::Open_eVision

[C++]

```
ECircleGauge& operator=(
 const ECircleGauge& other
 )
```

Parameters

other

ECircleGauge object to be copied

ECircleGauge::Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void Plot(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.

height
Height of the plot.

originX
Origin point coordinates of the plot along the X axis.

originY
Origin point coordinates of the plot along the Y axis.

color
The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to ECircleGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call ECircleGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECircleGauge::PlotWithCurrentPen

This method is deprecated.

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void PlotWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.
height
Height of the plot.
originX
Origin point coordinates of the plot along the X axis.
originY
Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to ECircleGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call ECircleGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECircleGauge::Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

Namespace: Euresys::Open_eVision

[C++]

```
void Process(
 EROIBW8* sourceImage,
 bool daughters
 )

void Process(
 EROIBW8* sourceImage,
 const ERegion& region,
 bool daughters
 )
```

Parameters

sourceImage
Pointer to the source image.
daughters
Flag indicating whether the daughters shapes inherit of the same behavior.
region
Region to use with the source image.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

ECircleGauge::GetRectangularSamplingArea

ECircleGauge::SetRectangularSamplingArea

Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRectangularSamplingArea()

void SetRectangularSamplingArea(bool bRectangularSamplingArea)
```

Remarks

By default, this flag is set to true: the sampling area always remains a rectangle. Setting this property is only useful when the thickness transition parameter is greater than 1. In fact, when thickness transition parameter is equal to 1, rectangle and parallelogram reduce to the same segment.

ECircleGauge::RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to ECircleGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveAllSkipRanges(
 )
```

ECircleGauge::RemoveSkipRange

After a call to ECircleGauge::AddSkipRange, removes the skip range with the given index.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveSkipRange(
 const OEV_UINT32 index
 )
```

Parameters

index

Index of the skip range to remove, as returned by ECircleGauge::AddSkipRange.

ECircleGauge::GetSamplingStep

ECircleGauge::SetSamplingStep

Approximate distance between sampled points during a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
float GetSamplingStep()
void SetSamplingStep(float f32SamplingStep)
```

Remarks

Irrelevant in case of a point location operation.
To fit a model, a series of point location operations are performed along the model. The point location gauges spacing is given by the sampling step.
By default, the sampling step is set to 5, which means 5 pixels when the field of view is not calibrated, and 5 physical units in case of a calibrated field of view.
Be aware that if the sampling step is too large, the number of sampled points along the model will not be sufficient enough to accurately locate it.

ECircleGauge::SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void SetMinNumFitSamples(
 int side0,
 int side1,
 int side2,
 int side3
 )
```

Parameters

side0

Required number of samples to correctly fit the circle. The default value is 3. It is the only parameter taken into account.

side1

Not used.

side2

Not used.

side3

Not used.

Remarks

Irrelevant in case of a point location operation. When the ECircleGauge::Measure method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ECircleGauge::GetSmoothing

ECircleGauge::SetSmoothing

Number of pixels used for the low-pass filtering operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetSmoothing()
void SetSmoothing(OEV_UINT32 un32Smoothing)
```

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

ECircleGauge::GetThickness

ECircleGauge::SetThickness

Number of parallel segments used to extract the data profile.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetThickness()
void SetThickness(OEV_UINT32 un32Thickness)
```

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

ECircleGauge::GetThreshold

ECircleGauge::SetThreshold

Threshold level used to delimit significant peaks in the data profile.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 un32Threshold)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value. To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected. When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

ECircleGauge::GetTolerance

ECircleGauge::SetTolerance

Searching area half thickness of the circle fitting gauge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetTolerance()
void SetTolerance(float tolerance)
```

Remarks

A circle fitting gauge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), the angular position from where it extents, its angular amplitude and its outline tolerance. By default, the searching area thickness of the circle fitting gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

ECircleGauge::GetTransitionChoice

ECircleGauge::SetTransitionChoice

Transition choice.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::ETransitionChoice GetTransitionChoice()
void SetTransitionChoice(Euresys::Open_eVision::ETransitionChoice eTransitionChoice)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. In case of ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd transition choice, set ECircleGauge::TransitionIndex to specify the desired transition. By default, the selected transition corresponds to the one with the largest amplitude (ETransitionChoice_LargestAmplitude).

ECircleGauge::GetTransitionIndex

ECircleGauge::SetTransitionIndex

Index (from 0 on) of the transition to be retained when the transition choice parameter is set to NthFromBegin or NthFromEnd.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetTransitionIndex()
void SetTransitionIndex(OEV_UINT32 un32TransitionIndex)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. By default, the first transition is retained (the index value is 0).

ECircleGauge::GetTransitionType

ECircleGauge::SetTransitionType

Transition type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionType GetTransitionType()
void SetTransitionType(Euresys::Open_eVision::ETransitionType eTransitionType)
```

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object. By default, the searched transition type is indifferently a black to white or a white to black transition (ETransitionType_BwOrWb).

ECircleGauge::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

ECircleGauge::GetValid

Flag indicating if at least one valid transition has been found.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetValid()
```

Remarks

A false value means that no measurement has been performed. A true value means that a transition was found along the sample path inspected with the last call to ECircleGauge::MeasureSample, and thus a point has been measured.

4.53. ECircleRegion Class

Manages a complete context for an ERegion shaped like a circle.

Base Class:ERegion

Namespace: Euresys::Open_eVision

Methods

Drag	Moves the specified handle to a new position and updates all placement parameters of the region.

ECircleRegion	Constructs an ECircleRegion context.
GetCenter	Center of the region
GetRadius	Radius of the region
HitTest	Detects if the cursor is placed over one of the dragging handles.
Load	Loads the ECircleRegion. The given ESerializer must have been created for reading.
operator!=	Checks if this ECircleRegion instance is not strictly equal to another
operator=	Assignment operator.
operator==	Checks if this ECircleRegion instance is strictly equal to another
Save	Saves the ECircleRegion. The given ESerializer must have been created for writing.
Scale	Creates a new region by scaling the ECircleRegion.
SetCenter	Center of the region
SetRadius	Radius of the region
Translate	Creates a new ECircleRegion by translating the ECircleRegion.

ECircleRegion::GetCenter

ECircleRegion::SetCenter

Center of the region

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

ECircleRegion::Drag

Moves the specified handle to a new position and updates all placement parameters of the region.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal pan offset. By default, no pan is added.

panY

Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with ECircleRegion::HitTest and ECircleRegion::Drag.

ECircleRegion::ECircleRegion

Constructs an ECircleRegion context.

Namespace: Euresys::Open_eVision

[C++]

```
void ECircleRegion(
 )

void ECircleRegion(
 float centerX,
 float centerY,
 float radius
 )

void ECircleRegion(
 const EPoint& center,
 float radius
 )
```

```
void ECircleRegion(
 const EPoint& pt1,
 const EPoint& pt2,
 const EPoint& pt3
 )
void ECircleRegion(
 const ECircle& circle
 )
void ECircleRegion(
 const ECircleRegion& other
 )
```

Parameters

centerX

The abscissa of the center of the ECircleRegion.

centerY

The ordinate of the center of the ECircleRegion.

radius

The radius of the ECircleRegion.

center

The center of the ECircleRegion.

pt1

One of the three points defining the ECircleRegion.

pt2

One of the three points defining the ECircleRegion.

pt3

One of the three points defining the ECircleRegion.

circle

The result of an ECircleGauge object.

other

ECircleRegion context to copy.

Remarks

When defining a ECircleRegion, the resulting radius value must not be 0 else an EError is thrown.
When defining a ECircleRegion, the resulting radius value must be small enough so that the region fit in memory.
When defining a ECircleRegion with three points, they must be non aligned else an EError is thrown.

ECircleRegion::HitTest

Detects if the cursor is placed over one of the dragging handles.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EEditionMode HitTest(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal pan offset. By default, no pan is added.

panY

Vertical pan offset. By default, no pan is added.

Remarks

Returns a handle identifier, as defined by EEditionMode.
If zooming and/or panning were used when drawing the region, the same values must be used with ECircleRegion::HitTest and ECircleRegion::Drag.

ECircleRegion::Load

Loads the ECircleRegion. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ECircleRegion::operator!=

Checks if this ECircleRegion instance is not strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const ECircleRegion& other
 )
```

Parameters

other
Reference to the other ECircleRegion instance

ECircleRegion::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
ECircleRegion& operator=(
 const ECircleRegion& other
 )
```

Parameters

other
Reference to the ECircleRegion used for the assignment

ECircleRegion::operator==

Checks if this ECircleRegion instance is strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ECircleRegion& other
 )
```

Parameters

other

Reference to the other ECircleRegion instance

ECircleRegion::GetRadius

ECircleRegion::SetRadius

Radius of the region

Namespace: Euresys::Open_eVision

```
[C++]
float GetRadius() const
void SetRadius(float radius)
```

ECircleRegion::Save

Saves the ECircleRegion. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

ECircleRegion::Scale

Creates a new region by scaling the ECircleRegion.

Namespace: Euresys::Open_eVision

[C++]

```
ECircleRegion Scale(
 float scale
 )
EEllipseRegion Scale(
 float scaleX,
 float scaleY
 )
```

Parameters

scale
Isotropic scale.
scaleX
Horizontal scale.
scaleY
Vertical scale.

ECircleRegion::Translate

Creates a new ECircleRegion by translating the ECircleRegion.

Namespace: Euresys::Open_eVision

[C++]

```
ECircleRegion Translate(
 float dx,
 float dy
 )
```

Parameters

dx
Horizontal translation in pixel value
dy
Vertical translation in pixel value

4.54. ECircleShape Class

Manages a circle shape.

Base Class:EShape
Derived Class(es):ECircleGauge
Namespace: Euresys::Open_eVision

Methods

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copies all the data of the current ECircleShape object into another ECircleShape object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
ECircleShape	Constructs a ECircleShape object.
GetAmplitude	Angular amplitude of the ECircleShape object.
GetAngle	The angle of the frame.
GetApex	Apex point coordinates of a ECircleShape object.
GetApexAngle	Angular position at the apex of a ECircleShape object.
GetArcLength	Circle arc length of a ECircleShape object.
GetCenter	Center point of the shape.
GetCenterX	Abscissa of the origin point of the shape.
GetCenterY	Ordinate of the origin point of the shape.
GetCircle	The circle.
GetDiameter	Diameter of a ECircleShape object.
GetDirect	Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.
GetEnd	End point coordinates of a ECircleShape object.
GetEndAngle	Angular position of the end of a ECircleShape object.
GetFull	Flag indicating whether the ECircleShape object is a full circle or not.
GetOrg	Origin point coordinates of a ECircleShape object.
GetOrgAngle	Angular position from where the ECircleShape object extents.
GetPoint	Returns the coordinates of a particular point specified by its location along the circle arc.
GetRadius	Radius of a ECircleShape object.
GetScale	The scale of the frame.
GetType	Shape type.
HitTest	Checks if there is a handle under the cursor.
operator=	Copies all the data from another ECircleShape object into the current ECircleShape object

SetAmplitude	Angular amplitude of the ECircleShape object.
SetAngle	The angle of the frame.
SetCenter	Center point of the shape.
SetCenterXY	-
SetCircle	The circle.
SetDiameter	Diameter of a ECircleShape object.
SetFromCenterAndOrigin	Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircleShape object.
SetFromOriginMiddleEnd	Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircleShape object.
SetRadius	Radius of a ECircleShape object.
SetScale	The scale of the frame.

ECircleShape::GetAmplitude

ECircleShape::SetAmplitude

Angular amplitude of the ECircleShape object.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAmplitude() const
void SetAmplitude(float ampl)
```

Remarks

The default value is 360. A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircleShape::GetAngle

ECircleShape::SetAngle

The angle of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
void SetAngle(float f32Angle)
```

ECircleShape::GetApex

Apex point coordinates of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetApex() const
```

ECircleShape::GetApexAngle

Angular position at the apex of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetApexAngle() const
```

Remarks

A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircleShape::GetArcLength

Circle arc length of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetArcLength() const
```

Remarks

A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance.

ECircleShape::GetCenter

ECircleShape::SetCenter

Center point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

ECircleShape::GetCenterX

Abscissa of the origin point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

ECircleShape::GetCenterY

Ordinate of the origin point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

ECircleShape::GetCircle

ECircleShape::SetCircle

The circle.

Namespace: Euresys::Open_eVision

[C++]

```
ECircle GetCircle() const
void SetCircle(const ECircle& circle)
```

ECircleShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

ECircleShape::CopyTo

Copies all the data of the current ECircleShape object into another ECircleShape object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ECircleShape& other,
 bool recursive
 )

ECircleShape* CopyTo(
 ECircleShape* dest,
 bool bRecursive
 )
```

Parameters

other

Pointer to the ECircleShape object in which the current ECircleShape object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

dest

-

bRecursive

-

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ECircleShape object will be created and returned.

ECircleShape::GetDiameter

ECircleShape::SetDiameter

Diameter of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetDiameter() const
void SetDiameter(float f32Diameter)
```

Remarks

A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the diameter is 100, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

ECircleShape::GetDirect

Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetDirect() const
```

Remarks

true (default) means that angles increase anticlockwisely in a direct coordinate system, and clockwisely in an inverse coordinate system. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards.

ECircleShape::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

-

n32CursorY

-

ECircleShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECircleShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECircleShape::ECircleShape

Constructs a ECircleShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void ECircleShape(
 const ECircleShape& other
 )

void ECircleShape(
 )

void ECircleShape(
 const EPoint& center,
 float diameter,
 float originAngle,
 bool direct
 )
```

```
void ECircleShape(
 const EPoint& center,
 float diameter,
 float originAngle,
 float amplitude
 )
void ECircleShape(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 bool fullCircle
 )
void ECircleShape(
 const EPoint& center,
 const EPoint& origin,
 bool fullCircle
 )
```

Parameters

other
Another ECircleShape object to be copied in the new ECircleShape object.

center
Center coordinates of the circle at its nominal position. The default value is (0,0).

diameter
Nominal diameter of the circle. The default value is 100.

originAngle
Nominal angular origin of the circle. The default value is 0.

direct
true (default) means that angles increase anticlockwisely in a direct coordinate system.

amplitude
Nominal angular amplitude of the circle. The default value is 360.

origin
Origin point coordinates of the circle.

middle
Middle point coordinates of the circle.

end
End point coordinates of the circle.

fullCircle
true (default) in case of a full turn circle. If fullCircle is false, origin and end give the circle's amplitude.

ECircleShape::GetEnd

End point coordinates of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetEnd() const
```

ECircleShape::GetEndAngle

Angular position of the end of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEndAngle() const
```

Remarks

A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircleShape::GetFull

Flag indicating whether the ECircleShape object is a full circle or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFull() const
```

Remarks

By default (true), the ECircleShape object is a full circle.

ECircleShape::GetPoint

Returns the coordinates of a particular point specified by its location along the circle arc.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range [-1, +1]).

ECircleShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

ECircleShape::operator=

Copies all the data from another ECircleShape object into the current ECircleShape object

Namespace: Euresys::Open_eVision

```
[C++]
ECircleShape& operator=(
 const ECircleShape& other
 )
```

Parameters

other

ECircleShape object to be copied

ECircleShape::GetOrg

Origin point coordinates of a ECircleShape object.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetOrg() const
```

ECircleShape::GetOrgAngle

Angular position from where the ECircleShape object extents.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOrgAngle() const
```

Remarks

A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ECircleShape::GetRadius

ECircleShape::SetRadius

Radius of a ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRadius() const

void SetRadius(float f32Radius)
```

Remarks

A ECircleShape object is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), the angular position from where it extents, its angular amplitude, and its outline tolerance. By default, the radius is 50, which means 50 pixels when the field of view is not calibrated, and 50 physical units in case of a calibrated field of view.

ECircleShape::GetScale

ECircleShape::SetScale

The scale of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetScale() const
void SetScale(float f32Scale)
```

ECircleShape::SetCenterXY

-

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

-

centerY

-

ECircleShape::SetFromCenterAndOrigin

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromCenterAndOrigin(
 const EPoint& center,
 const EPoint& origin,
 bool direct
 )
```

Parameters

center

Center coordinates of the circle at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the circle.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system.

ECircleShape::SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, angular position and amplitude) of an ECircleShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 bool fullCircle
 )
```

Parameters

origin

Origin point coordinates of the circle.

middle

Middle point coordinates of the circle.

end

End point coordinates of the circle.

fullCircle

true (default) in case of a full turn circle. If fullCircle is false, origin and end give the circle's amplitude.

ECircleShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EShapeType GetType()
```

4.55. EClassificationDataset Class

EClassificationDataset manages a dataset of images.

A dataset is a collection of images with different types of labeling: labeling for the classification of images, labeling for the segmentation of pixels, and/or labeling for the detection of objects (classification and localization).
The dataset maintains 3 sets of labels for each type of labeling:
- the classification labels that characterize an entire image;
- the segmentation labels that characterize the pixels of an image; and
- the object labels that characterize axis-aligned rectangle region of an image.

The classification and segmentation labels are entirely user defined. The set of segmentation labels will always contain at least the "Background" label representing pixels of the images that have no relevant information for your task (for example, in a defect segmentation application, the "Background" pixels would be the pixels without any defects).

For each type of labeling, an image can either be unlabeled or labeled. When an image in unlabelled for a given type of labeling, the image won't be used for training a deep learning tool that requires this type of labeling.
The image in the dataset can be stored as path to an image file or as an Open eVision image structure. Supported structures are 8-bits monochrome (EImageBW8), 16-bits monochrome (EImageBW16), and 24-bits color (EROIC24, EImageC24).
The dataset associates with each image a region of interest and a mask/don't care area. By default, the region of interest of an image is its full extent and its mask is empty.
A EClassificationDataset object is also responsible for providing tools to do data augmentation. Data augmentation is the process of generating new images on-the-fly by applying affine transformations to those already in the dataset. Data augmentation allows a deep neural network to be invariant to the applied transformation without having to capture and label real world images containing those transformations.

A dataset can contain images with different sizes (with and height of their region of interest). However, the dataset has a default resolution (see EClassificationDataset and EClassificationDataset) that is used by deep learning tools that require the same input image size such as the EClassifier. When the images have different sizes, the default resolution will be the resolution of the region of interest of the first image added to the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

AddImage	Adds an image to the dataset. The image can be specified by its path on the filesystem (parameter imagePath) or by an Open eVision mage buffer (parameter img). By default, an image will have no classification label and no segmentation. The label of the image can be directly specified when adding the image to the dataset. If the given label was not in the classification labels of the dataset, it will be automatically added to them. If no region of interest and/or mask is specified, the region of interest of the image will be its full extent and its mask will be empty. The method returns -1 if there was an error when inserting the image in the dataset or a numeric identifier greater or equal to 0 that can be used to access and manipulate the image in the dataset.
AddImageObject	Adds an object to the image.
AddImages	Adds all the images present in the directory specified by the parameter path and whose filename matches the filter. By default, all the images will have no classification label and no classification. However, a label can be directly specified for all of the images. The method returns the number of images added to the dataset.
AddLabel	Adds a label to the dataset.
AddObjectLabel	Adds an object label.
AddRegionToSegment	Adds a region (see ERegion) to the given segment of an image. If, before this call, the image had no segmentation, the pixels not in the given region will be set to the background segmentation label.
AddSegmentationLabel	Adds a segmentation label. The index of the new segmentation labels is returned by the method.
Clear	Clears the datasets.
EClassificationDataset	Constructs a EClassificationDataset object.
Export	Exports the dataset and its images to the given directory. - A new EClassificationDataset object with relative paths to the images is saved into the given directory. - The exported images will be placed in a sub directory named "Images". By default, the images will be saved under the filename "Image_[id].[ext]" where [id] is the index of the image in the dataset and [ext] is the image extension. If fileType is set to EImageFileType_Auto, [ext] will be the same as the original image. Otherwise, [ext] will be deduced from the specified file type. If keepFilename is set to true, the images will keep their original filename (except for their extensions). In two images have the same filename, an index will be automatically added to avoid nay conflict. A width, height, and number of channels can be optionally specified to reformat all the images in the dataset. Note that, in this case, only the ROI of the image will be exported.

ExtractSplit	Creates a new EClassificationDataset containing only the images of the given split type.
GetAbsoluteMaxOverlap	Absolute maximum overlap between objects in the dataset.
GetAvailableImageAnnotationFormat	Image annotation file formats that are available next to the image file.
GetChannels	Number of channels of the first image added to the dataset.
GetEnableDataAugmentation	Enable data augmentation.
GetEnableHorizontalFlip	Enable horizontal flipping in data augmentation.
GetEnableVerticalFlip	Enable vertical flipping in data augmentation.
GetGaussianNoiseMaximumStandardDeviation	The Gaussian noise maximum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EClassificationDataset::GaussianNoiseMinimumStandardDeviation and EClassificationDataset::GaussianNoiseMaximumStandardDeviation (also called the normal distribution). Its value must be superior or equal to EClassificationDataset::GaussianNoiseMinimumStandardDeviation.
GetGaussianNoiseMinimumStandardDeviation	The Gaussian noise minimum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EClassificationDataset::GaussianNoiseMinimumStandardDeviation and EClassificationDataset::GaussianNoiseMaximumStandardDeviation (also called the normal distribution). Its value must be betwteen 0 and EClassificationDataset::GaussianNoiseMaximumStandardDeviation.
GetHeight	Height of the region of interest of the first image added to the dataset.
GetImageCopy	Gets a copy of the i-th image of the dataset. For the variant where a pointer is returned, the user is responsible for clearing the memory of the returned pointer.
GetImageCopyWithDataAugmentation	Generates a new image from the i-th image of the dataset. For the variant where a pointer is returned, the user is responsible for clearing the memory of the returned pointer.
GetImageLabel	Gets the label of the i-th image of the dataset. An exception is thrown if the image has no classification label.
GetImageNumObjects	Number of objects for the speficied image.
GetImageObject	Gets the specified object for the given image.
GetImageObjects	Gets all the objects of the specified image.
GetImagePath	Gets the path of the i-th image of the dataset. If the image was not given using a path, the method will throw an exception.

GetImages	Gets a copy of all images in the dataset. If data augmentation is enabled, the returned images will be augmented versions of the ones in the dataset. The caller is responsible for clearing the memory allocated for each image.
GetImagesIndexesWithLabel	Gets a list of index corresponding to the images associated with the given label
GetImagesIndexesWithNoLabel	Gets a list of index corresponding to the images with no classification labels.
GetLabel	Gets the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1)
GetLabeledImagesIndexes	Gets a list of index corresponding to the images that have a classification label.
GetLabelWeight	Gets the weight associated to the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1)
GetMask	Gets the mask region/don't care area for the given image. The mask is defined with respect to the region of interest of the image.
GetMaxBrightnessOffset	Maximum absolute brightness offset. Its value must be between 0 and 1.
GetMaxContrastGain	Maximum contrast gain. Its value must be strictly positive and over EClassificationDataset::MinContrastGain.
GetMaxGamma	Maximum gamma for gamma correction. Its value must be higher than EClassificationDataset::MinGamma.
GetMaxHorizontalShear	Maximum absolute horizontal shear. It is represented as an angle from the vertical direction. Its value must be between 0 and 90 degrees.
GetMaxHorizontalShift	Maximum horizontal shift for data augmentation. The horizontal shift will be between -EClassificationDataset::MaxHorizontalShift and +EClassificationDataset::MaxHorizontalShift.
GetMaxHueOffset	Maximum absolute hue offset. Its value must be between 0 and 180 degrees.
GetMaxNumObjectPerImage	Maximum number of objects for an image in the dataset. If no images has object labeling (see EClassificationDataset::HasObjectLabeling), the method returns -1.
GetMaxRotationAngle	Maximum rotation angle for data augmentation. The rotation angle will be between -EClassificationDataset::MaxRotationAngle and +EClassificationDataset::MaxRotationAngle.
GetMaxSaturationGain	Maximum saturation gain. Its value must be over or equal to EClassificationDataset::MinSaturationGain.
GetMaxScale	Maximum scaling allowed for data augmentation.
GetMaxVerticalShear	Maximum absolute vertical shear. It is represented as an angle from the horizontal direction. Its value must be between 0 and 90 degrees.

GetMaxVerticalShift	Maximum vertical shift for data augmentation. The vertical shift will be between -EClassificationDataset::MaxHorizontalShift and +EClassificationDataset::MaxHorizontalShift.
GetMinContrastGain	Minimum contrast gain. Its value must be strictly positive and below EClassificationDataset::MaxContrastGain.
GetMinGamma	Minimum gamma for gamma correction. Its value must be strictly positive and below EClassificationDataset::MaxGamma.
GetMinSaturationGain	Minimum saturation gain. Its value must be strictly positive.
GetMinScale	Minimum scaling allowed for data augmentation.
GetNumImageFiles	Number of different image files contained in the dataset.
GetNumImages	Number of images in the dataset.
GetNumImagesForFile	Get the number of images in the dataset that are associated with the given image file.
GetNumImagesForSegmentationLabel	Number of images containing the given segmentation label.
GetNumImagesWithForegroundSegments	Number of images that have a ground truth segmentation that has foreground segments and is this not entirely composed of background pixels.
GetNumImagesWithObjectLabel	Number of images that contain an object with the given label.
GetNumImagesWithObjectLabeling	Number of images in the dataset that are labelled for object detection.
GetNumImagesWithObjects	Number of images in the dataset that are labelled for object detection and have 1 or more objects.
GetNumImagesWithoutForegroundSegments	Number of images that have a ground truth segmentation that has no foreground segments and is thus entirely composed of background pixels.
GetNumImagesWithoutObjectLabeling	Number of images in the dataset that are not labelled for object detection.
GetNumImagesWithoutObjects	Number of images in the dataset that are labelled for object detection but have been associated with no object.
GetNumImagesWithoutSegmentation	Number of images that doesn't have a ground truth segmentation.
GetNumImagesWithSegmentation	Number of images that have a ground truth segmentation.
GetNumLabeledImages	Number of images that have a classification label.
GetNumLabels	Number of labels in the dataset.
GetNumObjectLabels	Number of object labels.
GetNumObjectsWithLabel	Number of objects with the given label.

GetNumPixelsForSegmentationLabel	Number of pixels assigned to the given segmentation label.
GetNumSegmentationLabels	Number of segmentation labels. A dataset has always at least one segmentation label: "background".
GetNumSegmentedBlobs	Number of segmented blobs in the image for all non-background labels or for a specific label.
GetNumUnlabeledImages	Number of images that doesn't have any classification label.
GetObjectLabel	Object label.
GetObjectLabelWeight	Gets the weight of an objet label.
GetObjectSize	Object size for EasyLocate Interest point.
GetRegionForSegment	Gets a region corresponding to the pixels of the given segment.
GetRegionOfInterestHeight	Region of interest height for the specified image.
GetRegionOfInterestOriginX	Region of interest origin abscissa for the specified image.
GetRegionOfInterestOriginY	Region of interest origin ordinate for the specified image.
GetRegionOfInterestWidth	Region of interest width for the specified image.
GetSaltAndPepperNoiseMaximumDensity	The maximum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EClassificationDataset::SaltAndPepperNoiseMinimumDensity and EClassificationDataset::SaltAndPepperNoiseMaximumDensity) pixels to its minimum or maximum value. Its value must be betwteen EClassificationDataset::SaltAndPepperNoiseMinimumDensity and 1.
GetSaltAndPepperNoiseMinimumDensity	The minimum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EClassificationDataset::SaltAndPepperNoiseMinimumDensity and EClassificationDataset::SaltAndPepperNoiseMaximumDensity) pixels to its minimum or maximum value. Its value must be betwteen 0 and EClassificationDataset::SaltAndPepperNoiseMaximumDensity.
GetSameLabelMaxOverlap	Maximum overlap between objects with the same label in the dataset.
GetSegmentationLabel	Gets the segmentation label.
GetSegmentationLabelWeight	Gets the segmentation label weights.

GetSegmentationMap	Gets the segmentation map of an image. The segmentation map is a 16-bit EROIBW16 image where the value of each pixel is equal to the corresponding segmentation label index (see EClassificationDataset::GetSegmentationLabel). If an image has no segmentation (EClassificationDataset::HasSegmentation), the getter will throw an excpetion.
GetSpeckleNoiseMaximumStandardDeviation	The speckle noise maximum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution with a mean of 1 and with its standard deviation between EClassificationDataset::SpeckleNoiseMinimumStandardDeviation and EClassificationDataset::SpeckleNoiseMaximumStandardDeviation. Its value must be strictly higher than EClassificationDataset::SpeckleNoiseMinimumStandardDeviation.
GetSpeckleNoiseMinimumStandardDeviation	The speckle noise minimum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution with a mean of 1 and with its standard deviation between EClassificationDataset::SpeckleNoiseMinimumStandardDeviation and EClassificationDataset::SpeckleNoiseMaximumStandardDeviation. Its value must be strictly positive and lower than EClassificationDataset::SpeckleNoiseMaximumStandardDeviation.
GetSplit	Generates a split.
GetWidth	Width of the region of interest of the first image added to the dataset.
HasForegroundSegments	Whether the image segmentation contains segments that are not background. If the image has no segmentation, this method throws an exception.
HasLabel	Whether an image has a classification label.
HasObjectLabeling	Whether the image is labelled for object detection and use with ELocator.
HasSegmentation	Whether the image has a segmentation.
ImportPascalVOCXMLAnnotations	Imports Pascal VOC XML annotations (for EasyLocate Bounding Box). This method may add new labels to the object labels. The version that do not take an image index attempts to import the annotation from all images currently in the dataset.
ImportYOLOTXTAnnotations	Imports YOLO TXT annotations (for EasyLocate Bounding Box). You need to specify the list of labels associated with the YOLO TXT annotations through an array of string or through a file containing this list. This method may add new labels to the object labels. The version that do not take an image index attempts to import the annotation from all images currently in the dataset.
IsEmbeddedImage	Whether the image is embedded into the dataset and is not a reference towards an image file.
IsImageFile	Whether the image is stored as a file path towards an image.
Load	Loads a classification dataset from disk.
operator=	Assignment operator

RemoveImage	Removes the image at the given index. All annotations (label, segmentation, objects) will be lost.
RemoveImageObject	Removes an object from an image.
RemoveLabel	Removes a label from the dataset. All the images associated with this label will be set to unlabeled (see EClassificationDataset::HasLabel).
RemoveObjectLabel	Removes an object label.
RemoveSegmentationLabel	Remove a segmentation label. This method sets all the pixels in the dataset assigned to this label to the background label.
ResetImageObjectLabeling	Resets the object labeling for the specified image. This sets the image as having been labelled for object detection with no object present in the image.
ResetSegmentation	Resets the segmentation of an image by setting all pixels to background.
Save	Saves a classification dataset to disk, containing the file paths to the images in the dataset and their associated label.
SetBasePath	Sets the base path for the images specified with a relative path.
SetEnableDataAugmentation	Enable data augmentation.
SetEnableHorizontalFlip	Enable horizontal flipping in data augmentation.
SetEnableVerticalFlip	Enable vertical flipping in data augmentation.
SetGaussianNoiseMaximumStandardDeviation	The Gaussian noise maximum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EClassificationDataset::GaussianNoiseMinimumStandardDeviation and EClassificationDataset::GaussianNoiseMaximumStandardDeviation (also called the normal distribution). Its value must be superior or equal to EClassificationDataset::GaussianNoiseMinimumStandardDeviation.
SetGaussianNoiseMinimumStandardDeviation	The Gaussian noise minimum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EClassificationDataset::GaussianNoiseMinimumStandardDeviation and EClassificationDataset::GaussianNoiseMaximumStandardDeviation (also called the normal distribution). Its value must be betwteen 0 and EClassificationDataset::GaussianNoiseMaximumStandardDeviation.
SetImageLabel	Sets the label of images in the dataset.
SetImageObject	Sets the specified object for the given image.
SetLabel	Sets the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1). This operation does not add a new label to the dataset but simply renames an existing label.

SetLabelWeight	Sets the weight associated to the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1). This operation does not add a new label.
SetMask	Sets the mask region/don't care area for the given image. The mask is defined with respect to the region of interest of the image.
SetMaxBrightnessOffset	Maximum absolute brightness offset. Its value must be between 0 and 1.
SetMaxContrastGain	Maximum contrast gain. Its value must be strictly positive and over EClassificationDataset::MinContrastGain.
SetMaxGamma	Maximum gamma for gamma correction. Its value must be higher than EClassificationDataset::MinGamma.
SetMaxHorizontalShear	Maximum absolute horizontal shear. It is represented as an angle from the vertical direction. Its value must be between 0 and 90 degrees.
SetMaxHorizontalShift	Maximum horizontal shift for data augmentation. The horizontal shift will be between -EClassificationDataset::MaxHorizontalShift and +EClassificationDataset::MaxHorizontalShift.
SetMaxHueOffset	Maximum absolute hue offset. Its value must be between 0 and 180 degrees.
SetMaxRotationAngle	Maximum rotation angle for data augmentation. The rotation angle will be between -EClassificationDataset::MaxRotationAngle and +EClassificationDataset::MaxRotationAngle.
SetMaxSaturationGain	Maximum saturation gain. Its value must be over or equal to EClassificationDataset::MinSaturationGain.
SetMaxScale	Maximum scaling allowed for data augmentation.
SetMaxVerticalShear	Maximum absolute vertical shear. It is represented as an angle from the horizontal direction. Its value must be between 0 and 90 degrees.
SetMaxVerticalShift	Maximum vertical shift for data augmentation. The vertical shift will be between -EClassificationDataset::MaxHorizontalShift and +EClassificationDataset::MaxHorizontalShift.
SetMinContrastGain	Minimum contrast gain. Its value must be strictly positive and below EClassificationDataset::MaxContrastGain.
SetMinGamma	Minimum gamma for gamma correction. Its value must be strictly positive and below EClassificationDataset::MaxGamma.
SetMinSaturationGain	Minimum saturation gain. Its value must be strictly positive.
SetMinScale	Minimum scaling allowed for data augmentation.
SetObjectLabel	Sets an object label.
SetObjectLabelWeight	Sets the weight of an object label.
SetObjectSize	Object size for EasyLocate Interest point.

SetRegionOfInterest	Sets the region of interest for the specified image.
SetSaltAndPepperNoiseMaximumDensity	The maximum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EClassificationDataset::SaltAndPepperNoiseMinimumDensity and EClassificationDataset::SaltAndPepperNoiseMaximumDensity) pixels to its minimum or maximum value. Its value must be betwteen EClassificationDataset::SaltAndPepperNoiseMinimumDensity and 1.
SetSaltAndPepperNoiseMinimumDensity	The minimum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EClassificationDataset::SaltAndPepperNoiseMinimumDensity and EClassificationDataset::SaltAndPepperNoiseMaximumDensity) pixels to its minimum or maximum value. Its value must be betwteen 0 and EClassificationDataset::SaltAndPepperNoiseMaximumDensity.
SetSegmentationLabel	Sets the segmentation labels.
SetSegmentationLabelWeight	Sets the segmentation label weight.
SetSegmentationMap	Sets the segmentation map of an image. The segmentation map is a 16-bit EROIBW16 image where the value of each pixel is equal to the corresponding segmentation label index (see EClassificationDataset::GetSegmentationLabel). If an image has no segmentation (EClassificationDataset::HasSegmentation), the getter will throw an excpetion.
SetSpeckleNoiseMaximumStandardDeviation	The speckle noise maximum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution with a mean of 1 and with its standard deviation between EClassificationDataset::SpeckleNoiseMinimumStandardDeviation and EClassificationDataset::SpeckleNoiseMaximumStandardDeviation. Its value must be strictly higher than EClassificationDataset::SpeckleNoiseMinimumStandardDeviation.
SetSpeckleNoiseMinimumStandardDeviation	The speckle noise minimum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution with a mean of 1 and with its standard deviation between EClassificationDataset::SpeckleNoiseMinimumStandardDeviation and EClassificationDataset::SpeckleNoiseMaximumStandardDeviation. Its value must be strictly positive and lower than EClassificationDataset::SpeckleNoiseMaximumStandardDeviation.
SplitDataset	Splits the dataset in two parts to be used for training and validation respectively.
SplitDatasetForLocator	Splits the dataset in two parts for training and validation of a ELocator tool. Images without object labeling are excluded from the split.
SplitDatasetForSegmentation	Splits the dataset in two parts for a supervised segmenter. The two parts are to be used for training and validation respectively.

UnsetImageLabel	Unset the label of the given image of the dataset. After this operation, the given image will have no classification labels.
UnsetImageObjectLabeling	Unsets the object labeling for the specified image. This sets the image as not having been labelled for object detection. This image won't be use for training with a ELocator tool.
UnsetSegmentation	Unsets the segmentation of an image. After this operation, the image will have no segmentation.

EClassificationDataset::GetAbsoluteMaxOverlap

Absolute maximum overlap between objects in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAbsoluteMaxOverlap() const
```

EClassificationDataset::AddImage

Adds an image to the dataset.
The image can be specified by its path on the filesystem (parameter `imagePath`) or by an Open eVision mage buffer (parameter `img`).
By default, an image will have no classification label and no segmentation. The label of the image can be directly specified when adding the image to the dataset. If the given label was not in the classification labels of the dataset, it will be automatically added to them.
If no region of interest and/or mask is specified, the region of interest of the image will be its full extent and its mask will be empty.
The method returns -1 if there was an error when inserting the image in the dataset or a numeric identifier greater or equal to 0 that can be used to access and manipulate the image in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int AddImage(
 const std::string& imagePath
 )

int AddImage(
 const std::string& imagePath,
 const std::string& label
 )
```

```
int AddImage(
 const std::string& imagePath,
 int originX,
 int originY,
 int width,
 int height,
 const ERegion& mask
 )

int AddImage(
 const std::string& imagePath,
 int originX,
 int originY,
 int width,
 int height,
 const ERegion& mask,
 const std::string& label
 )

int AddImage(
 const EBaseROI& img
 )

int AddImage(
 const EBaseROI& img,
 const std::string& label
 )

int AddImage(
 const EBaseROI& img,
 int originX,
 int originY,
 int width,
 int height,
 const ERegion& mask
 )

int AddImage(
 const EBaseROI& img,
 int originX,
 int originY,
 int width,
 int height,
 const ERegion& mask,
 const std::string& label
 )
```

Parameters

imagePath
The path to an image
label
The label
originX
Region of interest origin abscissa
originY
Region of interest origin ordinate
width
Region of interest width
height
Region of interest height
mask
The mask for the image (with respect to the region of interest)
img
The image

Remarks

When adding Open eVision images (EBaseROI) to a dataset, the dataset will retain a copy of the image.
When specifying an individual region of interest and/or mask, the dataset will retain a copy of these.

EClassificationDataset::AddImageObject

Adds an object to the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void AddImageObject(
 int imageIndex,
 const ELocatorObject& obj
 )

void AddImageObject(
 int imageIndex,
 const std::string& label,
 const ERectangleRegion& box
 )
```

Parameters

imageIndex
Index of the image

obj
Object

label
Label of the object

box
Axis-aligned rectangle

Remarks

The image will be marked as labelled for object detection (EClassificationDataset::HasObjectLabeling equals to true) after a call to this method.
If the label of the object does not exist in the object labels of the dataset, the label will be added to the object labels of the dataset.

EClassificationDataset::AddImages

Adds all the images present in the directory specified by the parameter path and whose filename matches the filter.
By default, all the images will have no classification label and no classification. However, a label can be directly specified for all of the images.
The method returns the number of images added to the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int AddImages(
 const std::string& filter
 )

int AddImages(
 const std::string& filter,
 const std::string& label
 )

int AddImages(
 const std::string& filter,
 int originX,
 int originY,
 int width,
 int height,
 const ERegion& mask
 )
```

```
int AddImages(
 const std::string& filter,
 int originX,
 int originY,
 int width,
 int height,
 const ERegion& mask,
 const std::string& label
 )
```

Parameters

filter

A glob filter

label

A label.

originX

Region of interest origin abscissa

originY

Region of interest origin ordinate

width

Region of interest width

height

Region of interest height

mask

The mask for the images

Remarks

The filter is a glob pattern. This means the wildcard characters "*" and "?" correspond to "zero or more character" and "a single character" respectively. For example, the filter "*_good_*.png" will match any filename that contains the string "_good_" and has a png extension.

EClassificationDataset::AddLabel

Adds a label to the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void AddLabel(
 const std::string& label
 )
```

Parameters

label

Name of the new label

EClassificationDataset::AddObjectLabel

Adds an object label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddObjectLabel(
 const std::string& label,
 float weight
 )
```

Parameters

label
Label to add

weight
Weight of the label

EClassificationDataset::AddRegionToSegment

Adds a region (see ERegion) to the given segment of an image. If, before this call, the image had no segmentation, the pixels not in the given region will be set to the background segmentation label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddRegionToSegment(
 int imageIndex,
 int segmentationLabelIndex,
 const ERegion& region
 )

void AddRegionToSegment(
 int imageIndex,
 const std::string& label,
 const ERegion& region
 )
```

Parameters

imageIndex
Image index

segmentationLabelIndex
Segmentation label index

region
Region to add

label
Segmentation label

EClassificationDataset::AddSegmentationLabel

Adds a segmentation label. The index of the new segmentation labels is returned by the method.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int AddSegmentationLabel(
 const std::string& label,
 float labelWeight
 )
```

Parameters

label

Name of the segmentation label

labelWeight

Weight of the segmentation label

EClassificationDataset::SetBasePath

Sets the base path for the images specified with a relative path.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetBasePath(const std::string& basePath)
```

Remarks

The base path is not serialized. It must be set after loading a dataset file.

EClassificationDataset::GetChannels

Number of channels of the first image added to the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetChannels() const
```

EClassificationDataset::Clear

Clears the datasets.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Clear(
 )
```

EClassificationDataset::EClassificationDataset

Constructs a EClassificationDataset object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EClassificationDataset(
 )

void EClassificationDataset(
 const EClassificationDataset& other
 )
```

Parameters

other

Reference to the EClassificationDataset object that should be copied

EClassificationDataset::GetEnableDataAugmentation

EClassificationDataset::SetEnableDataAugmentation

Enable data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableDataAugmentation() const

void SetEnableDataAugmentation(bool enable)
```

EClassificationDataset::GetEnableHorizontalFlip

EClassificationDataset::SetEnableHorizontalFlip

Enable horizontal flipping in data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableHorizontalFlip() const
void SetEnableHorizontalFlip(bool enable)
```

EClassificationDataset::GetEnableVerticalFlip

EClassificationDataset::SetEnableVerticalFlip

Enable vertical flipping in data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableVerticalFlip() const
void SetEnableVerticalFlip(bool enable)
```

EClassificationDataset::Export

Exports the dataset and its images to the given directory. - A new EClassificationDataset object with relative paths to the images is saved into the given directory.
- The exported images will be placed in a sub directory named "Images".
By default, the images will be saved under the filename "Image_[id].[ext]" where [id] is the index of the image in the dataset and [ext] is the image extension. If fileType is set to EImageFileType_Auto, [ext] will be the same as the original image. Otherwise, [ext] will be deduced from the specified file type. If keepFilename is set to true, the images will keep their original filename (except for their extensions). In two images have the same filename, an index will be automatically added to avoid nay conflict. A width, height, and number of channels can be optionally specified to reformat all the images in the dataset. Note that, in this case, only the ROI of the image will be exported.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Export(
 const std::string& directory,
 Euresys::Open_eVision::EImageFileType fileType,
 int quality
 )
void Export(
 const std::string& directory,
 bool keepFilename,
 Euresys::Open_eVision::EImageFileType fileType,
 int quality
 )
```

```
void Export(
 const std::string& directory,
 int width,
 int height,
 int channels,
 Euresys::Open_eVision::EImageFileType fileType,
 int quality
 )
void Export(
 const std::string& directory,
 int width,
 int height,
 int channels,
 bool keepFilenames,
 Euresys::Open_eVision::EImageFileType fileType,
 int quality
 )
```

Parameters

directory

A string containing the full path to the directory.

fileType

File type for the exported file. If EImageFileType_Auto, the same file type as the original image is used.

quality

Quality or compression parameters for EBaseROI::SavePng, EBaseROI::SaveJpeg, or EBaseROI::SaveJpeg2K. A value of -1 means the default value.

keepFilename

Keep the original filename of the images

width

Width of the image.

height

Height of the image.

channels

Number of channels of the image (only 1 and 3 are valid, for grayscale and RGB respectively).

keepFilenames

-

EClassificationDataset::ExtractSplit

Creates a new EClassificationDataset containing only the images of the given split type.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EClassificationDataset ExtractSplit(
 const EDatasetSplit& split,
 Euresys::Open_eVision::EasyDeepLearning::EDatasetType type
 )
```

Parameters

split

Split

type

Type of the split to extract

EClassificationDataset::GetGaussianNoiseMaximumStandardDeviation

EClassificationDataset::SetGaussianNoiseMaximumStandardDeviation

The Gaussian noise maximum standard deviation.
The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EClassificationDataset::GaussianNoiseMinimumStandardDeviation and EClassificationDataset::GaussianNoiseMaximumStandardDeviation (also called the normal distribution).
Its value must be superior or equal to EClassificationDataset::GaussianNoiseMinimumStandardDeviation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetGaussianNoiseMaximumStandardDeviation() const

void SetGaussianNoiseMaximumStandardDeviation(float gaussianMaximumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EClassificationDataset::GetGaussianNoiseMinimumStandardDeviation

EClassificationDataset::SetGaussianNoiseMinimumStandardDeviation

The Gaussian noise minimum standard deviation.
The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EClassificationDataset::GaussianNoiseMinimumStandardDeviation and EClassificationDataset::GaussianNoiseMaximumStandardDeviation (also called the normal distribution).
Its value must be betwteen 0 and EClassificationDataset::GaussianNoiseMaximumStandardDeviation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetGaussianNoiseMinimumStandardDeviation() const
void SetGaussianNoiseMinimumStandardDeviation(float gaussianMinimumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EClassificationDataset::GetAvailableImageAnnotationFormat

Image annotation file formats that are available next to the image file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EImageAnnotationFormat
GetAvailableImageAnnotationFormat(
 const std::string& imageFile
 )
Euresys::Open_eVision::EasyDeepLearning::EImageAnnotationFormat
GetAvailableImageAnnotationFormat(
 int imgId
 )
```

Parameters

imageFile

Path to an image file

imgId

Index of an image

EClassificationDataset::GetImageCopy

Gets a copy of the i-th image of the dataset. For the variant where a pointer is returned, the user is responsible for clearing the memory of the returned pointer.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EBaseROI* GetImageCopy(
 int index
 )
void GetImageCopy(
 int index,
 EBaseROI& img
 )
```

Parameters

index

The index of the image.

img

Image object to copy the image into.

EClassificationDataset::GetImageCopyWithDataAugmentation

Generates a new image from the i-th image of the dataset. For the variant where a pointer is returned, the user is responsible for clearing the memory of the returned pointer.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EBaseROI* GetImageCopyWithDataAugmentation(
 int index,
 Euresys::Open_eVision::EasyDeepLearning::EDLDataAugmentationType generationType
 )

void GetImageCopyWithDataAugmentation(
 int index,
 EBaseROI& img,
 Euresys::Open_eVision::EasyDeepLearning::EDLDataAugmentationType generationType
 )
```

Parameters

index

The index of the image.

generationType

The type of transformation to generate (default: random).

img

Image object to copy the image into.

Remarks

If the data augmentation fails, the method will throw an exception.

EClassificationDataset::GetImageLabel

Gets the label of the i-th image of the dataset. An exception is thrown if the image has no classification label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetImageLabel(
 int index
 )
```

Parameters

index

The index of the image for which to get the label.

EClassificationDataset::GetImageNumObjects

Number of objects for the speficied image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetImageNumObjects(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image

EClassificationDataset::GetImageObject

Gets the specified object for the given image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ELocatorObject GetImageObject(
 int imageIndex,
 int objectIndex
 )
```

Parameters

imageIndex

Index of the image

objectIndex

Index of the object between 0 and EClassificationDataset::GetImageNumObjects

EClassificationDataset::GetImageObjects

Gets all the objects of the specified image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorObject> GetImageObjects(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image

EClassificationDataset::GetImagePath

Gets the path of the i-th image of the dataset. If the image was not given using a path, the method will throw an exception.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetImagePath(
 int index
 )
```

Parameters

index

The index of the image for which to get the path.

EClassificationDataset::GetImages

Gets a copy of all images in the dataset. If data augmentation is enabled, the returned images will be augmented versions of the ones in the dataset.
The caller is responsible for clearing the memory allocated for each image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EBaseROI*> GetImages(
 )

std::vector<Euresys::Open_eVision::EBaseROI*> GetImages(
 const std::string& label
 )
```

Parameters

label

The label.

EClassificationDataset::GetImagesIndexesWithLabel

Gets a list of index corresponding to the images associated with the given label

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<OEV_UINT32> GetImagesIndexesWithLabel(
 const std::string& label
 )
std::vector<OEV_UINT32> GetImagesIndexesWithLabel(
 int labelIndex
 )
```

Parameters

label

The label

labelIndex

The index of the label (starting from 0 to EClassificationDataset::NumLabels - 1)

EClassificationDataset::GetLabel

Gets the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1)

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetLabel(
 int i
 )
```

Parameters

i

Label index

EClassificationDataset::GetLabelWeight

Gets the weight associated to the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1)

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelWeight(
 int index
 )
```

Parameters

index

Label index

EClassificationDataset::GetMask

Gets the mask region/don't care area for the given image. The mask is defined with respect to the region of interest of the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ERegion GetMask(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image for which to get the mask region

EClassificationDataset::GetNumImagesForFile

Get the number of images in the dataset that are associated with the given image file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImagesForFile(
 const std::string& filePath
 )
```

Parameters

filePath

-

Remarks

The number of image files can be different than the number of images of the dataset. An image file can correspond to several dataset images (with different ROI).

EClassificationDataset::GetNumImagesForSegmentationLabel

Number of images containing the given segmentation label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImagesForSegmentationLabel(
 int labelId
 )
int GetNumImagesForSegmentationLabel(
 const std::string& label
 )
```

Parameters

labelId

Index of the segmentation label

label

Segmentation label

EClassificationDataset::GetNumImagesWithObjectLabel

Number of images that contain an object with the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImagesWithObjectLabel(
 int labelIdx
 )
int GetNumImagesWithObjectLabel(
 const std::string& label
 )
```

Parameters

labelIdx

Index of the object label.

label

Label.

EClassificationDataset::GetNumObjectsWithLabel

Number of objects with the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT64 GetNumObjectsWithLabel(
 int labelIdx
 )
```

```
OEV_UINT64 GetNumObjectsWithLabel(
 const std::string& label
 )
```

Parameters

labelIdx

Index of the object label.

label

Label.

EClassificationDataset::GetNumPixelsForSegmentationLabel

Number of pixels assigned to the given segmentation label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT64 GetNumPixelsForSegmentationLabel(
 int labelId
 )

OEV_UINT64 GetNumPixelsForSegmentationLabel(
 const std::string& label
 )
```

Parameters

labelId

Index of the segmentation label

label

Segmentation label

EClassificationDataset::GetNumSegmentedBlobs

Number of segmented blobs in the image for all non-background labels or for a specific label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumSegmentedBlobs(
 int imageId
 )

int GetNumSegmentedBlobs(
 int imageId,
 const std::string& label
 )
```

```
int GetNumSegmentedBlobs(
 int imageId,
 int labelId
 )
```

Parameters

imageId
Image index

label
Label

labelId
Label index

Remarks

A segmented blob is a contiguous area assigned to a label different than "Background".

EClassificationDataset::GetObjectLabel

Object label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetObjectLabel(
 int labelIndex
 )
```

Parameters

labelIndex
Index of the object label between 0 and EClassificationDataset::NumObjectLabels

EClassificationDataset::GetObjectLabelWeight

Gets the weight of an objet label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetObjectLabelWeight(
 int labelIndex
 )

float GetObjectLabelWeight(
 const std::string& label
 )
```

Parameters

labelIndex
Index of the object label

label
Label

EClassificationDataset::GetRegionForSegment

Gets a region corresponding to the pixels of the given segment.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ERegion GetRegionForSegment(
 int imageIndex,
 int segmentationLabelIndex
 )
ERegion GetRegionForSegment(
 int imageIndex,
 const std::string& segmentationLabel
 )
```

Parameters

imageIndex
Image index

segmentationLabelIndex
Segmentation label index

segmentationLabel
Segmentation label

EClassificationDataset::GetRegionOfInterestHeight

Region of interest height for the specified image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetRegionOfInterestHeight(
 int imageIndex
 )
```

Parameters

imageIndex
Index of the image.

EClassificationDataset::GetRegionOfInterestOriginX

Region of interest origin abscissa for the specified image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetRegionOfInterestOriginX(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image.

EClassificationDataset::GetRegionOfInterestOriginY

Region of interest origin ordinate for the specified image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetRegionOfInterestOriginY(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image.

EClassificationDataset::GetRegionOfInterestWidth

Region of interest width for the specified image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetRegionOfInterestWidth(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image.

EClassificationDataset::GetSegmentationLabel

Gets the segmentation label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetSegmentationLabel(
 int index
 )
```

Parameters

index

Index of the segmentation label

Remarks

The segmentation label index 0 is reserved and corresponds to the "background" label. This segmentation label can't be changed.

EClassificationDataset::GetSegmentationLabelWeight

Gets the segmentation label weights.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetSegmentationLabelWeight(
 int index
 )
float GetSegmentationLabelWeight(
 const std::string& label
 )
```

Parameters

index

Index of the segmentation label

label

Segmentation label

EClassificationDataset::GetSegmentationMap

Gets the segmentation map of an image. The segmentation map is a 16-bit EROIBW16 image where the value of each pixel is equal to the corresponding segmentation label index (see EClassificationDataset::GetSegmentationLabel).
If an image has no segmentation (EClassificationDataset::HasSegmentation), the getter will throw an excpetion.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EImageBW16 GetSegmentationMap(
 int imageIndex
 )
```

Parameters

imageIndex

Image index

EClassificationDataset::GetSplit

Generates a split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDatasetSplit GetSplit(
 float trainingProportion,
 float validationPropotion,
 Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType toolType,
 const std::string& goodLabel
 )

EDatasetSplit GetSplit(
 float trainingProportion,
 float validationPropotion,
 Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType toolType,
 const std::string& goodLabel,
 OEV_UINT32 seed
 )

EDatasetSplit GetSplit(
 int numTrainingImages,
 int numValidationImages,
 Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType toolType,
 const std::string& goodLabel
 )

EDatasetSplit GetSplit(
 int numTrainingImages,
 int numValidationImages,
 Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType toolType,
 const std::string& goodLabel,
 OEV_UINT32 seed
 )
```

Parameters

trainingProportion
Approximate proportion of training images

validationPropotion
Approximate proportion of validation images

toolType
Tool type for which to generate the split

goodLabel
The good label for EasySegment Unsupervised split

seed
Seed for randomization

numTrainingImages
Number of training images

numValidationImages
Number of validation images

EClassificationDataset::HasForegroundSegments

Whether the image segmentation contains segments that are not background. If the image has no segmentation, this method throws an exception.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasForegroundSegments(
 int imageId
 )
```

Parameters

imageId
Image index

EClassificationDataset::HasLabel

Whether an image has a classification label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasLabel(
 int index
 )
```

Parameters

index
Index of an image.

EClassificationDataset::HasObjectLabeling

Whether the image is labelled for object detection and use with ELocator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasObjectLabeling(
 int imageIndex
 )
```

Parameters

imageIndex
Index of the image

EClassificationDataset::HasSegmentation

Whether the image has a segmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasSegmentation(
 int imageId
 )
```

Parameters

imageId
Image index

EClassificationDataset::GetHeight

Height of the region of interest of the first image added to the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetHeight() const
```

EClassificationDataset::GetImagesIndexesWithNoLabel

Gets a list of index corresponding to the images with no classification labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<int> GetImagesIndexesWithNoLabel() const
```

EClassificationDataset::ImportPascalVOCXMLAnnotations

Imports Pascal VOC XML annotations (for EasyLocate Bounding Box).
This method may add new labels to the object labels.
The version that do not take an image index attempts to import the annotation from all images currently in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ImportPascalVOCXMLAnnotations(
 int imgId
 )

void ImportPascalVOCXMLAnnotations(
 )
```

Parameters

imgId

The image index for which to import the annotation

EClassificationDataset::ImportYOLOTXTAnnotations

Imports YOLO TXT annotations (for EasyLocate Bounding Box).
You need to specify the list of labels associated with the YOLO TXT annotations through an array of string or through a file containing this list. This method may add new labels to the object labels.
The version that do not take an image index attempts to import the annotation from all images currently in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ImportYOLOTXTAnnotations(
 int imgId,
 const std::string& labelFile
 )

void ImportYOLOTXTAnnotations(
 const std::string& labelFile
 )
```

Parameters

imgId
The image index for which to import the annotation

labelFile
Path to a file containing the label list

EClassificationDataset::IsEmbeddedImage

Whether the image is embedded into the dataset and is not a reference towards an image file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsEmbeddedImage(
 int index
 )
```

Parameters

index
Index of the image.

EClassificationDataset::IsImageFile

Whether the image is stored as a file path towards an image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsImageFile(
 int index
 )
```

Parameters

index
Index of the image.

EClassificationDataset::GetLabeledImagesIndexes

Gets a list of index corresponding to the images that have a classification label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::vector<int> GetLabeledImagesIndexes() const
```

EClassificationDataset::Load

Loads a classification dataset from disk.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

A string containing the full path to the dataset file.

serializer

The serializer.

EClassificationDataset::GetMaxBrightnessOffset

EClassificationDataset::SetMaxBrightnessOffset

Maximum absolute brightness offset. Its value must be between 0 and 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetMaxBrightnessOffset() const
void SetMaxBrightnessOffset(float offset)
```

Remarks

Brightness transformation is performed by adding a value taken between -EClassificationDataset::MaxBrightnessOffset and +EClassificationDataset::MaxBrightnessOffset to each pixel of the normalized image.

EClassificationDataset::GetMaxContrastGain

EClassificationDataset::SetMaxContrastGain

Maximum contrast gain. Its value must be strictly positive and over EClassificationDataset::MinContrastGain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxContrastGain() const
void SetMaxContrastGain(float val)
```

Remarks

Contrast transformation is performed by multiplying each mean-centered pixel value by a gain value taken between EClassificationDataset::MinContrastGain and EClassificationDataset::MaxContrastGain. A contrast transformation does not change the overall brightness of an image.

EClassificationDataset::GetMaxGamma

EClassificationDataset::SetMaxGamma

Maximum gamma for gamma correction. Its value must be higher than EClassificationDataset::MinGamma.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxGamma() const
void SetMaxGamma(float gamma)
```

Remarks

Gamma correction transformation is performed by raising each normalized pixel value to the power of gamma with gamma taken between EClassificationDataset::MinGamma and EClassificationDataset::MaxGamma.

EClassificationDataset::GetMaxHorizontalShear

EClassificationDataset::SetMaxHorizontalShear

Maximum absolute horizontal shear.
It is represented as an angle from the vertical direction. Its value must be between 0 and 90 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxHorizontalShear() const
void SetMaxHorizontalShear(float hShear)
```

EClassificationDataset::GetMaxHorizontalShift

EClassificationDataset::SetMaxHorizontalShift

Maximum horizontal shift for data augmentation.
The horizontal shift will be between -EClassificationDataset::MaxHorizontalShift and +EClassificationDataset::MaxHorizontalShift.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetMaxHorizontalShift() const
void SetMaxHorizontalShift(int maxShift)
```

EClassificationDataset::GetMaxHueOffset

EClassificationDataset::SetMaxHueOffset

Maximum absolute hue offset. Its value must be between 0 and 180 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetMaxHueOffset() const
void SetMaxHueOffset(float hueOffset)
```

Remarks

The hue is represented as an angle between 0 and 360 degrees. The hue transformation is performed by rotating the hue of each pixel by a value between -EClassificationDataset::MaxHueOffset and +EClassificationDataset::MaxHueOffset. This transformation only works for color images.

EClassificationDataset::GetMaxNumObjectPerImage

Maximum number of objects for an image in the dataset. If no images has object labeling (see EClassificationDataset::HasObjectLabeling), the method returns -1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetMaxNumObjectPerImage() const
```

EClassificationDataset::GetMaxRotationAngle

EClassificationDataset::SetMaxRotationAngle

Maximum rotation angle for data augmentation.
The rotation angle will be between -EClassificationDataset::MaxRotationAngle and +EClassificationDataset::MaxRotationAngle.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxRotationAngle() const

void SetMaxRotationAngle(float maxAngle)
```

EClassificationDataset::GetMaxSaturationGain

EClassificationDataset::SetMaxSaturationGain

Maximum saturation gain. Its value must be over or equal to EClassificationDataset::MinSaturationGain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxSaturationGain() const

void SetMaxSaturationGain(float saturationGain)
```

Remarks

The saturation transformation is performed by multiplying the saturation of each pixel by a value between EClassificationDataset::MinSaturationGain and EClassificationDataset::MaxSaturationGain.

EClassificationDataset::GetMaxScale

EClassificationDataset::SetMaxScale

Maximum scaling allowed for data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxScale() const

void SetMaxScale(float maxScale)
```

EClassificationDataset::GetMaxVerticalShear

EClassificationDataset::SetMaxVerticalShear

Maximum absolute vertical shear.
It is represented as an angle from the horizontal direction. Its value must be between 0 and 90 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxVerticalShear() const
void SetMaxVerticalShear(float vShear)
```

EClassificationDataset::GetMaxVerticalShift

EClassificationDataset::SetMaxVerticalShift

Maximum vertical shift for data augmentation.
The vertical shift will be between -EClassificationDataset::MaxHorizontalShift and +EClassificationDataset::MaxHorizontalShift.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetMaxVerticalShift() const
void SetMaxVerticalShift(int maxShift)
```

EClassificationDataset::GetMinContrastGain

EClassificationDataset::SetMinContrastGain

Minimum contrast gain. Its value must be strictly positive and below EClassificationDataset::MaxContrastGain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinContrastGain() const
void SetMinContrastGain(float val)
```

Remarks

Contrast transformation is performed by multiplying each mean-centered pixel value by a gain value taken between EClassificationDataset::MinContrastGain and EClassificationDataset::MaxContrastGain. A contrast transformation does not change the overall brightness of an image.

EClassificationDataset::GetMinGamma

EClassificationDataset::SetMinGamma

Minimum gamma for gamma correction. Its value must be strictly positive and below EClassificationDataset::MaxGamma.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinGamma() const

void SetMinGamma(float gamma)
```

Remarks

Gamma correction transformation is performed by raising each normalized pixel value to the power of gamma with gamma taken between EClassificationDataset::MinGamma and EClassificationDataset::MaxGamma.

EClassificationDataset::GetMinSaturationGain

EClassificationDataset::SetMinSaturationGain

Minimum saturation gain. Its value must be strictly positive.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinSaturationGain() const

void SetMinSaturationGain(float saturationGain)
```

Remarks

The saturation transformation is performed by multiplying the saturation of each pixel by a value between EClassificationDataset::MinSaturationGain and EClassificationDataset::MaxSaturationGain.

EClassificationDataset::GetMinScale

EClassificationDataset::SetMinScale

Minimum scaling allowed for data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinScale() const
void SetMinScale(float minScale)
```

EClassificationDataset::GetNumImageFiles

Number of different image files contained in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImageFiles() const
```

Remarks

The number of image files can be different than the number of images of the dataset. An image file can correspond to several dataset images (with different ROI).

EClassificationDataset::GetNumImages

Number of images in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImages() const
```

EClassificationDataset::GetNumImagesWithForegroundSegments

Number of images that have a ground truth segmentation that has foreground segments and is this not entirely composed of background pixels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImagesWithForegroundSegments() const
```

EClassificationDataset::GetNumImagesWithObjectLabeling

Number of images in the dataset that are labelled for object detection.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImagesWithObjectLabeling() const
```

EClassificationDataset::GetNumImagesWithObjects

Number of images in the dataset that are labelled for object detection and have 1 or more objects.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImagesWithObjects() const
```

EClassificationDataset::GetNumImagesWithoutForegroundSegments

Number of images that have a ground truth segmentation that has no foreground segments and is thus entirely composed of background pixels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImagesWithoutForegroundSegments() const
```

EClassificationDataset::GetNumImagesWithoutObjectLabeling

Number of images in the dataset that are not labelled for object detection.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImagesWithoutObjectLabeling() const
```

EClassificationDataset::GetNumImagesWithoutObjects

Number of images in the dataset that are labelled for object detection but have been associated with no object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImagesWithoutObjects() const
```

EClassificationDataset::GetNumImagesWithoutSegmentation

Number of images that doesn't have a ground truth segmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImagesWithoutSegmentation() const
```

EClassificationDataset::GetNumImagesWithSegmentation

Number of images that have a ground truth segmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImagesWithSegmentation() const
```

EClassificationDataset::GetNumLabeledImages

Number of images that have a classification label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumLabeledImages() const
```

EClassificationDataset::GetNumLabels

Number of labels in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumLabels() const
```

EClassificationDataset::GetNumObjectLabels

Number of object labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumObjectLabels() const
```

EClassificationDataset::GetNumSegmentationLabels

Number of segmentation labels. A dataset has always at least one segmentation label: "background".

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumSegmentationLabels() const
```

EClassificationDataset::GetNumUnlabeledImages

Number of images that doesn't have any classification label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumUnlabeledImages() const
```

EClassificationDataset::GetObjectSize

EClassificationDataset::SetObjectSize

Object size for EasyLocate Interest point.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetObjectSize() const

void SetObjectSize(int objectSize)
```

EClassificationDataset::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EClassificationDataset& operator=(
 const EClassificationDataset& other
 )
```

Parameters

other

Reference to the EClassificationDataset object used for the assignment

EClassificationDataset::RemoveImage

Removes the image at the given index. All annotations (label, segmentation, objects) will be lost.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveImage(
 int imgId
 )
```

Parameters

imgId

Index of the image to remove

EClassificationDataset::RemoveImageObject

Removes an object from an image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveImageObject(
 int imageIndex,
 int objectIndex
 )
```

Parameters

imageIndex

Index of the image

objectIndex

Index of the object between 0 and EClassificationDataset::GetImageNumObjects

EClassificationDataset::RemoveLabel

Removes a label from the dataset.
All the images associated with this label will be set to unlabeled (see EClassificationDataset::HasLabel).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveLabel(
 const std::string& label
 )

void RemoveLabel(
 int labelId
 )
```

Parameters

label
Name of the label to remove

labelId
Index of the label to remove

EClassificationDataset::RemoveObjectLabel

Removes an object label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveObjectLabel(
 int labelIndex
 )

void RemoveObjectLabel(
 const std::string& label
 )
```

Parameters

labelIndex
Index of the object label to remove

label
Label to remove

EClassificationDataset::RemoveSegmentationLabel

Remove a segmentation label.
This method sets all the pixels in the dataset assigned to this label to the background label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void RemoveSegmentationLabel(
 const std::string& label
 )
void RemoveSegmentationLabel(
 int index
 )
```

Parameters

label

Name of the segmentation label to remove

index

Index of the segmentation label to remove

EClassificationDataset::ResetImageObjectLabeling

Resets the object labeling for the specified image. This sets the image as having been labelled for object detection with no object present in the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void ResetImageObjectLabeling(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image

EClassificationDataset::ResetSegmentation

Resets the segmentation of an image by setting all pixels to background.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void ResetSegmentation(
 int imageIndex
 )
```

Parameters

imageIndex

Image index

EClassificationDataset::GetSaltAndPepperNoiseMaximumDensity

EClassificationDataset::SetSaltAndPepperNoiseMaximumDensity

The maximum density of the salt and pepper noise.
The salt and pepper noise sets the value of a number of randomly selected (between EClassificationDataset::SaltAndPepperNoiseMinimumDensity and EClassificationDataset::SaltAndPepperNoiseMaximumDensity) pixels to its minimum or maximum value.
Its value must be betwteen EClassificationDataset::SaltAndPepperNoiseMinimumDensity and 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSaltAndPepperNoiseMaximumDensity() const

void SetSaltAndPepperNoiseMaximumDensity(float saltAndPepperMaximumDensity)
```

Remarks

This noise is computed after all the other noises.

EClassificationDataset::GetSaltAndPepperNoiseMinimumDensity

EClassificationDataset::SetSaltAndPepperNoiseMinimumDensity

The minimum density of the salt and pepper noise.
The salt and pepper noise sets the value of a number of randomly selected (between EClassificationDataset::SaltAndPepperNoiseMinimumDensity and EClassificationDataset::SaltAndPepperNoiseMaximumDensity) pixels to its minimum or maximum value.
Its value must be betwteen 0 and EClassificationDataset::SaltAndPepperNoiseMaximumDensity.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSaltAndPepperNoiseMinimumDensity() const

void SetSaltAndPepperNoiseMinimumDensity(float saltAndPepperMinimumDensity)
```

Remarks

This noise is computed after all the other noises.

EClassificationDataset::GetSameLabelMaxOverlap

Maximum overlap between objects with the same label in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSameLabelMaxOverlap() const
```

EClassificationDataset::Save

Saves a classification dataset to disk, containing the file paths to the images in the dataset and their associated label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

A string containing the full path to the dataset file.

serializer

The serializer.

Remarks

This method only save the image that were given to this EClassificationDataset instance as EBaseROI pointers.
To obtain a portable EClassificationDataset file, please use EClassificationDataset::Export.

EClassificationDataset::SetImageLabel

Sets the label of images in the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetImageLabel(
 int index,
 const std::string& label
 )

void SetImageLabel(
 const std::string& filter,
 const std::string& label
 )
```

Parameters

index

The index of the image for which to set the label.

label

The label

filter

A glob filter

Remarks

The filter is a glob pattern. This means the wildcard characters "*" and "?" correspond to "zero or more character" and "a single character" respectively. For example, the filter "*_good_*.png" will match any filename that contains the string "_good_" and has a png extension.

EClassificationDataset::SetImageObject

Sets the specified object for the given image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetImageObject(
 int imageIndex,
 int objectIndex,
 const ELocatorObject& obj
 )

void SetImageObject(
 int imageIndex,
 int objectIndex,
 const std::string& label
 )

void SetImageObject(
 int imageIndex,
 int objectIndex,
 const ERectangleRegion& region
 )
```

Parameters

imageIndex
Index of the image

objectIndex
Index of the object between 0 and EClassificationDataset::GetImageNumObjects

obj
Object

label
New label for the object

region
New region for the object

Remarks

If the label of the object does not exist in the object labels of the dataset, the label will be added to the object labels of the dataset.

EClassificationDataset::SetLabel

Sets the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1). This operation does not add a new label to the dataset but simply renames an existing label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetLabel(
 int index,
 const std::string& label
 )
```

Parameters

index
Label index

label
Replacement label

EClassificationDataset::SetLabelWeight

Sets the weight associated to the i-th label of the dataset (starting from 0 to EClassificationDataset::NumLabels - 1). This operation does not add a new label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetLabelWeight(
 int index,
 const float& weight
 )
```

Parameters

index

Label index

weight

-

EClassificationDataset::SetMask

Sets the mask region/don't care area for the given image. The mask is defined with respect to the region of interest of the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetMask(
 int imageIndex,
 const ERegion& mask
 )
```

Parameters

imageIndex

Index of the image for which to get the mask region

mask

Mask to set on the image

EClassificationDataset::SetObjectLabel

Sets an object label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetObjectLabel(
 int labelIndex,
 const std::string& newLabel
 )

void SetObjectLabel(
 const std::string& oldLabel,
 const std::string& newLabel
 )
```

Parameters

labelIndex
Index of the object label

newLabel
New label to be set

oldLabel
Old label to change

EClassificationDataset::SetObjectLabelWeight

Sets the weight of an object label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetObjectLabelWeight(
 int labelIndex,
 float weight
 )
void SetObjectLabelWeight(
 const std::string& label,
 float weight
 )
```

Parameters

labelIndex
Index of the object label

weight
New weight

label
Label

EClassificationDataset::SetRegionOfInterest

Sets the region of interest for the specified image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetRegionOfInterest(
 int imageIndex,
 int xOrg,
 int yOrg,
 int width,
 int height
 )
```

Parameters

imageIndex
Index of the image.
xOrg
ROI origin abscissa
yOrg
ROI origin ordinate
width
ROI witdh
height
ROI height

EClassificationDataset::SetSegmentationLabel

Sets the segmentation labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetSegmentationLabel(
 int index,
 const std::string& label
 )
```

Parameters

index
Index of the segmentation label
label
String representing the segmentation label

Remarks

The segmentation label index 0 is reserved and corresponds to the "background" label. This segmentation label can't be changed.

EClassificationDataset::SetSegmentationLabelWeight

Sets the segmentation label weight.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetSegmentationLabelWeight(
 int index,
 float weight
 )
```

```
void SetSegmentationLabelWeight(
 const std::string& label,
 float weight
 )
```

Parameters

index
Index of the segmentation label

weight
Weight of the segmentation label

label
Segmentation label

EClassificationDataset::SetSegmentationMap

Sets the segmentation map of an image. The segmentation map is a 16-bit EROIBW16 image where the value of each pixel is equal to the corresponding segmentation label index (see EClassificationDataset::GetSegmentationLabel).
If an image has no segmentation (EClassificationDataset::HasSegmentation), the getter will throw an excpetion.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetSegmentationMap(
 int imageIndex,
 const EROIBW16& segmentationMap
 )
```

Parameters

imageIndex
Image index

segmentationMap
Segmentation map

EClassificationDataset::GetSpeckleNoiseMaximumStandardDeviation

EClassificationDataset::SetSpeckleNoiseMaximumStandardDeviation

The speckle noise maximum standard deviation.
The speckle noise is a multiplicative noise sampled from a Gamma distribution with a mean of 1 and with its standard deviation between EClassificationDataset::SpeckleNoiseMinimumStandardDeviation and EClassificationDataset::SpeckleNoiseMaximumStandardDeviation.
Its value must be strictly higher than EClassificationDataset::SpeckleNoiseMinimumStandardDeviation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetSpeckleNoiseMaximumStandardDeviation() const
void SetSpeckleNoiseMaximumStandardDeviation(float speckleMaximumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EClassificationDataset::GetSpeckleNoiseMinimumStandardDeviation

EClassificationDataset::SetSpeckleNoiseMinimumStandardDeviation

The speckle noise minimum standard deviation.
The speckle noise is a multiplicative noise sampled from a Gamma distribution with a mean of 1 and with its standard deviation between EClassificationDataset::SpeckleNoiseMinimumStandardDeviation and EClassificationDataset::SpeckleNoiseMaximumStandardDeviation.
Its value must be strictly positive and lower than EClassificationDataset::SpeckleNoiseMaximumStandardDeviation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetSpeckleNoiseMinimumStandardDeviation() const
void SetSpeckleNoiseMinimumStandardDeviation(float speckleMinimumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EClassificationDataset::SplitDataset

Splits the dataset in two parts to be used for training and validation respectively.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SplitDataset(
 EClassificationDataset& d1,
 EClassificationDataset& d2,
 float proportion,
 bool random
 )
```

Parameters

d1

First part of the dataset

d2

Second part of the dataset

proportion

Proportion of image of each class to put into the first part. The remaining images are put in d2

random

Randomly sample the images.

EClassificationDataset::SplitDatasetForLocator

Splits the dataset in two parts for training and validation of a ELocator tool. Images without object labeling are excluded from the split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SplitDatasetForLocator(
 EClassificationDataset& d1,
 EClassificationDataset& d2,
 float proportion,
 bool random
 )
```

Parameters

d1

First part of the dataset

d2

Second part of the dataset

proportion

Proportion of image of each class to put into the first part. The remaining images are put in d2

random

Randomly sample the images.

Remarks

The method ensures that all the object labels are represented in d1. Thus, even when the parameter random is set to false, the images in d1 and d2 can be ordered differently than they were in the original dataset.

EClassificationDataset::SplitDatasetForSegmentation

Splits the dataset in two parts for a supervised segmenter. The two parts are to be used for training and validation respectively.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SplitDatasetForSegmentation(
 EClassificationDataset& d1,
 EClassificationDataset& d2,
 float proportion,
 bool random
 )
```

Parameters

d1

First part of the dataset

d2

Second part of the dataset

proportion

Proportion of image of each class to put into the first part. The remaining images are put in d2

random

Randomly sample the images.

Remarks

The method ensures that all the segmentation labels are represented in d1. Thus, even when the parameter random is set to false, the images in d1 and d2 can be ordered differently than they were in the original dataset.

EClassificationDataset::UnsetImageLabel

Unset the label of the given image of the dataset. After this operation, the given image will have no classification labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void UnsetImageLabel(
 int index
 )
```

Parameters

index

Index of the image for which to unset the label

EClassificationDataset::UnsetImageObjectLabeling

Unsets the object labeling for the specified image. This sets the image as not having been labelled for object detection. This image won't be use for training with a ELocator tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UnsetImageObjectLabeling(
 int imageIndex
 )
```

Parameters

imageIndex

Index of the image

EClassificationDataset::UnsetSegmentation

Unsets the segmentation of an image. After this operation, the image will have no segmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UnsetSegmentation(
 int imageIndex
 )
```

Parameters

imageIndex

Image index

EClassificationDataset::GetWidth

Width of the region of interest of the first image added to the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetWidth() const
```

4.56. EClassificationMetrics Class

Collection of metrics used to evaluate the state of an EClassifier.
A metric is a value summarizing a collection of classification results (EClassificationResult).
New results can be added to the object individually with EClassificationMetrics::AddResult or collectively with EClassificationMetrics::AddMetrics.
EClassificationMetrics contains the following metrics:
- the accuracy (see EClassificationMetrics::Accuracy)
- the error (see EClassificationMetrics::Error)

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

AddMetrics	Adds the other metrics to the current metrics of this object.
AddResult	Adds the given result with the corresponding ground truth label to the metrics.
CanComputeWeightedError	Whether the object can be used to get weighted, balanced, and label errors.
EClassificationMetrics	Constructs an EClassificationMetrics object.
GetAccuracy	The accuracy of the classifier.
GetBalancedAccuracy	The balanced accuracy.
GetBalancedError	The balanced error.
GetConfusion	Confusion value of one label with another. The confusion value of a label with another is the number of images belonging to this label that are classified as belonging to the other label.
GetError	The error of the classifier.
GetLabelAccuracy	The accuracy of the classifier for a given label.
GetLabelError	The error of the classifier for a given label.
GetWeightedAccuracy	The label weighted accuracy.
GetWeightedError	The label weighted error.
IsValid	Indicates whether the object contains at least one classification result.
Load	Loads a classification metric. The given ESerializer must have been created for reading.
operator=	Assignment operator
Save	Saves a classification metric. The given ESerializer must have been created for writing.

EClassificationMetrics::GetAccuracy

The accuracy of the classifier.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAccuracy() const
```

Remarks

The accuracy is the number of images that were correctly classified (also called the true positives) over the total number of images that was used to evaluate the classifier.

EClassificationMetrics::AddMetrics

Adds the other metrics to the current metrics of this object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddMetrics(
 const EClassificationMetrics& other
 )
```

Parameters

other

Classification metrics

EClassificationMetrics::AddResult

Adds the given result with the corresponding ground truth label to the metrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddResult(
 const EClassificationResult& result,
 const std::string& groundtruthLabel
 )
```

Parameters

result

A reference to an EClassificationResult object.

groundtruthLabel

The ground truth label corresponding to the result

EClassificationMetrics::GetBalancedAccuracy

The balanced accuracy.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetBalancedAccuracy() const
```

Remarks

The balanced accuracy is the label weighted accuracy with the same weight for each labels.

EClassificationMetrics::GetBalancedError

The balanced error.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetBalancedError() const
```

Remarks

The balanced error is the label weighted error with the same weight for each labels.
If EClassificationMetrics::CanComputeWeightedError is false, this method is an alias for EClassificationMetrics::Error.

EClassificationMetrics::CanComputeWeightedError

Whether the object can be used to get weighted, balanced, and label errors.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool CanComputeWeightedError(
 )
```

EClassificationMetrics::EClassificationMetrics

Constructs an EClassificationMetrics object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EClassificationMetrics(
 )

void EClassificationMetrics(
 const EClassificationMetrics& other
 )
```

Parameters

other

Reference to the EClassificationMetrics object that should be copied

EClassificationMetrics::GetError

The error of the classifier.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetError() const
```

Remarks

The error, which is also called the loss, is the quantity that is minimized during the training of the deep neural network. For classification, the error is the crossentropy.

EClassificationMetrics::GetConfusion

Confusion value of one label with another.
The confusion value of a label with another is the number of images belonging to this label that are classified as belonging to the other label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetConfusion(
 const std::string& trueClass,
 const std::string& predictedClass
 )
```

Parameters

trueClass

The label for which to obtain the confusion value

predictedClass

The label with which there is a confusion

EClassificationMetrics::GetLabelAccuracy

The accuracy of the classifier for a given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelAccuracy(
 const std::string& label
 )
```

Parameters

label

-

Remarks

The label accuracy is the number of images of a given label that were correctly classified (also called the true positives) over the total number of images of that label that was used to evaluate the classifier.
If there is no results for the given label, the method will throw an exception.

EClassificationMetrics::GetLabelError

The error of the classifier for a given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetLabelError(
 const std::string& label
 )
```

Parameters

label

The label from which to get the error.

Remarks

The label error corresponds to the error only for images of the given label.
If there is no results for the given label, the method will throw an exception.
If EClassificationMetrics::CanComputeWeightedError is false, this method is an alias for EClassificationMetrics::Error.

EClassificationMetrics::GetWeightedAccuracy

The label weighted accuracy.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetWeightedAccuracy(
 const std::vector<float>& weights
 )
float GetWeightedAccuracy(
 const EClassificationDataset& dataset
 )
```

Parameters

weights

Array of label weights for the labels of the classifier (see EClassifier)

dataset

Dataset from which to use the label weights

Remarks

The weighted accuracy is the weighted average of the label accuracies (see EClassificationMetrics::GetLabelAccuracy). If there is no results for a given label, it will be ignored in the weighted accuracy.

EClassificationMetrics::GetWeightedError

The label weighted error.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetWeightedError(
 const std::vector<float>& weights
 )
float GetWeightedError(
 const EClassificationDataset& dataset
 )
```

Parameters

weights

Array of label weights for the labels of the classifier (see EClassifier)

dataset

Dataset from which to use the label weights

Remarks

The weighted error is the weighted average of the label errors (see EClassificationMetrics::GetLabelError).
If there isn't any result for a given label, it will be ignored in the weighted error.
If EClassificationMetrics::CanComputeWeightedError is false, this method is an alias for EClassificationMetrics::Error.

EClassificationMetrics::IsValid

Indicates whether the object contains at least one classification result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsValid(
 )
```

EClassificationMetrics::Load

Loads a classification metric. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EClassificationMetrics::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EClassificationMetrics& operator=(
 const EClassificationMetrics& other
 )
```

Parameters

other

Reference to the EClassificationMetrics object used for the assignment

EClassificationMetrics::Save

Saves a classification metric. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.57. EClassificationResult Class

An EClassificationResult object represents the result of a classification.
The most probable label and its probability are accessible through the methods EClassificationResult::BestLabel and EClassificationResult::BestProbability.
The probability and ranking of all labels are accessible through the EClassificationResult and EClassificationResult methods.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

DrawHeatmap	-
EClassificationResult	Constructs a non-valid EClassificationResult.
GetBestLabel	Gets the most probable label.
GetBestLabelId	Gets the most probable label id.
GetBestProbability	Gets the probability associated with the most probable label
GetColorizedHeatmap	Colorized heatmap with no transparency.
GetColorizedHeatmap WithTransparency	Colorized heatmap with transparency. When a minimum and maximum alpha values are given, the transparency value for a pixel will depend on the value of the heatmap at that pixel. The range of heatmap values [0, 255] will be mapped to the range [minAlpha, maxAlpha].
GetGroundtruthLabel	Ground truth label.
GetHeatmap	Heatmap.
GetLabel	Classification label. The labels are indexed from 0 to EClassificationResult::NumLabels.
GetLabelColor	Color of a label.
GetNumLabels	Number of labels for which we have a probability or a ranking.
GetProbability	Gets the probability corresponding to the given label.
GetRanking	Gets the ranking corresponding to the given label. The ranking goes from 1 (most probable label) to EClassifier (least probable label).
HasGroundtruth	Whether the result has a ground truth.
HasHeatmap	Whether the result contains an heatmap.
IsValid	Indicates whether the result was produced by EClassifier. A default contructed EClassificationResult is not valid.

operator= Assignment operator

EClassificationResult::GetBestLabel

Gets the most probable label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetBestLabel() const
```

EClassificationResult::GetBestLabelId

Gets the most probable label id.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetBestLabelId() const
```

EClassificationResult::GetBestProbability

Gets the probability associated with the most probable label

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetBestProbability() const
```

EClassificationResult::DrawHeatmap

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void DrawHeatmap(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawHeatmap(
 EDrawAdapter* graphicsContext,
 float minAlpha,
 float maxAlpha,
 Euresys::Open_eVision::EasyDeepLearning::EHeatmapColormap colormap,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawHeatmap(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawHeatmap(
 HDC graphicsContext,
 float minAlpha,
 float maxAlpha,
 Euresys::Open_eVision::EasyDeepLearning::EHeatmapColormap colormap,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

-

zoomX

-

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

minAlpha

Minimum transparency value.

maxAlpha

Maximum transparency value.

colormap

Color map

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EClassificationResult::EClassificationResult

Constructs a non-valid EClassificationResult.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EClassificationResult(
 )
void EClassificationResult(
 const EClassificationResult& other
 )
```

Parameters

other

Reference to the EClassificationResult object that should be copied

EClassificationResult::GetColorizedHeatmap

Colorized heatmap with no transparency.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EImageC24 GetColorizedHeatmap(
 Euresys::Open_eVision::EasyDeepLearning::EHeatmapColormap colormap
 )
```

Parameters

colormap

Color map

EClassificationResult::GetColorizedHeatmapWithTransparency

Colorized heatmap with transparency.
When a minimum and maximum alpha values are given, the transparency value for a pixel will depend on the value of the heatmap at that pixel. The range of heatmap values [0, 255] will be mapped to the range [minAlpha, maxAlpha].

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageC24A GetColorizedHeatmapWithTransparency(
 float alpha,
 Euresys::Open_eVision::EasyDeepLearning::EHeatmapColormap colormap
 )
EImageC24A GetColorizedHeatmapWithTransparency(
 float minAlpha,
 float maxAlpha,
 Euresys::Open_eVision::EasyDeepLearning::EHeatmapColormap colormap
 )
```

Parameters

alpha
Transparency to apply everywhere

colormap
Color map

minAlpha
Minimum transparency value.

maxAlpha
Maximum transparency value.

EClassificationResult::GetLabel

Classification label. The labels are indexed from 0 to EClassificationResult::NumLabels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetLabel(
 int labelId
 )
```

Parameters

labelId
Id of the label

EClassificationResult::GetLabelColor

Color of a label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ERGBColor GetLabelColor(
 int i
 )
ERGBColor GetLabelColor(
 const std::string& label
 )
```

Parameters

i

Index of the label for which to get the color (between 0 and EClassificationResult::NumLabels - 1)

label

Label for which to get the color

Remarks

The label color is controlled at the tool level. To change a color in a result, change the label color in the tool.

EClassificationResult::GetProbability

Gets the probability corresponding to the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetProbability(
 const std::string& label
 )
```

Parameters

label

The label

EClassificationResult::GetRanking

Gets the ranking corresponding to the given label. The ranking goes from 1 (most probable label) to EClassifier (least probable label).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetRanking(
 const std::string& label
 )
```

Parameters

label

The label

EClassificationResult::GetGroundtruthLabel

Ground truth label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetGroundtruthLabel() const
```

EClassificationResult::HasGroundtruth

Whether the result has a ground truth.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasGroundtruth(
 )
```

EClassificationResult::HasHeatmap

Whether the result contains an heatmap.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasHeatmap(
 )
```

EClassificationResult::GetHeatmap

Heatmap.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageBW8 GetHeatmap() const
```

EClassificationResult::IsValid

Indicates whether the result was produced by EClassifier.
A default contructed EClassificationResult is not valid.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsValid(
 )
```

EClassificationResult::GetNumLabels

Number of labels for which we have a probability or a ranking.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumLabels() const
```

EClassificationResult::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EClassificationResult& operator=(
 const EClassificationResult& other
 )
```

Parameters

other

Reference to the EClassificationResult object used for the assignment

4.58. EClassifier Class

EClassifier allows to train a classifier using an EClassificationDataset object and classify new images.

As required by Deep Learning techniques, the input image of EClassifier must be of the same format (width, height, number of channels). By default, this format will be the one of the first image added to the dataset used for training unless its width and height is smaller than the minimum width and height supported by the classifier (See EClassifier::MinimumWidth and EClassifier::MinimumHeight). In this case, the input resolution will be the minimum resolution supported by the classifier. The format can also be specified by the EClassifier::Width, EClassifier::Height and EClassifier::Channels. methods.

By default, images that don't satisfy the image format of the classifier are automatically reformatted. This behavior can be controlled through the EClassifier::EnableAutomaticImageReformat method. When the automatic image reformatting is disabled, training or classifying an image that doesn't satisfy the input image format will result in an exception.

Once trained, the input image format cannot be changed.

Base Class:EDeepLearningTool

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Classify	Classifies images and returns the complete results as an EClassificationResult object. The method throws an exception if the input image does not fulfill the input specification.
EClassifier	Constructs a EClassifier object.
Evaluate	Evaluates the EClassifier using the given EClassificationDataset.
GetAvailableModelTypes	Get the i-th available models to be used with EClassifier::ModelType.
GetCapacity	DEPRECATED. Use EClassifier::ModelType instead. Capacity of classifier to use. Be aware that changing the classifier capacity will delete the effect of any previous training. The capacity will be translated to the corresponding type (Small, Normal, or Large).
GetChannels	Number of channels for input images of the classifier. The number of channels can be either 1 (monochrome image) or 3 (RGB image). By default, this value will be set from the format of the first image added to the training dataset.
GetColorizedHeatMap	-

GetColorizedHeatmapWithTransparency	Colorized heatmap with transparency. When the minimum and maximum alpha values are different from each other, the transparency value for a pixel will depend on the value of the heatmap at that pixel. The range of heatmap values [0, 255] will be mapped to the range [minAlpha, maxAlpha].
GetComputeHeatmapWithResult	Whether to always compute the heatmap along the result when applying the tool. If true, the heatmap is available through EClassificationResult::Heatmap.
GetEnableAutomaticImageReformat	Enable automatic image reformat (true by default).
GetEnableHistogramEqualization	Enable histogram equalization of all images passing through the classifier. (false by default)
GetHeatMap	Gets a heatmap associated with the given label. A heatmap is an image that indicates which pixels in the original image best explain the given label.
GetHeight	Height for input images of the classifier. By default, this value will be set from the format of the first image of the dataset used for training.
GetMinimumHeight	Minimum height for input images of the classifier. This value is equal to 1 for standard networks.
GetMinimumWidth	Minimum width for input images of the classifier. This value is equal to 1 for standard networks.
GetModelType	Model type for training. Default: "Normal". The list of available models is available through EClassifier.
GetNumAvailableModelTypes	Number of available models for training an EasyClassify tool.
GetToolType	Type of the deep learning tool.
GetTrainingMetrics	Gets the metrics obtained with the training dataset at the given iteration. The iterations are indexed between 0 and EDeepLearningTool::NumTrainedIterations - 1.
GetUsePretrainedModel	Whether to use a pretrained model when training. Default: true if the pretrained model are found on the disk.
GetValidationMetrics	Gets the metrics obtained with the validation dataset at the given iteration. The iterations are indexed between 0 and EDeepLearningTool::NumTrainedIterations - 1.
GetWidth	Width for input images of the classifier. By default, this value will be set from the format of the first image of the dataset used for training.
HasPretrainedModel	Whether a pretrained model for the current configuration can be found.
LoadAsPretrained	Loads the classifier and use it as a pretrained model.
operator=	Assignment operator
SerializeSettings	Serializes the classifier settings.

SetCapacity	DEPRECATED. Use EClassifier::ModelType instead. Capacity of classifier to use. Be aware that changing the classifier capacity will delete the effect of any previous training. The capacity will be translated to the corresponding type (Small, Normal, or Large).
SetChannels	Number of channels for input images of the classifier. The number of channels can be either 1 (monochrome image) or 3 (RGB image). By default, this value will be set from the format of the first image added to the training dataset.
SetComputeHeatmapWithResult	Whether to always compute the heatmap along the result when applying the tool. If true, the heatmap is available through EClassificationResult::Heatmap.
SetEnableAutomaticImageReformat	Enable automatic image reformat (true by default).
SetEnableHistogramEqualization	Enable histogram equalization of all images passing through the classifier. (false by default)
SetHeight	Height for input images of the classifier. By default, this value will be set from the format of the first image of the dataset used for training.
SetModelType	Model type for training. Default: "Normal". The list of available models is available through EClassifier.
SetUsePretrainedModel	Whether to use a pretrained model when training. Default: true if the pretrained model are found on the disk.
SetWidth	Width for input images of the classifier. By default, this value will be set from the format of the first image of the dataset used for training.

EClassifier::GetCapacity

EClassifier::SetCapacity

This property is deprecated.

DEPRECATED. Use EClassifier::ModelType instead. Capacity of classifier to use. Be aware that changing the classifier capacity will delete the effect of any previous training. The capacity will be translated to the corresponding type (Small, Normal, or Large).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EClassifierCapacity GetCapacity() const

void SetCapacity(Euresys::Open_eVision::EasyDeepLearning::EClassifierCapacity capacity)
```

EClassifier::GetChannels

EClassifier::SetChannels

Number of channels for input images of the classifier. The number of channels can be either 1 (monochrome image) or 3 (RGB image). By default, this value will be set from the format of the first image added to the training dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetChannels() const

void SetChannels(OEV_UINT32 channel)
```

Remarks

If the classifier is not trained or the value was not explicitely set, its value will be 0.

EClassifier::Classify

Classifies images and returns the complete results as an EClassificationResult object.
The method throws an exception if the input image does not fulfill the input specification.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EClassificationResult Classify(
 const EBaseROI& img
 )

EClassificationResult Classify(
 const EBaseROI& img,
 ERegion& mask
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<const Euresys::Open_eVision::EBaseROI*>& imgList,
 ERegion& mask
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<const Euresys::Open_eVision::EBaseROI*>& imgList
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<Euresys::Open_eVision::EImageBW8>& imgList,
 ERegion& mask
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<Euresys::Open_eVision::EImageBW8>& imgList
 )
```

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<Euresys::Open_eVision::EImageBW16>& imgList,
 ERegion& mask
 )
std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<Euresys::Open_eVision::EImageBW16>& imgList
 )
std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<Euresys::Open_eVision::EImageC24>& imgList,
 ERegion& mask
 )
std::vector<Euresys::Open_eVision::EasyDeepLearning::EClassificationResult> Classify(
 std::vector<Euresys::Open_eVision::EImageC24>& imgList
 )
```

Parameters

img

Image to classify

mask

Mask of image to classify

imgList

Vector of images to classify

Remarks

Classifying a set of images is usually faster than classifying each image sequentially.
To maximize the classification speed on a GPU, EClassifier and the size of the set of input images must be equal to the value returned by EDeepLearningTool.

EClassifier::GetComputeHeatmapWithResult

EClassifier::SetComputeHeatmapWithResult

Whether to always compute the heatmap along the result when applying the tool. If true, the heatmap is available through EClassificationResult::Heatmap.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetComputeHeatmapWithResult() const
void SetComputeHeatmapWithResult(bool computeHeatmap)
```

EClassifier::EClassifier

Constructs a EClassifier object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EClassifier(
 )
void EClassifier(
 const EClassifier& other
 )
```

Parameters

other

Reference to the EClassifier object that should be copied

EClassifier::GetEnableAutomaticImageReformat

EClassifier::SetEnableAutomaticImageReformat

Enable automatic image reformat (true by default).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool GetEnableAutomaticImageReformat() const
void SetEnableAutomaticImageReformat(bool val)
```

EClassifier::GetEnableHistogramEqualization

EClassifier::SetEnableHistogramEqualization

Enable histogram equalization of all images passing through the classifier. (false by default)

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool GetEnableHistogramEqualization() const
void SetEnableHistogramEqualization(bool val)
```

EClassifier::Evaluate

Evaluates the EClassifier using the given EClassificationDataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EClassificationMetrics Evaluate(
 EClassificationDataset& dataset
 )
```

Parameters

dataset

EClassificationDataset with which to evaluate the classifier

Remarks

This method computes various metrics (see EClassificationMetrics on the given dataset. The method ignores the label weights and the data augmentation settings of the given dataset. However, the label weights can be taken into consideration in the returned EClassificationMetrics object.

EClassifier::GetAvailableModelTypes

Get the i-th available models to be used with EClassifier::ModelType.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetAvailableModelTypes(
 int i
 )
```

Parameters

i

Index of the model between 0 and EClassifier::NumAvailableModelTypes - 1.

EClassifier::GetColorizedHeatMap

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EImageC24 GetColorizedHeatMap(
 const EBaseROI& image,
 const std::string& label
 )
```

Parameters

image

-

label

-

EClassifier::GetColorizedHeatmapWithTransparency

Colorized heatmap with transparency.
When the minimum and maximum alpha values are different from each other, the transparency value for a pixel will depend on the value of the heatmap at that pixel. The range of heatmap values [0, 255] will be mapped to the range [minAlpha, maxAlpha].

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageC24A GetColorizedHeatmapWithTransparency(
 const EBaseROI& image,
 const std::string& label,
 float minAlpha,
 float maxAlpha,
 Euresys::Open_eVision::EasyDeepLearning::EHeatmapColormap colormap
 )
```

Parameters

image
A reference to the image we want to generate the heatmap from.

label
A reference to the string representing the label we want to explain for the given image.

minAlpha
Minimum transparency value.

maxAlpha
Maximum transparency value.

colormap
Color map

EClassifier::GetHeatMap

Gets a heatmap associated with the given label. A heatmap is an image that indicates which pixels in the original image best explain the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageBW8 GetHeatMap(
 const EBaseROI& image,
 const std::string& label
 )
```

Parameters

image

A reference to the image we want to generate the heatmap from.

label

A reference to the string representing the label we want to explain for the given image.

EClassifier::GetTrainingMetrics

Gets the metrics obtained with the training dataset at the given iteration.
The iterations are indexed between 0 and EDeepLearningTool::NumTrainedIterations - 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EClassificationMetrics GetTrainingMetrics(
 int id
 )
```

Parameters

id

The iteration index

EClassifier::GetValidationMetrics

Gets the metrics obtained with the validation dataset at the given iteration.
The iterations are indexed between 0 and EDeepLearningTool::NumTrainedIterations - 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EClassificationMetrics GetValidationMetrics(
 int id
 )
```

Parameters

id

The iteration index

EClassifier::HasPretrainedModel

Whether a pretrained model for the current configuration can be found.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasPretrainedModel(
 )
```

EClassifier::GetHeight

EClassifier::SetHeight

Height for input images of the classifier. By default, this value will be set from the format of the first image of the dataset used for training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetHeight() const

void SetHeight(OEV_UINT32 height)
```

Remarks

If the classifier is not trained or the value was not explicitely set, its value will be 0.

EClassifier::LoadAsPretrained

Loads the classifier and use it as a pretrained model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void LoadAsPretrained(
 const std::string& path
 )
```

Parameters

path

-

EClassifier::GetMinimumHeight

Minimum height for input images of the classifier. This value is equal to 1 for standard networks.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetMinimumHeight() const
```

EClassifier::GetMinimumWidth

Minimum width for input images of the classifier. This value is equal to 1 for standard networks.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetMinimumWidth() const
```

EClassifier::GetModelType

EClassifier::SetModelType

Model type for training. Default: "Normal".
The list of available models is available through EClassifier.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetModelType() const
void SetModelType(const std::string& name)
```

EClassifier::GetNumAvailableModelTypes

Number of available models for training an EasyClassify tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
static int GetNumAvailableModelTypes()
```

EClassifier::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EClassifier& operator=(
 const EClassifier& other
 )
```

Parameters

other

Reference to the EClassifier object used for the assignment

EClassifier::SerializeSettings

Serializes the classifier settings.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer

Pointer to ESerializer

EClassifier::GetToolType

Type of the deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetToolType() const
```

EClassifier::GetUsePretrainedModel

EClassifier::SetUsePretrainedModel

Whether to use a pretrained model when training.
Default: true if the pretrained model are found on the disk.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool GetUsePretrainedModel() const

void SetUsePretrainedModel(bool usePretrainedModel)
```

EClassifier::GetWidth

EClassifier::SetWidth

Width for input images of the classifier. By default, this value will be set from the format of the first image of the dataset used for training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetWidth() const
void SetWidth(OEV_UINT32 width)
```

Remarks

If the classifier is not trained or the value was not explicitely set, its value will be 0.

4.59. ECode Class

Represents a Code.

Namespace: Euresys::Open_eVision

Methods

DrawPosition	Draws the Position of the Code.
DrawPositionWithCurrentPen	Draws the Position of the Code using the pen currently set in the graphical context.
ECode	Creates an ECode object.
GetBarCode	-
GetCodeType	Code type.
GetDecodedString	Decoded code string.
GetMatrixCode	-
GetPosition	Code position.
GetQRCode	-
operator=	Assignment operator

ECode::GetBarCode

-

Namespace: Euresys::Open_eVision

[C++]

```
const EBarCode& GetBarCode() const
```

ECode::GetCodeType

Code type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ECodeType GetCodeType() const
```

ECode::GetDecodedString

Decoded code string.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetDecodedString() const
```

ECode::DrawPosition

Draws the Position of the Code.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPosition(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawPosition(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

Handle of the graphics context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECode::DrawPositionWithCurrentPen

This method is deprecated.

Draws the Position of the Code using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPositionWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECode::ECode

Creates an ECode object.

Namespace: Euresys::Open_eVision

[C++]

```
void ECode(
 )
void ECode(
 const ECode& other
 )
```

Parameters

other

Another ECode object to be copied in the new ECode object.

ECode::GetMatrixCode

-

Namespace: Euresys::Open_eVision

[C++]

```
const EMatrixCode& GetMatrixCode() const
```

ECode::operator=

Assignment operator

Namespace: Euresys::Open_eVision

[C++]

```
ECode& operator=(
 const ECode& other
 )
```

Parameters

other

ECode object to be copied.

ECode::GetPosition

Code position.

Namespace: Euresys::Open_eVision

```
[C++]
EQuadrangle GetPosition() const
```

ECode::GetQRCode

-

Namespace: Euresys::Open_eVision

```
[C++]
const EQRCode& GetQRCode() const
```

4.60. ECodedElement Class

This class encapsulates either an object or a hole in an object, in a coded image.

Remarks

This abstract class provides a large set of methods applicable to a particular coded element. The set includes methods to get the features of a coded element, to draw coded elements, and to render flexible masks.

Derived Class(es):EHoleEObject

Namespace: Euresys::Open_eVision

Methods

AsHole	Down-casts the coded element as a hole.
AsObject	Down-casts the coded element as an object.
ComputeConvexHull	Computes the convex hull of the coded element.
ComputeFeretBox	Computes the Feret box at a specific orientation.
ComputePixelGrayAverage	Computes the average gray-level value of the pixels of a given image over the coded element.
ComputePixelGrayDeviation	Computes the standard deviation of the gray-level values of a given image over the coded element.
ComputePixelGrayVariance	Computes the variance of the gray-level values of a given image over the coded element.

ComputePixelMax	Computes the maximum gray level of the pixels of a given image over the coded element.
ComputePixelMin	Computes the minimum gray level of the pixels of a given image over the coded element.
ComputeWeightedGravityCenter	Computes the gravity center of a given image over the coded element.
GetArea	Returns the number of pixels inside the coded element.
GetBottomLimit	Returns the highest (integer) Y-coordinate of all the pixels of the coded element.
GetBoundingBox	Returns the bounding box of the coded element (Feret box at orientation 0 degrees).
GetBoundingBoxCenter	Returns the coordinates of the center of the bounding box of the coded element.
GetBoundingBoxCenterX	Returns the abscissa of the center of the bounding box of the coded element.
GetBoundingBoxCenterY	Returns the ordinate of the center of the bounding box of the coded element.
GetBoundingBoxHeight	Returns the height of the bounding box (Feret diameter at 90 degrees).
GetBoundingBoxWidth	Returns the width of the bounding box (Feret diameter at 0 degrees).
GetCentralMoment	Computes the central, two-dimensional moment of order (p,q).
GetContour	Returns the coordinates of the starting point of the countour of the coded element.
GetContourPath	Returns the contour path. The contour paths is computed using EContourMode_ ClockwiseAlwaysClosed.
GetContourX	Returns the abscissa of the starting point of the countour of the coded element.
GetContourY	Returns the ordinate of the starting point of the countour of the coded element.
GetConvexHull	Gets the convex hull of the coded element.
GetEccentricity	Returns the eccentricity of the ellipse of inertia.
GetElementIndex	Returns the index of the coded element.
GetEllipseAngle	Returns the angle of the ellipse of inertia.
GetEllipseHeight	Returns the length of the short axis of the ellipse of inertia.
GetEllipseWidth	Returns the length of the long axis of the ellipse of inertia.
GetFeretBox22Box	Returns the Feret box at orientation 22.5 degrees.
GetFeretBox22Center	Returns the coordinates of the center of the Feret box oriented at 22.5 degrees.
GetFeretBox22CenterX	Returns the abscissa of the center of the Feret box oriented at 22.5 degrees.

GetFeretBox22CenterY	Returns the ordinate of the center of the Feret box oriented at 22.5 degrees.
GetFeretBox22Height	Returns the height of the Feret box oriented at 22.5 degrees (Feret diameter at 112.5 degrees).
GetFeretBox22Width	Returns the width of the Feret box oriented at 22.5 degrees (Feret diameter at 22.5 degrees).
GetFeretBox45Box	Returns the Feret box at orientation 45 degrees.
GetFeretBox45Center	Returns the coordinates of the center of the Feret box oriented at 45 degrees.
GetFeretBox45CenterX	Returns the abscissa of the center of the Feret box oriented at 45 degrees.
GetFeretBox45CenterY	Returns the ordinate of the center of the Feret box oriented at 45 degrees.
GetFeretBox45Height	Returns the height of the Feret box oriented at 45 degrees (Feret diameter at 135 degrees).
GetFeretBox45Width	Returns the width of the Feret box oriented at 45 degrees (Feret diameter at 45 degrees).
GetFeretBox68Box	Returns the Feret box at orientation 67.5 degrees.
GetFeretBox68Center	Returns the coordinates of the center of the Feret box oriented at 67.5 degrees.
GetFeretBox68CenterX	Returns the abscissa of the center of the Feret box oriented at 67.5 degrees.
GetFeretBox68CenterY	Returns the ordinate of the center of the Feret box oriented at 67.5 degrees.
GetFeretBox68Height	Returns the height of the Feret box oriented at 67.5 degrees (Feret diameter at 157.5 degrees).
GetFeretBox68Width	Returns the width of the Feret box oriented at 67.5 degrees (Feret diameter at 67.5 degrees).
GetGravityCenter	Returns the gravity center of the coded element.
GetGravityCenterX	Returns the abscissa of the gravity center of the coded element.
GetGravityCenterY	Returns the ordinate of the gravity center of the coded element.
GetLargestRun	Returns the length of the largest run inside the coded element.
GetLayerIndex	Returns the index of the layer in the coded image to which the coded element belongs.
GetLeftLimit	Returns the lowest (integer) X-coordinate of all the pixels of the coded element.
GetMinimumEnclosingRectangle	Returns the Minimum-Area Enclosing Rectangle.
GetMinimumEnclosingRectangleAngle	Returns the angle of the Minimum-Area Enclosing Rectangle.
GetMinimumEnclosingRectangleCenter	Returns the coordinates of the center of the Minimum-Area Enclosing Rectangle.

GetMinimumEnclosingRectangleCenterX	Returns the abscissa of the center of the Minimum-Area Enclosing Rectangle.
GetMinimumEnclosingRectangleCenterY	Returns the ordinate of the center of the Minimum-Area Enclosing Rectangle.
GetMinimumEnclosingRectangleHeight	Returns the height of the Minimum-Area Enclosing Rectangle.
GetMinimumEnclosingRectangleWidth	Returns the width of the Minimum-Area Enclosing Rectangle.
GetMoment	Computes the raw, two-dimensional moment of order (p,q).
GetNormalizedCentralMoment	Computes the scale-invariant, central, two-dimensional moment of order (p,q).
GetRightLimit	Returns the highest (integer) X-coordinate of all the pixels of the coded element.
GetRunCount	Returns the number of runs inside the coded element.
GetRunsIterator	Returns an iterator to the runs of the coded element.
GetSigmaX	Returns the centered moment of inertia around X (average squared X-deviation).
GetSigmaXX	Returns the centered cross moment of inertia (average X-deviation * Y-deviation).
GetSigmaXY	Returns the reduced, centered moment of inertia (around the principal inertia axis).
GetSigmaY	Returns the centered moment of inertia around Y (average squared Y-deviation).
GetSigmaYY	Returns the reduced, centered moment of inertia (around the secondary inertia axis).
GetTopLimit	Returns the lowest (integer) Y-coordinate of all the pixels of the coded element.
IsCodedElement	Tests whether the coded element is an object, a hole or a coded element.
IsHole	Tests whether the coded element is an object, a hole or a coded element.
IsObject	Tests whether the coded element is an object, a hole or a coded element.
operator==	-
RenderMask	Creates a Flexible Mask from the coded element.
ToRegion	Creates an ERegion from the ECodedElement. The ERegion represents all the pixels which are within the bounding box of the Pattern.

ECodedElement::GetArea

Returns the number of pixels inside the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetArea() const
```

Remarks

Equivalently, the area corresponds to the sum of the length of the runs of the coded element.

ECodedElement::AsHole

Down-casts the coded element as a hole.

Namespace: Euresys::Open_eVision

```
[C++]
EHole& AsHole(
 )
const EHole& AsHole(
 )
```

Remarks

This method throws an exception if the coded element is in fact an object.

ECodedElement::AsObject

Down-casts the coded element as an object.

Namespace: Euresys::Open_eVision

```
[C++]
EObject& AsObject(
 )
const EObject& AsObject(
 )
```

Remarks

This method throws an exception if the coded element is in fact a hole.

ECodedElement::GetBottomLimit

Returns the highest (integer) Y-coordinate of all the pixels of the coded element.

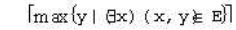
Namespace: Euresys::Open_eVision

[C++]

```
int GetBottomLimit() const
```

Remarks

For a coded element E, this value is defined as: <latex>\left\lceil \max \left\{ y \mid (\exists x) (x,y) \in E \right\} \right\rceil </latex>



ECodedElement::GetBoundingBox

Returns the bounding box of the coded element (Feret box at orientation 0 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
ERotatedBoundingBox GetBoundingBox() const
```

ECodedElement::GetBoundingBoxCenter

Returns the coordinates of the center of the bounding box of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetBoundingBoxCenter() const
```

ECodedElement::GetBoundingBoxCenterX

Returns the abscissa of the center of the bounding box of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
float GetBoundingBoxCenterX() const
```

ECodedElement::GetBoundingBoxCenterY

Returns the ordinate of the center of the bounding box of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
float GetBoundingBoxCenterY() const
```

ECodedElement::GetBoundingBoxHeight

Returns the height of the bounding box (Feret diameter at 90 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
float GetBoundingBoxHeight() const
```

ECodedElement::GetBoundingBoxWidth

Returns the width of the bounding box (Feret diameter at 0 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
float GetBoundingBoxWidth() const
```

ECodedElement::ComputeConvexHull

Computes the convex hull of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
void ComputeConvexHull(
 EPathVector& result
 )
```

Parameters

result

The output vector where to store the convex hull.

ECodedElement::ComputeFeretBox

Computes the Feret box at a specific orientation.

Namespace: Euresys::Open_eVision

[C++]

```
ERotatedBoundingBox ComputeFeretBox(
 float angle
 )
```

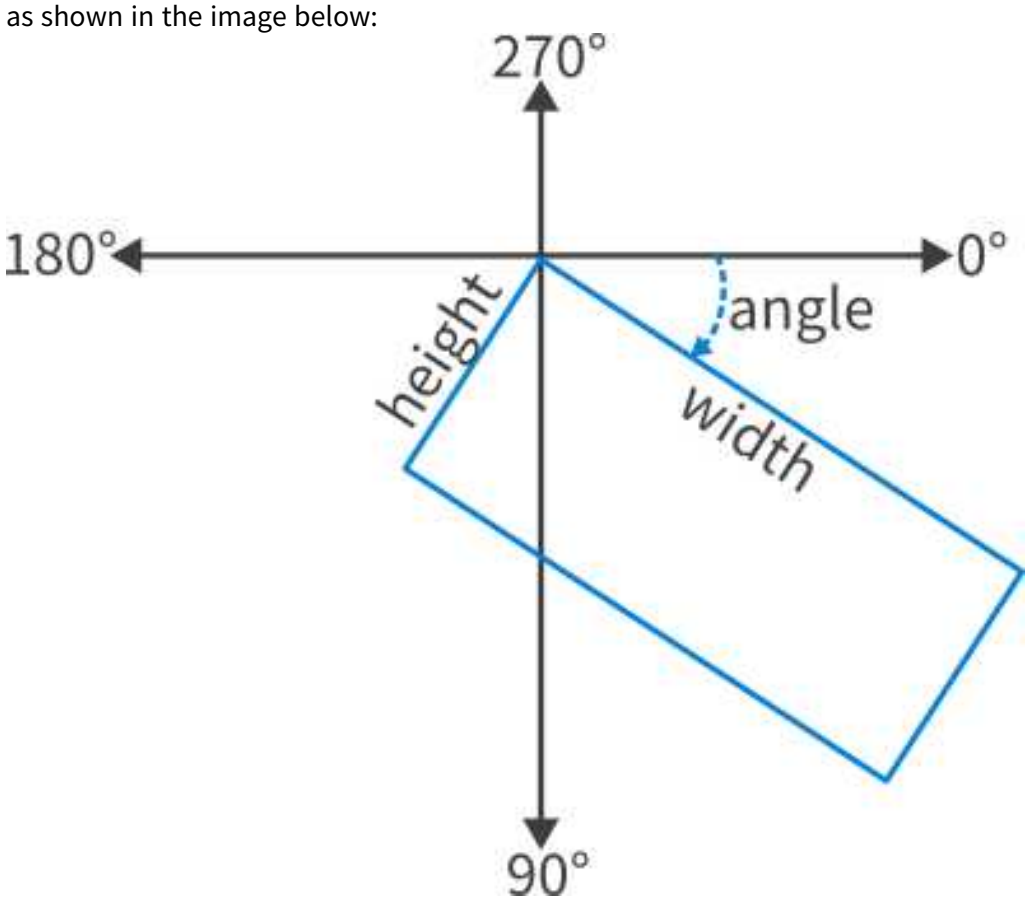
Parameters

angle

The orientation of interest (in the current angle units).

Remarks

The angle of the Feret box is the angle made between the X-axis and the width side of the box as shown in the image below:



ECodedElement::ComputePixelGrayAverage

Computes the average gray-level value of the pixels of a given image over the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
float ComputePixelGrayAverage(
 const EROIBW8& image
 )
```

Parameters

image

The input image.

ECodedElement::ComputePixelGrayDeviation

Computes the standard deviation of the gray-level values of a given image over the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
float ComputePixelGrayDeviation(
 const EROIBW8& image
 )
```

Parameters

image

The input image.

ECodedElement::ComputePixelGrayVariance

Computes the variance of the gray-level values of a given image over the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
double ComputePixelGrayVariance(
 const EROIBW8& image
 )
```

Parameters

image

The input image.

ECodedElement::ComputePixelMax

Computes the maximum gray level of the pixels of a given image over the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
EBW8 ComputePixelMax(
 const EROIBW8& image
 )
```

Parameters

image

The input image.

ECodedElement::ComputePixelMin

Computes the minimum gray level of the pixels of a given image over the coded element.

Namespace: Euresys::Open_eVision

```
[C++]

EBW8 ComputePixelMin(
 const EROIBW8& image
 )
```

Parameters

image

The input image.

ECodedElement::ComputeWeightedGravityCenter

Computes the gravity center of a given image over the coded element.

Namespace: Euresys::Open_eVision

```
[C++]

EPoint ComputeWeightedGravityCenter(
 const EROIBW8& image
 )
```

Parameters

image

The input image.

ECodedElement::GetContour

Returns the coordinates of the starting point of the countour of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]

EPoint GetContour() const
```

Remarks

More precisely, the leftmost pixel over the topmost row of the coded element is taken into consideration.

ECodedElement::GetContourPath

Returns the contour path.
The contour paths is computed using EContourMode_ClockwiseAlwaysClosed.

Namespace: Euresys::Open_eVision

[C++]

```
EPathVector GetContourPath() const
```

ECodedElement::GetContourX

Returns the abscissa of the starting point of the countour of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
int GetContourX() const
```

ECodedElement::GetContourY

Returns the ordinate of the starting point of the countour of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
int GetContourY() const
```

ECodedElement::GetConvexHull

Gets the convex hull of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
const EPathVector& GetConvexHull() const
```

ECodedElement::GetEccentricity

Returns the eccentricity of the ellipse of inertia.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEccentricity() const
```

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element. The eccentricity is zero for circular objects and one for a line-shaped objects.

ECodedElement::GetElementIndex

Returns the index of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetElementIndex() const
```

Remarks

If the coded element is an object, its index is relative to the layer to which it belongs. If the coded element is a hole, its index is relative to its parent object.

ECodedElement::GetEllipseAngle

Returns the angle of the ellipse of inertia.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEllipseAngle() const
```

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element.

ECodedElement::GetEllipseHeight

Returns the length of the short axis of the ellipse of inertia.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEllipseHeight() const
```

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element.

ECodedElement::GetEllipseWidth

Returns the length of the long axis of the ellipse of inertia.

Namespace: Euresys::Open_eVision

```
[C++]
float GetEllipseWidth() const
```

Remarks

The ellipse of inertia is defined as the ellipse that has the same second order moments as the original coded element.

ECodedElement::GetFeretBox22Box

Returns the Feret box at orientation 22.5 degrees.

Namespace: Euresys::Open_eVision

```
[C++]
ERotatedBoundingBox GetFeretBox22Box() const
```

ECodedElement::GetFeretBox22Center

Returns the coordinates of the center of the Feret box oriented at 22.5 degrees.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetFeretBox22Center() const
```

ECodedElement::GetFeretBox22CenterX

Returns the abscissa of the center of the Feret box oriented at 22.5 degrees.

Namespace: Euresys::Open_eVision

```
[C++]
float GetFeretBox22CenterX() const
```

ECodedElement::GetFeretBox22CenterY

Returns the ordinate of the center of the Feret box oriented at 22.5 degrees.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox22CenterY() const
```

ECodedElement::GetFeretBox22Height

Returns the height of the Feret box oriented at 22.5 degrees (Feret diameter at 112.5 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox22Height() const
```

ECodedElement::GetFeretBox22Width

Returns the width of the Feret box oriented at 22.5 degrees (Feret diameter at 22.5 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox22Width() const
```

ECodedElement::GetFeretBox45Box

Returns the Feret box at orientation 45 degrees.

Namespace: Euresys::Open_eVision

[C++]

```
ERotatedBoundingBox GetFeretBox45Box() const
```

ECodedElement::GetFeretBox45Center

Returns the coordinates of the center of the Feret box oriented at 45 degrees.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetFeretBox45Center() const
```

ECodedElement::GetFeretBox45CenterX

Returns the abscissa of the center of the Feret box oriented at 45 degrees.

Namespace: Euresys::Open_eVision

```
[C++]
float GetFeretBox45CenterX() const
```

ECodedElement::GetFeretBox45CenterY

Returns the ordinate of the center of the Feret box oriented at 45 degrees.

Namespace: Euresys::Open_eVision

```
[C++]
float GetFeretBox45CenterY() const
```

ECodedElement::GetFeretBox45Height

Returns the height of the Feret box oriented at 45 degrees (Feret diameter at 135 degrees).

Namespace: Euresys::Open_eVision

```
[C++]
float GetFeretBox45Height() const
```

ECodedElement::GetFeretBox45Width

Returns the width of the Feret box oriented at 45 degrees (Feret diameter at 45 degrees).

Namespace: Euresys::Open_eVision

```
[C++]
float GetFeretBox45Width() const
```

ECodedElement::GetFeretBox68Box

Returns the Feret box at orientation 67.5 degrees.

Namespace: Euresys::Open_eVision

```
[C++]
ERotatedBoundingBox GetFeretBox68Box() const
```

ECodedElement::GetFeretBox68Center

Returns the coordinates of the center of the Feret box oriented at 67.5 degrees.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetFeretBox68Center() const
```

ECodedElement::GetFeretBox68CenterX

Returns the abscissa of the center of the Feret box oriented at 67.5 degrees.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox68CenterX() const
```

ECodedElement::GetFeretBox68CenterY

Returns the ordinate of the center of the Feret box oriented at 67.5 degrees.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox68CenterY() const
```

ECodedElement::GetFeretBox68Height

Returns the height of the Feret box oriented at 67.5 degrees (Feret diameter at 157.5 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox68Height() const
```

ECodedElement::GetFeretBox68Width

Returns the width of the Feret box oriented at 67.5 degrees (Feret diameter at 67.5 degrees).

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretBox68Width() const
```

ECodedElement::GetCentralMoment

Computes the central, two-dimensional moment of order (p,q).

Namespace: Euresys::Open_eVision

[C++]

```
float GetCentralMoment(
 OEV_UINT32 p,
 OEV_UINT32 q
 )
```

Parameters

p

Order of the moment along the X-axis.

q

Order of the moment along the Y-axis.

Remarks

<latex>\mu_{pq} = \sum_{x} \sum_{y} (x - \bar{x})^p(y - \bar{y})^q I(x,y)</latex>

$$\mu_{pq} = \sum_{x} \sum_{y} (x - \bar{x})^p (y - \bar{y})^q I(x, y)$$

ECodedElement::GetMoment

Computes the raw, two-dimensional moment of order (p,q).

Namespace: Euresys::Open_eVision

[C++]

```
double GetMoment(
 OEV_UINT32 p,
 OEV_UINT32 q
 )
```

Parameters

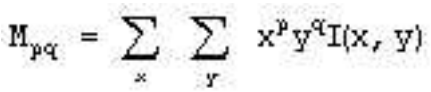
p

Order of the moment along the X-axis.

q

Order of the moment along the Y-axis.

Remarks



<latex>M_{pq} = \sum_x \sum_y x^p y^q I(x,y)</latex>

ECodedElement::GetNormalizedCentralMoment

Computes the scale-invariant, central, two-dimensional moment of order (p,q).

Namespace: Euresys::Open_eVision

```
[C++]
float GetNormalizedCentralMoment(
 OEV_UINT32 p,
 OEV_UINT32 q
 )
```

Parameters

p

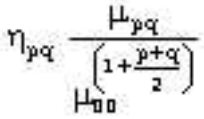
Order of the moment along the X-axis.

q

Order of the moment along the Y-axis.

Remarks

<latex>\eta_{pq} = \frac{\mu_{pq}} {\mu_{00}^{\left(1 + \frac{p+q}{2}\right)}}</latex>



ECodedElement::GetGravityCenter

Returns the gravity center of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetGravityCenter() const
```

ECodedElement::GetGravityCenterX

Returns the abscissa of the gravity center of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
float GetGravityCenterX() const
```

Remarks

For a coded element E, this value is defined as: <latex>\frac{\sum_{(x,y)\in E} x}{\sum_{(x,y)\in E} 1}</latex> $\frac{\sum_{(x,y)\in E} x}{\sum_{(x,y)\in E} 1}$

ECodedElement::GetGravityCenterY

Returns the ordinate of the gravity center of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
float GetGravityCenterY() const
```

Remarks

For a coded element E, this value is defined as: <latex>\frac{\sum_{(x,y)\in E} y}{\sum_{(x,y)\in E} 1}</latex> $\frac{\sum_{(x,y)\in E} y}{\sum_{(x,y)\in E} 1}$

ECodedElement::GetIsCodedElement

Tests whether the coded element is an object, a hole or a coded element.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsCodedElement() const
```

ECodedElement::GetIsHole

Tests whether the coded element is an object, a hole or a coded element.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsHole() const
```

ECodedElement::GetIsObject

Tests whether the coded element is an object, a hole or a coded element.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsObject() const
```

ECodedElement::GetLargestRun

Returns the length of the largest run inside the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetLargestRun() const
```

ECodedElement::GetLayerIndex

Returns the index of the layer in the coded image to which the coded element belongs.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetLayerIndex() const
```

Remarks

If the coded element is a hole, its layer index is defined as that of its parent object.

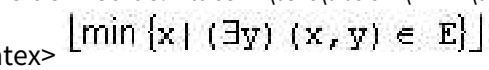
ECodedElement::GetLeftLimit

Returns the lowest (integer) X-coordinate of all the pixels of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
int GetLeftLimit() const
```

Remarks

For a coded element E, this value is defined as: <latex>\left\lfloor \min \left\{ x \mid (\exists y) (x,y) \in E \right\} \right\rfloor </latex> 

ECodedElement::GetMinimumEnclosingRectangle

Returns the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

```
[C++]
ERotatedBoundingBox GetMinimumEnclosingRectangle() const
```

Remarks

The Minimum-Area Enclosing Rectangle is defined as the Feret box with the minimum surface among all the possible orientations.

ECodedElement::GetMinimumEnclosingRectangleAngle

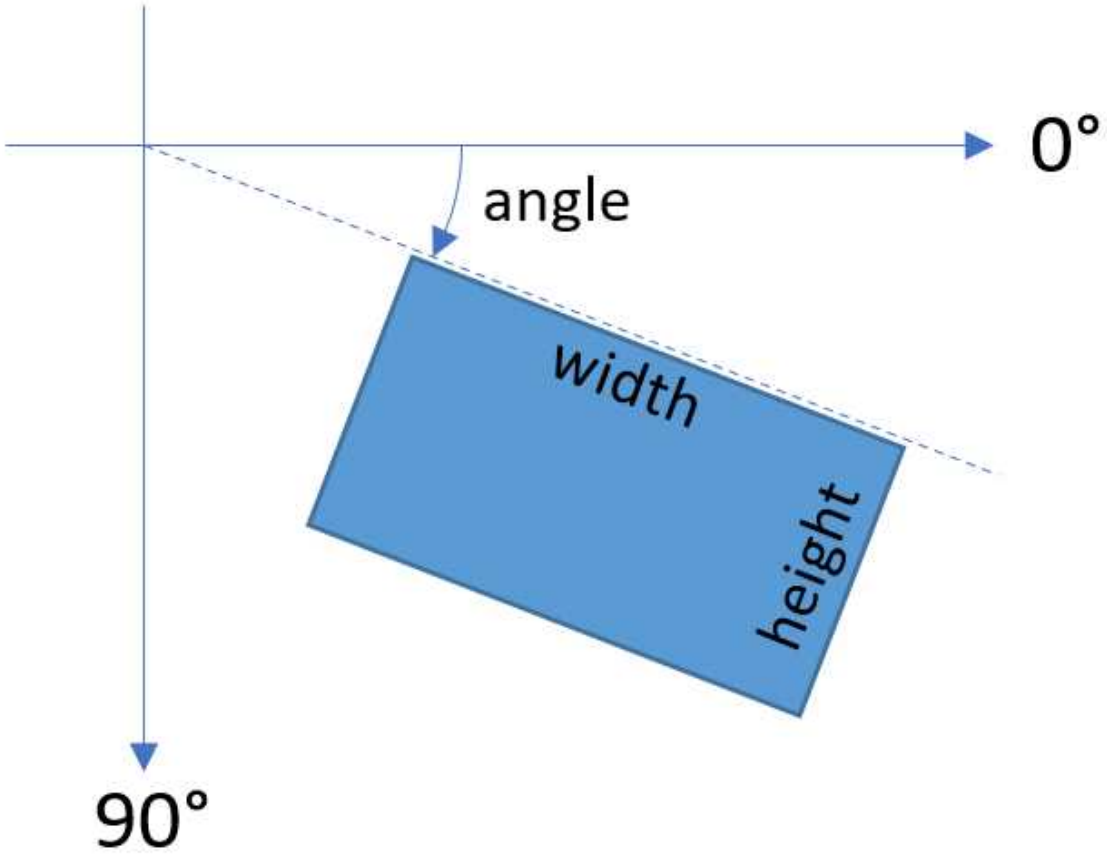
Returns the angle of the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

```
[C++]
float GetMinimumEnclosingRectangleAngle() const
```

Remarks

The angle is the angle between the width side of the rectangle and the horizontal. It always lies in the range [0 ; pi/2[.



ECodedElement::GetMinimumEnclosingRectangleCenter

Returns the coordinates of the center of the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetMinimumEnclosingRectangleCenter() const
```

ECodedElement::GetMinimumEnclosingRectangleCenterX

Returns the abscissa of the center of the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumEnclosingRectangleCenterX() const
```

ECodedElement::GetMinimumEnclosingRectangleCenterY

Returns the ordinate of the center of the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumEnclosingRectangleCenterY() const
```

ECodedElement::GetMinimumEnclosingRectangleHeight

Returns the height of the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumEnclosingRectangleHeight() const
```

ECodedElement::GetMinimumEnclosingRectangleWidth

Returns the width of the Minimum-Area Enclosing Rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumEnclosingRectangleWidth() const
```

ECodedElement::operator==

-

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const ECodedElement& other
 )
```

Parameters

other

-

ECodedElement::RenderMask

Creates a Flexible Mask from the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
void RenderMask(
 EROIBW8& destination,
 int offsetX,
 int offsetY
 )
void RenderMask(
 EROIBW8& destination
 )
```

Parameters

destination

The image in which the generated mask will be stored.

offsetX

The X-offset that must be applied to bring the zero X-coordinate in the coded image on the first column of the result image (defaults to zero).

offsetY

The Y-offset that must be applied to bring the zero Y-coordinate in the coded image on the first row of the result image (defaults to zero).

Remarks

The size of the result image will not be changed: It must be properly sized beforehand.

ECodedElement::GetRightLimit

Returns the highest (integer) X-coordinate of all the pixels of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
int GetRightLimit() const
```

Remarks

For a coded element E, this value is defined as: <latex>\left\lceil \max \left\{ x \mid (\exists y) (x,y) \in E \right\} \right\rceil </latex> $\left\lceil \max \left\{ x \mid (\exists y)\ (x,y) \in E \right\} \right\rceil$

ECodedElement::GetRunCount

Returns the number of runs inside the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetRunCount() const
```

ECodedElement::GetRunsIterator

Returns an iterator to the runs of the coded element.

Namespace: Euresys::Open_eVision

[C++]

```
const EObjectRunsIterator& GetRunsIterator() const
```

ECodedElement::GetSigmaX

Returns the centered moment of inertia around X (average squared X-deviation).

Namespace: Euresys::Open_eVision

[C++]

```
float GetSigmaX() const
```

ECodedElement::GetSigmaXX

Returns the centered cross moment of inertia (average X-deviation * Y-deviation).

Namespace: Euresys::Open_eVision

[C++]

```
float GetSigmaXX() const
```

ECodedElement::GetSigmaXY

Returns the reduced, centered moment of inertia (around the principal inertia axis).

Namespace: Euresys::Open_eVision

[C++]

```
float GetSigmaXY() const
```

ECodedElement::GetSigmaY

Returns the centered moment of inertia around Y (average squared Y-deviation).

Namespace: Euresys::Open_eVision

```
[C++]
float GetSigmaY() const
```

ECodedElement::GetSigmaYY

Returns the reduced, centered moment of inertia (around the secondary inertia axis).

Namespace: Euresys::Open_eVision

```
[C++]
float GetSigmaYY() const
```

ECodedElement::GetTopLimit

Returns the lowest (integer) Y-coordinate of all the pixels of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
int GetTopLimit() const
```

Remarks

For a coded element E, this value is defined as: <latex>\left\lfloor \min \left\{ y \mid (\exists x) (x,y) \in E \right\} \right\rfloor </latex> $\left\lfloor \min \{y \mid (\exists x)\ (x,y) \in E\} \right\rfloor$

ECodedElement::ToRegion

Creates an ERegion from the ECodedElement. The ERegion represents all the pixels which are within the bounding box of the Pattern.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion ToRegion(
 )
```

4.61. ECodedImage Class

This class is deprecated.

This class handles runs, objects and features in EasyObject.

Remarks

These entities are stored into three separate dynamic lists for efficient storage. This class pertains to the EasyObject legacy API and should not be used for new developments. It has been replaced by ECodedImage2.

Namespace: Euresys::Open_eVision

Methods

AddFeat	Adds a feature to the list of features.
AnalyseObjects	After an image segmentation (see ECodedImage::BuildObjects), computes the values of given "features", i.e. geometric parameters.
BlankFeatures	Resets all values of all features.
BuildHoles	Creates holes.
BuildLabeledObjects	Segments an image into connected blobs comprised of pixels of the same class.
BuildLabeledRuns	Extracts the runs from the image by comparing adjacent pixel values.
BuildObjects	Groups runs to form separated objects (connected blobs), from runs detected by ECodedImage::BuildRuns or from a source image/ROI.
BuildRuns	Converts the specified ROI to classes, and extracts the runs from it.
DrawObject	Draws the designated object in solid color.
DrawObjectFeature	Draws a graphical representation of a feature of the designated object in solid color.
DrawObjectFeatureWithCurrentPen	Draws a graphical representation of a feature of the designated object in solid color.
DrawObjects	Draws all objects in solid color.
DrawObjectsFeature	Draws a graphical representation of a feature.
DrawObjectsFeatureWithCurrentPen	Draws a graphical representation of a feature.
DrawObjectsWithCurrentPen	Draws all objects in solid color.
DrawObjectWithCurrentPen	Draws the designated object in solid color.
ECodedImage	Constructs a void coded image.
FeatureAverage	Computes the average of the features of all currently selected objects.
FeatureDeviation	Computes the average and standard deviation of the features of all currently selected objects.
FeatureMaximum	Computes the maximum of the features of all currently selected objects.
FeatureMinimum	Computes the minimum of the features of all currently selected objects.

FeatureVariance	Computes the average and variance of the features of all currently selected objects.
GetBlackClass	Black class index (below the lower threshold).
GetConnexity	Connexity mode, that is how neighboring pixels are considered to belong to the same objects.
GetContinuous	Flag indicating whether the objects have to be built in the continuous mode or in the normal mode.
GetCurrentObjData	Returns the data of the current object.
GetCurrentObjPtr	Pointer to the current objects list item, or NULL.
GetCurrentRunData	Returns the data of the current run.
GetCurrentRunPtr	Pointer to the current run list item, or NULL.
GetDrawDiagonals	Flag indicating whether the limit rectangle diagonals must be drawn or not.
GetFeatData	Gets the EFeatureData associated to a given feature list item.
GetFeatDataSize	Returns the data size of the specified feature.
GetFeatDataType	Returns the data type of the specified feature.
GetFeatNum	Returns the code number of the specified feature.
GetFeatPtrByNum	Returns a pointer to the feature list item for a given feature number, or NULL.
GetFeatSize	Returns the size (number of elements) of the feature array for the specified feature.
GetFirstHole	Returns a pointer to the first hole related to the specified object.
GetFirstObjData	Moves the cursor to the first object and returns the associated data.
GetFirstObjPtr	Pointer to the first objects list item, or NULL.
GetFirstRunData	Moves the cursor to the first run and returns the associated data.
GetFirstRunPtr	Pointer to the first run list item, or NULL.
GetHighColorThreshold	Upper threshold (color) used for image segmentation.
GetHighImage	Image used as an adaptive upper threshold.
GetHighThreshold	Upper threshold (gray level) used for image segmentation.
GetHoleParentObject	Returns a pointer to the real object including the specified hole.
GetLastObjData	Moves the cursor to the last object and returns the associated data.
GetLastObjPtr	Pointer to the last objects list item, or NULL.
GetLastRunData	Moves the cursor to the last run and returns the associated data.
GetLastRunPtr	Pointer to the last run list item, or NULL.
GetLimitAngle	Angle of the skewed bounding box feature, in the current angle unit.
GetLowColorThreshold	Lower threshold (color) used for image segmentation.
GetLowImage	Image used as an adaptive lower threshold.
GetLowThreshold	Lower threshold (gray level) used for image segmentation.

GetMaxObjects	Maximum number of objects to look for.
GetNeutralClass	Neutral class index (between both thresholds).
GetNextHole	Returns a pointer to the hole following the specified hole, in the objects list.
GetNextObjData	Moves the cursor to the next object and returns the associated data.
GetNextObjPtr	Returns a pointer to the next objects list item, or NULL.
GetNextRunData	Moves the cursor to the next run and returns the associated data.
GetNextRunPtr	Returns a pointer to the next run list item, or NULL.
GetNumFeatures	Number of features currently in use.
GetNumHoleRuns	Total number of hole runs in the list of object runs.
GetNumHoles	Returns the number of holes related to the specified object.
GetNumObjectRuns	Returns the number of runs comprised in a given object.
GetNumObjects	Number of objects in the coded image.
GetNumRuns	Total number of runs in the list of object runs.
GetNumSelectedObjects	Number of objects currently selected.
GetObjDataPtr	Gets the EObjectData associated to a given objects list item.
GetObjectData	If an object identification number is given, moves the cursor to the object with a given identification number and returns the associated data. If a list item is given, returns the object data associated with an object list item (cf. EListItem). See also ECodedImage::GetCurrentObjData.
GetObjectFeature	Allows retrieving the value of a feature of a given object.
GetObjFirstRunPtr	Returns a pointer to the first run list item of an object.
GetObjLastRunPtr	Returns a pointer to the last run list item of an object.
GetObjPtr	Returns a pointer to the given objects list item.
GetObjPtrByCoordinates	Returns a pointer to the objects list item that contains the point of given coordinates, or NULL.
GetObjPtrByPos	Returns a pointer to the objects list item of given absolute position, or NULL.
GetPreviousObjData	Moves the cursor to the previous object and returns the associated data.
GetPreviousObjPtr	Returns a pointer to the previous objects list item, or NULL.
GetPreviousRunData	Moves the cursor to the previous run and returns the associated data.
GetPreviousRunPtr	Returns a pointer to the previous run list item, or NULL.
GetRunData	Returns the run data associated to the specified run.
GetRunDataPtr	Gets the ERunData associated to a given run list item.
GetRunPtr	Returns a pointer to the run list item of given absolute position, or NULL.

GetRunPtrByCoordinates	Returns a pointer to the run list item that contains the point of given coordinates, or NULL.
GetThreshold	Threshold mode (gray level) used for image segmentation.
GetTrueThreshold	Absolute threshold level, when using a single threshold.
GetWhiteClass	White class index (above the upper threshold).
IsHole	Returns true if the specified object is a hole, false otherwise.
IsObjectSelected	Sets bSelected to true if an object is selected.
ObjectConvexHull	Computes the convex hull of an object and stores it in a EPathVector.
RemoveAllFeats	Deletes all features from the features list.
RemoveAllObjects	Deletes all objects from the objects list.
RemoveAllRuns	Deletes all runs from the runs list.
RemoveHoles	Permanently erases, from the objects list, holes related to the specified object.
RemoveObject	Deletes an object from the objects list.
RemoveRun	Deletes a run from the runs list.
ResetContinuousMode	When the continuous mode is activated, this method resets the sequence of images.
SelectAllObjects	Selects all objects.
SelectHoles	Selects the holes related to the specified object.
SelectObject	Selects an object.
SelectObjectsUsingFeature	Selects or deselects objects, according to the value of a specified feature.
SelectObjectsUsingPosition	Selects or deselects objects, using a delimiting rectangle.
SetBlackClass	Black class index (below the lower threshold).
SetConnexity	Connexity mode, that is how neighboring pixels are considered to belong to the same objects.
SetContinuous	Flag indicating whether the objects have to be built in the continuous mode or in the normal mode.
SetDrawDiagonals	Flag indicating whether the limit rectangle diagonals must be drawn or not.
SetFeatInfo	Sets the appropriate data size and type for a predefined feature.
SetFirstRunPtr	Sets the first run list item of an object.
SetHighColorThreshold	Upper threshold (color) used for image segmentation.
SetHighImage	Image used as an adaptive upper threshold.
SetHighThreshold	Upper threshold (gray level) used for image segmentation.
SetLastRunPtr	Sets the last run list item of an object.
SetLimitAngle	Angle of the skewed bounding box feature, in the current angle unit.

SetLowColorThreshold	Lower threshold (color) used for image segmentation.
SetLowImage	Image used as an adaptive lower threshold.
SetLowThreshold	Lower threshold (gray level) used for image segmentation.
SetMaxObjects	Maximum number of objects to look for.
SetNeutralClass	Neutral class index (between both thresholds).
SetNumSelectedObjects	Number of objects currently selected.
SetThreshold	Threshold mode (gray level) used for image segmentation.
SetThresholdImage	Single threshold used for image segmentation.
SetWhiteClass	White class index (above the upper threshold).
SortObjectsUsingFeature	Sorts objects according to the value of some feature.
UnselectAllObjects	Deselects all objects.
UnselectHoles	Unselects the holes related to the specified object.
UnselectObject	Deselects an object.

ECodedImage::AddFeat

This method is deprecated.

Adds a feature to the list of features.

Namespace: Euresys::Open_eVision

[C++]

```
void AddFeat(
 EFeatureData* feature,
 int numberOfObjects
 )
```

Parameters

feature

Pointer to an EFeatureData describing the feature.

numberOfObjects

Number of objects for which the feature will be stored.

ECodedImage::AnalyseObjects

This method is deprecated.

After an image segmentation (see ECodedImage::BuildObjects), computes the values of given "features", i.e. geometric parameters.

Namespace: Euresys::Open_eVision

[C++]

```
void AnalyseObjects(
 Euresys::Open_eVision::ELegacyFeature feature1,
 Euresys::Open_eVision::ELegacyFeature feature2,
 Euresys::Open_eVision::ELegacyFeature feature3,
 Euresys::Open_eVision::ELegacyFeature feature4,
 Euresys::Open_eVision::ELegacyFeature feature5,
 Euresys::Open_eVision::ELegacyFeature feature6,
 Euresys::Open_eVision::ELegacyFeature feature7,
 Euresys::Open_eVision::ELegacyFeature feature8,
 Euresys::Open_eVision::ELegacyFeature feature9,
 Euresys::Open_eVision::ELegacyFeature feature10
 )
```

Parameters

feature1
Feature code, as defined by ELegacyFeature.

feature2
Feature code, as defined by ELegacyFeature.

feature3
Feature code, as defined by ELegacyFeature.

feature4
Feature code, as defined by ELegacyFeature.

feature5
Feature code, as defined by ELegacyFeature.

feature6
Feature code, as defined by ELegacyFeature.

feature7
Feature code, as defined by ELegacyFeature.

feature8
Feature code, as defined by ELegacyFeature.

feature9
Feature code, as defined by ELegacyFeature.

feature10
Feature code, as defined by ELegacyFeature.

ECodedImage::GetBlackClass

ECodedImage::SetBlackClass

This property is deprecated.

Black class index (below the lower threshold).

Namespace: Euresys::Open_eVision

[C++]

```
short GetBlackClass()
void SetBlackClass(short n16BlackClass)
```

Remarks

Non zero when the black runs (below the lower threshold) are coded. 0 means "do not code this class". <!-- 1 by default. -->

ECodedImage::BlankFeatures

This method is deprecated.

Resets all values of all features.

Namespace: Euresys::Open_eVision

[C++]

```
void BlankFeatures(
 )
```

ECodedImage::BuildHoles

This method is deprecated.

Creates holes.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildHoles(
 )
void BuildHoles(
 EListItem* object_
 )
```

Parameters

*object_*

Pointer to the objects list item, for which the holes have to be computed.

Remarks

If no argument, the holes are related to all the previously selected real objects. If holes already exist (resulting from a previous call to the ECodedImage::BuildHoles function), they will be removed from the objects list before the new hole building. Otherwise, the holes are related only to the specified object. Previously created holes are not removed before the new holes are built. If holes related to object have already been constructed, they won't be recreated. If object is a hole or is NULL, no hole will be built. The newly created holes will be added to the list of the objects found in the image. Building holes requires two preliminary steps: the construction of real objects and the selection of objects on which the hole detection has to be performed. At the end of the object construction, all the objects are selected.

ECodedImage::BuildLabeledObjects

This method is deprecated.

Segments an image into connected blobs comprised of pixels of the same class.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildLabeledObjects(
 EROIBW8* sourceImage
 )

void BuildLabeledObjects(
 EROIBW16* sourceImage
 )
```

Parameters

sourceImage

Pointer to a source ROI.

Remarks

Uses EBW8 (EBW16) information for class indices, i.e. 255 (65,535) possible classes. Class 0 is not coded. Building objects is the process of grouping pixels from an image to form connected blobs. The pixels are assigned class indices based either on thresholding (ECodedImage::BuildObjects) or on the pixel values themselves (BuildLabeledObjects). A blob is a set of connected pixels of the same class.

ECodedImage::BuildLabeledRuns

This method is deprecated.

Extracts the runs from the image by comparing adjacent pixel values.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildLabeledRuns(
 EROIBW8* sourceImage
 )

void BuildLabeledRuns(
 EROIBW16* sourceImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

Remarks

Uses EBW8 (EBW16) information for class indices, i.e. 255 (65,535) possible classes. Class 0 is not coded. Building runs is the process of grouping pixels from an image to form horizontal segments. The pixels are assigned class indices based either on thresholding (ECodedImage::BuildRuns) or on the pixel values themselves (BuildLabeledRuns). A run is a set of horizontally connected pixels of the same class.

ECodedImage::BuildObjects

This method is deprecated.

Groups runs to form separated objects (connected blobs), from runs detected by ECodedImage::BuildRuns or from a source image/ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildObjects(
 EROIBW8* sourceImage
 )

void BuildObjects(
 EROIC24* sourceImage
 )

void BuildObjects(
 )

void BuildObjects(
 EROIBW1* sourceImage
 )
```

Parameters

sourceImage

Pointer to a source ROI.

Remarks

Without argument, the method groups the runs detected by ECodedImage::BuildRuns to form separate objects, i.e. connected components. With a source ROI as argument, the method segments it into connected blobs comprised of pixels of the same class. The EROIBW8 parameter is converted to white/neutral/black classes, using two thresholds. The EROIC24 parameter is converted to white/black classes, using two color thresholds. Then, the method extracts runs from them, and groups the runs to form separate objects, i.e. connected components. Building objects is the process of grouping pixels from an image to form connected blobs. The pixels are assigned class indices based either on thresholding (BuildObjects) or on the pixel values themselves (ECodedImage::BuildLabeledObjects). A blob is a set of connected pixels of the same class.

ECodedImage::BuildRuns

This method is deprecated.

Converts the specified ROI to classes, and extracts the runs from it.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildRuns(
 EROIBW8* sourceImage
 )

void BuildRuns(
 EROIC24* sourceImage
 )

void BuildRuns(
 EROIBW1* sourceImage
 )
```

Parameters

sourceImage

Pointer to the source ROI.

Remarks

The EROIBW8 parameter is converted to white/neutral/black classes, using two thresholds. The EROIC24 parameter is converted to white/black classes, using two color thresholds. Then, the method extracts runs from them. Building runs is the process of grouping pixels from an image to form horizontal segments. The pixels are assigned class indices based either on thresholding (BuildRuns) or on the pixel values themselves (ECodedImage::BuildLabeledRuns). A run is a set of horizontally connected pixels of the same class.

ECodedImage::GetConnexity

ECodedImage::SetConnexity

This property is deprecated.

Connexity mode, that is how neighboring pixels are considered to belong to the same objects.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EConnexity GetConnexity()
void SetConnexity(Euresys::Open_eVision::EConnexity eConnexity)
```

ECodedImage::GetContinuous

ECodedImage::SetContinuous

This property is deprecated.

Flag indicating whether the objects have to be built in the continuous mode or in the normal mode.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetContinuous()
void SetContinuous(bool bContinuous)
```

Remarks

true if objects are built in the continuous mode, false if objects are built in the normal mode.

ECodedImage::GetCurrentObjPtr

This property is deprecated.

Pointer to the current objects list item, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetCurrentObjPtr()
```

ECodedImage::GetCurrentRunPtr

This property is deprecated.

Pointer to the current run list item, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetCurrentRunPtr()
```

ECodedImage::GetDrawDiagonals

ECodedImage::SetDrawDiagonals

This property is deprecated.

Flag indicating whether the limit rectangle diagonals must be drawn or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetDrawDiagonals()
void SetDrawDiagonals(bool bDrawDiagonals)
```

Remarks

If true (default), diagonals are drawn.

ECodedImage::DrawObject

This method is deprecated.

Draws the designated object in solid color.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObject(
 EDrawAdapter* graphicContext,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObject(
 EDrawAdapter* graphicContext,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawObject(
 HDC graphicContext,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObject(
 HDC graphicContext,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObject(
 HDC graphicContext,
 const ERGBColor& color,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObject(
 HDC graphicContext,
 const ERGBColor& color,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

*object_*

Pointer to the list item (from the objects list) corresponding to the object to be drawn.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number of by list pointer. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjectFeature

This method is deprecated.

Draws a graphical representation of a feature of the designated object in solid color.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObjectFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObjectFeature(
 HDC graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawObjectFeature(
 HDC graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObjectFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObjectFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::ELegacyFeature feature,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by ELegacyFeature (use any value matching the aforementioned features, f.i. ELegacyFeature_EllipseWidth will draw the ellipse of inertia).

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

*object_*

Pointer to the list item (from the objects list) corresponding to the object to be drawn.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number of by list pointer. If a required feature has not been computed for some object, nothing is drawn and no error message is issued! Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjectFeatureWithCurrentPen

This method is deprecated.

Draws a graphical representation of a feature of the designated object in solid color.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObjectFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by ELegacyFeature (use any value matching the aforementioned features, f.i. ELegacyFeature_EllipseWidth will draw the ellipse of inertia).

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number of by list pointer. If a required feature has not been computed for some object, nothing is drawn and no error message is issued! Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjects

This method is deprecated.

Draws all objects in solid color.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjects(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObjects(
 HDC graphicContext,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObjects(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by ESelectionFlag.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the ECodedImage::DrawObject method instead). Optionally, only the selected or deselected objects can be drawn. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjectsFeature

This method is deprecated.

Draws a graphical representation of a feature.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectsFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawObjectsFeature(
 HDC graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObjectsFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::ELegacyFeature feature,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by ELegacyFeature (use any value matching the aforementioned features, f.i. ELegacyFeature_EllipseWidth will draw the ellipse of inertia).

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by ESelectionFlag.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the ECodedImage::DrawObjectFeature method instead). Optionally, only the selected or deselected objects can be drawn. Only the following features can be drawn: * ELegacyFeature_GravityCenter: upright cross; * ELegacyFeature_Centroid: skewed cross; * ELegacyFeature_Limit: upright bounding rectangle with diagonals; * ELegacyFeature_Limit45: skewed bounding rectangle with diagonals; * ELegacyFeature_EllipseWidth: ellipse of inertia (edge and main axis). If a required feature has not been computed for some object, nothing is drawn and no error message is issued! Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjectsFeatureWithCurrentPen

This method is deprecated.

Draws a graphical representation of a feature.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectsFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::ELegacyFeature feature,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

feature

Feature to be drawn, as defined by ELegacyFeature (use any value matching the aforementioned features, f.i. ELegacyFeature_EllipseWidth will draw the ellipse of inertia).

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by ESelectionFlag.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the ECodedImage::DrawObjectFeature method instead). Optionally, only the selected or deselected objects can be drawn. Only the following features can be drawn: * ELegacyFeature_GravityCenter: upright cross; * ELegacyFeature_Centroid: skewed cross; * ELegacyFeature_Limit: upright bounding rectangle with diagonals; * ELegacyFeature_Limit45: skewed bounding rectangle with diagonals; * ELegacyFeature_EllipseWidth: ellipse of inertia (edge and main axis). If a required feature has not been computed for some object, nothing is drawn and no error message is issued! Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjectsWithCurrentPen

This method is deprecated.

Draws all objects in solid color.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectsWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::ESelectionFlag selectionFlag,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

selectionFlag

Tells how the selected/unselected state of the objects is handled, as defined by ESelectionFlag.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all objects (to vary the colors, draw the objects separately using the ECodedImage::DrawObject method instead). Optionally, only the selected or deselected objects can be drawn. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::DrawObjectWithCurrentPen

This method is deprecated.

Draws the designated object in solid color.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectWithCurrentPen(
 HDC graphicContext,
 int objectNumber,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObjectWithCurrentPen(
 HDC graphicContext,
 EListItem* object_,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

objectNumber

Number of the object to be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

*object_*

Pointer to the list item (from the objects list) corresponding to the object to be drawn.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. The objects can be specified either by number of by list pointer. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage::ECodedImage

This method is deprecated.

Constructs a void coded image.

Namespace: Euresys::Open_eVision

```
[C++]
void ECodedImage(
 )
```

ECodedImage::FeatureAverage

This method is deprecated.

Computes the average of the features of all currently selected objects.

Namespace: Euresys::Open_eVision

```
[C++]
void FeatureAverage(
 Euresys::Open_eVision::ELegacyFeature feature,
 float& average
 )
```

Parameters

feature

Feature code, as defined by ELegacyFeature.

average

Reference to the feature average.

Remarks

This measures the central tendency of a population of objects.

ECodedImage::FeatureDeviation

This method is deprecated.

Computes the average and standard deviation of the features of all currently selected objects.

Namespace: Euresys::Open_eVision

```
[C++]
void FeatureDeviation(
 Euresys::Open_eVision::ELegacyFeature feature,
 float& average,
 float& deviation
 )
```

Parameters

feature
Feature code, as defined by ELegacyFeature.

average
Reference to the feature average.

deviation
Reference to the feature standard deviation.

ECodedImage::FeatureMaximum

This method is deprecated.

Computes the maximum of the features of all currently selected objects.

Namespace: Euresys::Open_eVision

```
[C++]
void FeatureMaximum(
 Euresys::Open_eVision::ELegacyFeature feature,
 float& maximum
 )
```

Parameters

feature
Feature code, as defined by ELegacyFeature.

maximum
Reference to the feature maximum.

ECodedImage::FeatureMinimum

This method is deprecated.

Computes the minimum of the features of all currently selected objects.

Namespace: Euresys::Open_eVision

```
[C++]
void FeatureMinimum(
 Euresys::Open_eVision::ELegacyFeature feature,
 float& minimum
 )
```

Parameters

feature
Feature code, as defined by ELegacyFeature.

minimum
Reference to the feature minimum.

ECodedImage::FeatureVariance

This method is deprecated.

Computes the average and variance of the features of all currently selected objects.

Namespace: Euresys::Open_eVision

```
[C++]
void FeatureVariance(
 Euresys::Open_eVision::ELegacyFeature feature,
 float& average,
 float& variance
 )
```

Parameters

feature
Feature code, as defined by ELegacyFeature.

average
Reference to the feature average.

variance
Reference to the feature variance.

Remarks

This measures the central tendency and the dispersion of a population of objects.

ECodedImage::GetFirstObjPtr

This property is deprecated.

Pointer to the first objects list item, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetFirstObjPtr()
```

ECodedImage::GetCurrentObjData

This method is deprecated.

Returns the data of the current object.

Namespace: Euresys::Open_eVision

[C++]

```
void GetCurrentObjData(
 EObjectData* objectData
 )
```

Parameters

objectData

Pointer to an EObjectData structure to receive the data.

Remarks

ECodedImage::GetCurrentRunData

This method is deprecated.

Returns the data of the current run.

Namespace: Euresys::Open_eVision

[C++]

```
void GetCurrentRunData(
 ERunData* run
 )
```

Parameters

run

Pointer to an ERunData to receive the data.

ECodedImage::GetFeatData

This method is deprecated.

Gets the EFeatureData associated to a given feature list item.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFeatData(
 EListItem* currentFeature,
 EFeatureData* featureData
 )
```

Parameters

currentFeature

Pointer to the feature list item.

featureData

Pointer to a EFeatureData to receive the data.

ECodedImage::GetFeatDataSize

This method is deprecated.

Returns the data size of the specified feature.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EDataSize GetFeatDataSize(
 int position
 )

Euresys::Open_eVision::EDataSize GetFeatDataSize(
 EListItem* currentFeature
 )
```

Parameters

position

Absolute position in the features list, counting from 0 on.

currentFeature

Pointer to the feature list item.

Remarks

The features data sizes are defined in EDataSize.

ECodedImage::GetFeatDataType

This method is deprecated.

Returns the data type of the specified feature.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EDataType GetFeatDataType(
 int position
 )
Euresys::Open_eVision::EDataType GetFeatDataType(
 EListItem* currentFeature
 )
```

Parameters

position

Absolute position in the features list, counting from 0 on.

currentFeature

Pointer to the feature list item.

Remarks

The features data types are defined in EDataType.

ECodedImage::GetFeatNum

This method is deprecated.

Returns the code number of the specified feature.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFeatNum(
 int position
 )
int GetFeatNum(
 EListItem* currentFeature
 )
```

Parameters

position

Absolute position in the features list, counting from 0 on.

currentFeature

Pointer to the feature list item.

Remarks

The features code numbers are defined in ELegacyFeature.

ECodedImage::GetFeatPtrByNum

This method is deprecated.

Returns a pointer to the feature list item for a given feature number, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetFeatPtrByNum(
 int numFeat
 )
```

Parameters

numFeat

Feature number, as defined by ELegacyFeature.

ECodedImage::GetFeatSize

This method is deprecated.

Returns the size (number of elements) of the feature array for the specified feature.

Namespace: Euresys::Open_eVision

```
[C++]
int GetFeatSize(
 int position
 )

int GetFeatSize(
 EListItem* currentFeature
 )
```

Parameters

position

Absolute position in the features list, counting from 0 on.

currentFeature

Pointer to the feature list item.

ECodedImage::GetFirstHole

This method is deprecated.

Returns a pointer to the first hole related to the specified object.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetFirstHole(
 EListItem* parentObject
 )
```

Parameters

parentObject

Pointer to the objects list item, for which the first hole has to be pointed out.

Remarks

If parentObject refers to a hole (instead of a real object) or to an object comprising no hole, the ECodedImage::GetFirstHole function returns NULL.

ECodedImage::GetFirstObjData

This method is deprecated.

Moves the cursor to the first object and returns the associated data.

Namespace: Euresys::Open_eVision

[C++]

```
void GetFirstObjData(
 EObjectData* object_
 )
```

Parameters

*object_*

Pointer to an EObjectData to receive the data.

Remarks

ECodedImage::GetFirstRunData

This method is deprecated.

Moves the cursor to the first run and returns the associated data.

Namespace: Euresys::Open_eVision

[C++]

```
void GetFirstRunData(
 ERunData* run
 )
```

Parameters

run

Pointer to an ERunData to receive the data.

ECodedImage::GetFirstRunPtr

This method is deprecated.

Pointer to the first run list item, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetFirstRunPtr(
 )
```

ECodedImage::GetHoleParentObject

This method is deprecated.

Returns a pointer to the real object including the specified hole.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetHoleParentObject(
 EListItem* hole
 )
```

Parameters

hole

Pointer to the hole list item, for which the parent object has to be pointed out.

ECodedImage::GetLastObjData

This method is deprecated.

Moves the cursor to the last object and returns the associated data.

Namespace: Euresys::Open_eVision

```
[C++]
void GetLastObjData(
 EObjectData* object_
 )
```

Parameters

*object_*

Pointer to an EObjectData structure to receive the data.

ECodedImage::GetLastRunData

This method is deprecated.

Moves the cursor to the last run and returns the associated data.

Namespace: Euresys::Open_eVision

```
[C++]
void GetLastRunData(
 ERunData* run
 )
```

Parameters

run

Pointer to an ERunData to receive the data.

ECodedImage::GetLastRunPtr

This method is deprecated.

Pointer to the last run list item, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetLastRunPtr(
 )
```

ECodedImage::GetNextHole

This method is deprecated.

Returns a pointer to the hole following the specified hole, in the objects list.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetNextHole(
 EListItem* hole
 )
```

Parameters

hole

Pointer to the hole list item, for which the following hole has to be pointed out.

Remarks

If there is no hole yet in the list, the ECodedImage::GetNextHole function returns NULL.

ECodedImage::GetNextObjData

This method is deprecated.

Moves the cursor to the next object and returns the associated data.

Namespace: Euresys::Open_eVision

[C++]

```
void GetNextObjData(
 EObjectData* object_
 )
```

Parameters

*object_*

Pointer to an EObjectData to receive the data.

ECodedImage::GetNextObjPtr

This method is deprecated.

Returns a pointer to the next objects list item, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetNextObjPtr(
 EListItem* listItem
 )
```

Parameters

listItem

Pointer to the current objects list item.

ECodedImage::GetNextRunData

This method is deprecated.

Moves the cursor to the next run and returns the associated data.

Namespace: Euresys::Open_eVision

[C++]

```
void GetNextRunData(
 ERunData* run
 )

void GetNextRunData(
 ERunData* run,
 EListItem* listItem
 )
```

Parameters

run

Pointer to an ERunData to receive the data.

listItem

Pointer to the current run list item.

ECodedImage::GetNextRunPtr

This method is deprecated.

Returns a pointer to the next run list item, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetNextRunPtr(
 EListItem* listItem
 )
```

Parameters

listItem

Pointer to the current run list item.

ECodedImage::GetNumHoles

This method is deprecated.

Returns the number of holes related to the specified object.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumHoles(
 EListItem* object_
 )
```

Parameters

*object_*

Pointer to the object list item whose holes have to be counted.

Remarks

By default, the parameter object is set to NULL, meaning that the function returns the total number of holes added to the objects list. After a call to the ECodedImage::BuildHoles function, the ECodedImage::NumObjects and ECodedImage::NumSelectedObjects properties contain the total number of objects (i.e. real objects + holes).

ECodedImage::GetNumObjectRuns

This method is deprecated.

Returns the number of runs comprised in a given object.

Namespace: Euresys::Open_eVision

```
[C++]
int GetNumObjectRuns(
 int objectNumber
 )
int GetNumObjectRuns(
 EListItem* listItem
 )
```

Parameters

objectNumber
Object identification number.

listItem
Pointer to an objects list item.

ECodedImage::GetObjDataPtr

This method is deprecated.

Gets the EObjectData associated to a given objects list item.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetObjDataPtr(
 EListItem* currentFeature,
 EObjectData* objectData
 )
```

Parameters

currentFeature
Pointer to the current objects list item.

objectData
Pointer to a EObjectData to receive the data.

ECodedImage::GetObjectData

This method is deprecated.

If an object identification number is given, moves the cursor to the object with a given identification number and returns the associated data. If a list item is given, returns the object data associated with an object list item (cf. EListItem). See also ECodedImage::GetCurrentObjData.

Namespace: Euresys::Open_eVision

[C++]

```
void GetObjectData(
 EObjectData* object_,
 int objectNumber
 )

void GetObjectData(
 EObjectData* object_,
 EListItem* listItem
 )
```

Parameters

*object_*

Pointer to an EObjectData structure to receive the object data.

objectNumber

Object identification number.

listItem

Pointer to the current object list item (cf. EListItem).

ECodedImage::GetObjectFeature

This method is deprecated.

Allows retrieving the value of a feature of a given object.

Namespace: Euresys::Open_eVision

[C++]

```
void GetObjectFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* objectNumber,
 OEV_INT8& result
 )

void GetObjectFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 short& result
 )

void GetObjectFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 int& result
 )
```

```
void GetObjectFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 float& result
 )

void GetObjectFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 EListItem* object_,
 double& result
 )

void GetObjectFeature(
 int feature,
 int objectNumber,
 OEV_INT8& result
 )

void GetObjectFeature(
 int feature,
 int objectNumber,
 short& result
 )

void GetObjectFeature(
 int feature,
 int objectNumber,
 int& result
 )

void GetObjectFeature(
 int feature,
 int objectNumber,
 float& result
 )

void GetObjectFeature(
 int feature,
 int objectNumber,
 double& result
 )

void GetObjectFeature(
 EListItem* feature,
 int objectNumber,
 OEV_INT8& result
 )

void GetObjectFeature(
 EListItem* feature,
 int objectNumber,
 short& result
 )
```

```
void GetObjectFeature(
 EListItem* feature,
 int objectNumber,
 int& result
 )
void GetObjectFeature(
 EListItem* feature,
 int objectNumber,
 float& result
 )
void GetObjectFeature(
 EListItem* feature,
 int objectNumber,
 double& result
 )
```

Parameters

feature
Pointer to the feature list item.

objectNumber
Object number.

result
Reference to the feature value.

*object_*
Pointer to the list item (from the objects list) corresponding to the object.

ECodedImage::GetObjFirstRunPtr

This method is deprecated.

Returns a pointer to the first run list item of an object.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetObjFirstRunPtr(
 int objectNumber
 )
EListItem* GetObjFirstRunPtr(
 EListItem* listItem
 )
```

Parameters

objectNumber
Object identification number.

listItem
Pointer to the objects list item.

ECodedImage::GetObjLastRunPtr

This method is deprecated.

Returns a pointer to the last run list item of an object.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetObjLastRunPtr(
 int objectNumber
 )
EListItem* GetObjLastRunPtr(
 EListItem* objectNumber
 )
```

Parameters

objectNumber

Object identification number.

ECodedImage::GetObjPtr

This method is deprecated.

Returns a pointer to the given objects list item.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetObjPtr(
 int objectNumber
 )
```

Parameters

objectNumber

Object identification number.

ECodedImage::GetObjPtrByCoordinates

This method is deprecated.

Returns a pointer to the objects list item that contains the point of given coordinates, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetObjPtrByCoordinates(
 int x,
 int y
 )
```

Parameters

x

Point abscissa.

y

Point ordinate.

Remarks

This function is useful for object selection with a mouse.

ECodedImage::GetObjPtrByPos

This method is deprecated.

Returns a pointer to the objects list item of given absolute position, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetObjPtrByPos(
 int position
 )
```

Parameters

position

Absolute position in the objects list, counting from 0 on.

ECodedImage::GetPreviousObjData

This method is deprecated.

Moves the cursor to the previous object and returns the associated data.

Namespace: Euresys::Open_eVision

```
[C++]
void GetPreviousObjData(
 EObjectData* object_
 )
```

Parameters

*object_*

Pointer to an EObjectData to receive the data.

ECodedImage::GetPreviousObjPtr

This method is deprecated.

Returns a pointer to the previous objects list item, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetPreviousObjPtr(
 EListItem* listItem
 )
```

Parameters

listItem

Pointer to the current objects list item.

ECodedImage::GetPreviousRunData

This method is deprecated.

Moves the cursor to the previous run and returns the associated data.

Namespace: Euresys::Open_eVision

```
[C++]
void GetPreviousRunData(
 ERunData* run,
 EListItem* listItem
 )
void GetPreviousRunData(
 ERunData* run
 )
```

Parameters

run

Pointer to an ERunData to receive the data.

listItem

Pointer to the current run list item.

ECodedImage::GetPreviousRunPtr

This method is deprecated.

Returns a pointer to the previous run list item, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetPreviousRunPtr(
 EListItem* listItem
 )
```

Parameters

listItem

Pointer to the current run list item.

ECodedImage::GetRunData

This method is deprecated.

Returns the run data associated to the specified run.

Namespace: Euresys::Open_eVision

[C++]

```
void GetRunData(
 ERunData* run,
 int position
 )

void GetRunData(
 ERunData* run,
 EListItem* listItem
 )
```

Parameters

run

Pointer to an ERunData to receive the data.

position

Absolute position in the run list, counting from 0 on.

listItem

Pointer to the current run list item.

ECodedImage::GetRunDataPtr

This method is deprecated.

Gets the ERunData associated to a given run list item.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRunDataPtr(
 EListItem* currentFeature,
 ERunData* runData
 )
```

Parameters

currentFeature

Pointer to the current run list item.

runData

Pointer to a ERunData to receive the data.

ECodedImage::GetRunPtr

This method is deprecated.

Returns a pointer to the run list item of given absolute position, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetRunPtr(
 int position
 )
```

Parameters

position

Absolute position in the run list, counting from 0 on.

ECodedImage::GetRunPtrByCoordinates

This method is deprecated.

Returns a pointer to the run list item that contains the point of given coordinates, or NULL.

Namespace: Euresys::Open_eVision

[C++]

```
EListItem* GetRunPtrByCoordinates(
 int x,
 int y
 )
```

Parameters

x

Point abscissa.

y
Point ordinate.

Remarks

This function is useful for run selection with a mouse.

ECodedImage::GetHighColorThreshold

ECodedImage::SetHighColorThreshold

This property is deprecated.

Upper threshold (color) used for image segmentation.

Namespace: Euresys::Open_eVision

```
[C++]
EC24 GetHighColorThreshold()
void SetHighColorThreshold(EC24 c24HighThreshold)
```

Remarks

The threshold value is constant over the whole image.

ECodedImage::GetHighImage

ECodedImage::SetHighImage

This property is deprecated.

Image used as an adaptive upper threshold.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW8* GetHighImage()
void SetHighImage(EROIBW8* pImage)
```

Remarks

The threshold is adaptive (specified pixel by pixel).

ECodedImage::GetHighThreshold

ECodedImage::SetHighThreshold

This property is deprecated.

Upper threshold (gray level) used for image segmentation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetHighThreshold()
void SetHighThreshold(OEV_UINT32 un32HighThreshold)
```

Remarks

The threshold value is constant over the whole image.

ECodedImage::IsHole

This method is deprecated.

Returns true if the specified object is a hole, false otherwise.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsHole(
 EListItem* object_
 )
```

Parameters

*object_*

Pointer to the objects list item.

ECodedImage::IsObjectSelected

This method is deprecated.

Sets bSelected to true if an object is selected.

Namespace: Euresys::Open_eVision

```
[C++]
void IsObjectSelected(
 int objectNumber,
 bool& selected
 )
void IsObjectSelected(
 EListItem* listItem,
 bool& selected
 )
```

Parameters

objectNumber
Object identification number.
selected
Reference to the result.
listItem
Pointer to the objects list item.

Remarks

ECodedImage::GetLastObjPtr

This property is deprecated.

Pointer to the last objects list item, or NULL.

Namespace: Euresys::Open_eVision

```
[C++]
EListItem* GetLastObjPtr()
```

ECodedImage::GetLimitAngle

ECodedImage::SetLimitAngle

This property is deprecated.

Angle of the skewed bounding box feature, in the current angle unit.

Namespace: Euresys::Open_eVision

```
[C++]
float GetLimitAngle()
void SetLimitAngle(float f32Angle)
```

ECodedImage::GetLowColorThreshold

ECodedImage::SetLowColorThreshold

This property is deprecated.

Lower threshold (color) used for image segmentation.

Namespace: Euresys::Open_eVision

```
[C++]
EC24 GetLowColorThreshold()
void SetLowColorThreshold(EC24 c24LowThreshold)
```

Remarks

The threshold value is constant over the whole image.

ECodedImage::GetLowImage

ECodedImage::SetLowImage

This property is deprecated.

Image used as an adaptive lower threshold.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW8* GetLowImage()
void SetLowImage(EROIBW8* pImage)
```

Remarks

The threshold value is adaptive (specified pixel by pixel).

ECodedImage::GetLowThreshold

ECodedImage::SetLowThreshold

This property is deprecated.

Lower threshold (gray level) used for image segmentation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetLowThreshold()
void SetLowThreshold(OEV_UINT32 un32LowThreshold)
```

Remarks

The threshold value is constant over the whole image.

ECodedImage::GetMaxObjects

ECodedImage::SetMaxObjects

This property is deprecated.

Maximum number of objects to look for.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMaxObjects()
void SetMaxObjects(OEV_UINT32 un32MaxObjects)
```

Remarks

After having found that amount of object, the process will stop and return an error message. If not set, no maximum value is defined.

ECodedImage::GetNeutralClass

ECodedImage::SetNeutralClass

This property is deprecated.

Neutral class index (between both thresholds).

Namespace: Euresys::Open_eVision

[C++]

```
short GetNeutralClass()
void SetNeutralClass(short n16NeutralClass)
```

Remarks

Non zero when the neutral runs (between both thresholds) are coded. 0 means "do not code this class". <!-- 2 by default. -->

ECodedImage::GetNumFeatures

This property is deprecated.

Number of features currently in use.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumFeatures()
```

ECodedImage::GetNumHoleRuns

This property is deprecated.

Total number of hole runs in the list of object runs.

Namespace: Euresys::Open_eVision

```
[C++]
int GetNumHoleRuns()
```

Remarks

After a call to ECodedImage::BuildHoles, ECodedImage::NumRuns contains the total number of runs (real object runs + hole runs).

ECodedImage::GetNumObjects

This property is deprecated.

Number of objects in the coded image.

Namespace: Euresys::Open_eVision

```
[C++]
int GetNumObjects()
```

Remarks

After a call to ECodedImage::BuildHoles, ECodedImage::NumObjects returns the total number of objects (real objects + holes).

ECodedImage::GetNumRuns

This property is deprecated.

Total number of runs in the list of object runs.

Namespace: Euresys::Open_eVision

```
[C++]
int GetNumRuns()
```

Remarks

After a call to ECodedImage::BuildHoles, ECodedImage::NumRuns returns the total number of runs (real object runs + hole runs).

ECodedImage::GetNumSelectedObjects

ECodedImage::SetNumSelectedObjects

This property is deprecated.

Number of objects currently selected.

Namespace: Euresys::Open_eVision

```
[C++]
int GetNumSelectedObjects()
void SetNumSelectedObjects(int n32Nb_Selected_Objects)
```

Remarks

After a call to ECodedImage::BuildHoles, ECodedImage::NumSelectedObjects returns the total number of objects (real objects + holes).

ECodedImage::ObjectConvexHull

This method is deprecated.

Computes the convex hull of an object and stores it in a EPathVector.

Namespace: Euresys::Open_eVision

```
[C++]
void ObjectConvexHull(
 EListItem* object_,
 EPathVector* destinationVector
 )
```

Parameters

*object_*

Pointer to the objects list item.

destinationVector

Vector containing the vertices coordinates of the convex hull.

ECodedImage::RemoveAllFeats

This method is deprecated.

Deletes all features from the features list.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveAllFeats(
 )
```

ECodedImage::RemoveAllObjects

This method is deprecated.

Deletes all objects from the objects list.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveAllObjects(
 )
```

ECodedImage::RemoveAllRuns

This method is deprecated.

Deletes all runs from the runs list.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveAllRuns(
 )
```

ECodedImage::RemoveHoles

This method is deprecated.

Permanently erases, from the objects list, holes related to the specified object.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveHoles(
 EListItem* object_
 )
```

Parameters

*object_*

Pointer to the objects list item, for which the holes have to be erased.

Remarks

If the parameter is NULL, all the holes are deleted. By default, the parameter is set to NULL, meaning that all the holes have to be erased from the objects list.

ECodedImage::RemoveObject

This method is deprecated.

Deletes an object from the objects list.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveObject(
 int objectNumber
 )

void RemoveObject(
 EListItem* listItem
 )
```

Parameters

objectNumber

Object identification number.

listItem

Pointer to the current objects list item.

ECodedImage::RemoveRun

This method is deprecated.

Deletes a run from the runs list.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveRun(
 int position
 )

void RemoveRun(
 EListItem* listItem
 )
```

Parameters

position

Absolute position in the run list, counting from 0 on.

listItem

Pointer to the current run list item.

ECodedImage::ResetContinuousMode

This method is deprecated.

When the continuous mode is activated, this method resets the sequence of images.

Namespace: Euresys::Open_eVision

```
[C++]
void ResetContinuousMode(
 )
```

Remarks

Thus, the next call for an object building will not take into account any previous image. If the continuous mode is disabled, this method does nothing.

ECodedImage::SelectAllObjects

This method is deprecated.

Selects all objects.

Namespace: Euresys::Open_eVision

```
[C++]
void SelectAllObjects(
 )
```

ECodedImage::SelectHoles

This method is deprecated.

Selects the holes related to the specified object.

Namespace: Euresys::Open_eVision

```
[C++]
void SelectHoles(
 EListItem* parentObject
 )
```

Parameters

parentObject

Pointer to the objects list item, for which the holes have to be selected.

Remarks

If the parameter is NULL, all the holes are selected. By default, the parameter is set to NULL, meaning that all the holes have to be selected. If parentObject is a hole (instead of a real object) or has no hole, no selection is performed.

ECodedImage::SelectObject

This method is deprecated.

Selects an object.

Namespace: Euresys::Open_eVision

[C++]

```
void SelectObject(
 int objectNumber
 )

void SelectObject(
 EListItem* listItem
 )
```

Parameters

objectNumber

Object identification number.

listItem

Pointer to the objects list item.

ECodedImage::SelectObjectsUsingFeature

This method is deprecated.

Selects or deselects objects, according to the value of a specified feature.

Namespace: Euresys::Open_eVision

[C++]

```
void SelectObjectsUsingFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 OEV_INT8 minimum,
 OEV_INT8 maximum,
 Euresys::Open_eVision::ESelectOption operation
 )

void SelectObjectsUsingFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 short minimum,
 short maximum,
 Euresys::Open_eVision::ESelectOption operation
 )
```

```
void SelectObjectsUsingFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 int minimum,
 int maximum,
 Euresys::Open_eVision::ESelectOption operation
 )
void SelectObjectsUsingFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 float minimum,
 float maximum,
 Euresys::Open_eVision::ESelectOption operation
 )
void SelectObjectsUsingFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 double minimum,
 double maximum,
 Euresys::Open_eVision::ESelectOption operation
 )
```

Parameters

feature

Feature code, as defined by EFeature.

minimum

Selection interval lower bound.

maximum

Selection interval upper bound.

operation

Selection mode, as defined by ESelectOption.

ECodedImage::SelectObjectsUsingPosition

This method is deprecated.

Selects or deselects objects, using a delimiting rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void SelectObjectsUsingPosition(
 EBaseROI* roi,
 Euresys::Open_eVision::ESelectByPosition operation
 )
```

```
void SelectObjectsUsingPosition(
 int originX,
 int originY,
 int width,
 int height,
 Euresys::Open_eVision::ESelectByPosition operation
 )
```

Parameters

roi
Pointer to an image/ROI whose position parameters will define the selection rectangle.
operation
Selection mode, as defined by ESelectByPosition.
originX
Abscissa of the upper left corner of the rectangle.
originY
Ordinate of the upper left corner of the rectangle.
width
Rectangle width, in pixels.
height
Rectangle height, in pixels.

Remarks

The rectangle coordinates are always specified with respect to the whole image.

ECodedImage::SetFeatInfo

This method is deprecated.

Sets the appropriate data size and type for a predefined feature.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFeatInfo(
 EFeatureData* feature,
 Euresys::Open_eVision::ELegacyFeature featureCode
 )
```

Parameters

feature
Feature code, as defined by ELegacyFeature.
featureCode
Pointer to a EFeatureData structure describing the feature.

ECodedImage::SetFirstRunPtr

This method is deprecated.

Sets the first run list item of an object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFirstRunPtr(
 EListItem* firstRun,
 int objectNumber
 )

void SetFirstRunPtr(
 EListItem* firstRun,
 EListItem* currentObject
 )
```

Parameters

firstRun

Pointer to the first run of the object.

objectNumber

Object identification number.

currentObject

Pointer to the objects list item.

ECodedImage::SetLastRunPtr

This method is deprecated.

Sets the last run list item of an object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetLastRunPtr(
 EListItem* lastRun,
 int objectNumber
 )

void SetLastRunPtr(
 EListItem* lastRun,
 EListItem* currentObject
 )
```

Parameters

lastRun

Pointer to the last run of the object.

objectNumber
Object identification number.
currentObject
Pointer to the objects list item.

ECodedImage::SortObjectsUsingFeature

This method is deprecated.

Sorts objects according to the value of some feature.

Namespace: Euresys::Open_eVision

[C++]

```
void SortObjectsUsingFeature(
 Euresys::Open_eVision::ELegacyFeature feature,
 Euresys::Open_eVision::ESortOption Operation
 )
```

Parameters

feature
Feature code, as defined by ELegacyFeature.
Operation
Selection mode, as defined by ESortOption.

ECodedImage::GetThreshold

ECodedImage::SetThreshold

This property is deprecated.

Threshold mode (gray level) used for image segmentation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 un32Threshold)
```

Remarks

By default, the "minimum residue" mode is used to determine the threshold. The threshold is constant over the whole image. When using a single threshold instead of a low threshold and a high threshold, the neutral class is ignored. Only the black and white classes are relevant.

ECodedImage::SetThresholdImage

This property is deprecated.

Single threshold used for image segmentation.

Namespace: Euresys::Open_eVision

```
[C++]
void SetThresholdImage(EROIBW8* pImage)
```

Remarks

The threshold value is adaptive (specified pixel by pixel). When using a single threshold instead of a low threshold and a high threshold, the neutral class is ignored. Only the black and white classes are relevant.

ECodedImage::GetTrueThreshold

This property is deprecated.

Absolute threshold level, when using a single threshold.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetTrueThreshold()
```

ECodedImage::UnselectAllObjects

This method is deprecated.

Deselects all objects.

Namespace: Euresys::Open_eVision

```
[C++]
void UnselectAllObjects(
 )
```

ECodedImage::UnselectHoles

This method is deprecated.

Unselects the holes related to the specified object.

Namespace: Euresys::Open_eVision

[C++]

```
void UnselectHoles(
 EListItem* parentObject
 )
```

Parameters

parentObject

Pointer to the objects list item, for which the holes have to be unselected.

Remarks

If the parameter is NULL, all the holes are unselected. By default, the parameter is set to NULL, meaning that all the holes have to be unselected. If parentObject is a hole (instead of a real object) or if parentObject has no hole, nothing is changed.

ECodedImage::UnselectObject

This method is deprecated.

Deselects an object.

Namespace: Euresys::Open_eVision

[C++]

```
void UnselectObject(
 int objectNumber
 )

void UnselectObject(
 EListItem* listItem
 )
```

Parameters

objectNumber

Object identification number.

listItem

Pointer to the objects list item.

Remarks

Once an object has been unselected, it doesn't allow browsing a list of selected objects anymore (using ECodedImage::GetPreviousObjPtr or ECodedImage::GetNextObjPtr).

ECodedImage::GetWhiteClass

ECodedImage::SetWhiteClass

This property is deprecated.

White class index (above the upper threshold).

Namespace: Euresys::Open_eVision

```
[C++]
short GetWhiteClass()
void SetWhiteClass(short n16WhiteClass)
```

Remarks

Non zero when the white runs (above the upper threshold) are coded. 0 means "do not code this class". <!-- 3 by default. -->

4.62. ECodedImage2 Class

The main class of the EasyObject API that represents a coded image, as produced by the encoder.

Remarks

It provides methods to get features of the encoded image, to access an object in a particular layer of the encoded image, to draw objects or objects features in a particular layer of the encoded image, to render a mask, etc...

Namespace: Euresys::Open_eVision

Methods

ClearFeatureCache	Clears the internal cache for the computed features.
Draw	Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.
DrawFeature	Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.
DrawFeatureWithCurrentPen	Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.
DrawHole	Draw the designated hole.
DrawHoleFeature	Draw a given feature of the designated hole.
DrawHoleFeatureWithCurrentPen	Draw a given feature of the designated hole.
DrawHoleWithCurrentPen	Draw the designated hole.
DrawObject	Draw the designated object.
DrawObjectFeature	Draw a given feature of the designated object.
DrawObjectFeatureWithCurrentPen	Draw a given feature of the designated object.

DrawObjectWithCurrentPen	Draw the designated object.
DrawWithCurrentPen	Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.
ECodedImage2	Constructs a coded image.
FindObject	Finds an object in the coded image using its coordinates.
GetHeight	Returns the height of the coded image.
GetLayerCount	Returns the number of layers that are encoded.
GetObj	Random access to an object in a given layer.
GetObjCount	Returns the number of objects in a given layer.
GetParentObject	Returns a reference to the object that contains a given hole.
GetStartY	Returns the lowest row index, for all the runs of all the objects in the coded image.
GetWidth	Returns the width of the coded image.
RenderMask	Creates a Flexible Mask from a specified layer of the encoded image.
ToRegion	Converts the encoded image to an ERegion.

ECodedImage2::ClearFeatureCache

Clears the internal cache for the computed features.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearFeatureCache(
 )
```

Remarks

This is useful to reduce memory consumption.

ECodedImage2::Draw

Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

element

The coded element to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawFeature

Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

```
void DrawFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

```
void DrawFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

```
void DrawFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

element

The coded element to draw.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

The features EDrawableFeature_FeretBox and EDrawableFeature_WeightedGravityCenter are only at one's disposal when drawing selections.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawFeatureWithCurrentPen

This method is deprecated.

Draw a given feature of the designated coded element, of all the objects from a layer, or of all the coded elements from a given selection.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

element

The coded element to draw.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

The features EDrawableFeature_FeretBox and EDrawableFeature_WeightedGravityCenter are only at one's disposal when drawing selections.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawHole

Draw the designated hole.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawHole(
 EDrawAdapter* graphicContext,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHole(
 EDrawAdapter* graphicContext,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHole(
 HDC graphicContext,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHole(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHole(
 HDC graphicContext,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawHole(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

objectIndex

Index of the parent object of the hole to draw.

holeIndex

Index of the hole to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image <contains several layers. Indeed, in such a case, no default layer exists.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawHoleFeature

Draw a given feature of the designated hole.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawHoleFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawHoleFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawHoleFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawHoleFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

```
void DrawHoleFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawHoleFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext
Graphic context on which to draw.

feature
The feature of interest.

objectIndex
Index of the parent object of the hole to draw.

holeIndex
Index of the hole to draw.

zoomX
Horizontal zooming factor. By default, no scaling is used.

zoomY
Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX
Horizontal panning value expressed in pixels. By default, no panning occurs.

panY
Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals
Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the <default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features EDrawableFeature_FeretBox and EDrawableFeature_WeightedGravityCenter will <result in an exception, as they only make sense for EObjectSelection.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawHoleFeatureWithCurrentPen

This method is deprecated.

Draw a given feature of the designated hole.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawHoleFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawHoleFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext
Graphic context on which to draw.

feature
The feature of interest.

objectIndex
Index of the parent object of the hole to draw.

holeIndex
Index of the hole to draw.

zoomX
Horizontal zooming factor. By default, no scaling is used.

zoomY
Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX
Horizontal panning value expressed in pixels. By default, no panning occurs.

panY
Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals
Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex
The index of the layer of interest. If this parameter is left unspecified, the <default layer will be taken into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features EDrawableFeature_FeretBox and EDrawableFeature_WeightedGravityCenter will <result in an exception, as they only make sense for EObjectSelection.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawHoleWithCurrentPen

This method is deprecated.

Draw the designated hole.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawHoleWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawHoleWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

objectIndex

Index of the parent object of the hole to draw.

holeIndex

Index of the hole to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image <contains several layers. Indeed, in such a case, no default layer exists.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawObject

Draw the designated object.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObject(
 EDrawAdapter* graphicContext,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObject(
 EDrawAdapter* graphicContext,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObject(
 HDC graphicContext,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawObject(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawObject(
 HDC graphicContext,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawObject(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image <contains several layers. Indeed, in such a case, no default layer exists.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawObjectFeature

Draw a given feature of the designated object.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawObjectFeature(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawObjectFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawObjectFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

```
void DrawObjectFeature(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void DrawObjectFeature(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features EDrawableFeature_FeretBox and EDrawableFeature_WeightedGravityCenter will result in an exception, as they only make sense for EObjectSelection.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawObjectFeatureWithCurrentPen

This method is deprecated.

Draw a given feature of the designated object.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void DrawObjectFeatureWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawableFeature feature,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

feature

The feature of interest.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the ellipses and of the rectangles are to be drawn.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

Trying to draw the features EDrawableFeature_FeretBox and EDrawableFeature_WeightedGravityCenter will result in an exception, as they only make sense for EObjectSelection.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawObjectWithCurrentPen

This method is deprecated.

Draw the designated object.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjectWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawObjectWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

objectIndex

Index of the object to draw.

zoomX

Horizontal zooming factor. By default, no scaling is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image <contains several layers. Indeed, in such a case, no default layer exists.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::DrawWithCurrentPen

This method is deprecated.

Draw the designated coded element, all the objects from a layer, or all the coded elements from a given selection.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 const ECodedElement& element,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 layerIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawWithCurrentPen(
 HDC graphicContext,
 const EObjectSelection& selection,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawWithCurrentPen(
 HDC graphicContext,
 const EObjectSelection& selection,
 OEV_UINT32 elementIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

element

The coded element to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

layerIndex

Index of the layer of interest. If no layer index is specified, the default layer is taken into consideration. Note that this methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

selection

The selection of coded elements to draw.

elementIndex

The index in the selection of the coded element to draw. If no element index is specified, all the elements of the selection are take into consideration.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ECodedImage2::ECodedImage2

Constructs a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void ECodedImage2(
 )
```

ECodedImage2::FindObject

Finds an object in the coded image using its coordinates.

Namespace: Euresys::Open_eVision

[C++]

```
const EObject* FindObject(
 int x,
 int y
 )
```

```
const EObject* FindObject(
 OEV_UINT32 layerIndex,
 int x,
 int y
 )
```

Parameters

x

The X-coordinate of the object.

y

The Y-coordinate of the object.

layerIndex

The index of the layer of interest.

Remarks

If no layer index is specified, all the layers of the coded image are scanned until an object is found at these coordinates.

ECodedImage2::GetObj

Random access to an object in a given layer.

Namespace: Euresys::Open_eVision

[C++]

```
EObject& GetObj(
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )

const EObject& GetObj(
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )

EObject& GetObj(
 OEV_UINT32 objectIndex
 )

const EObject& GetObj(
 OEV_UINT32 objectIndex
 )
```

Parameters

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

objectIndex

The index of the object in the layer.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2::GetObjCount

Returns the number of objects in a given layer.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetObjCount(
 OEV_UINT32 layerIndex
 )

OEV_UINT32 GetObjCount(
 )
```

Parameters

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2::GetParentObject

Returns a reference to the object that contains a given hole.

Namespace: Euresys::Open_eVision

[C++]

```
EObject& GetParentObject(
 const EHole& hole
 )

const EObject& GetParentObject(
 const EHole& hole
 )
```

Parameters

hole

The hole of interest.

ECodedImage2::GetHeight

Returns the height of the coded image.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetHeight() const
```

Remarks

If the continuous mode is not activated, this height corresponds to the height of the source image. If the continuous mode is activated, this value equals to the highest row index, for all the runs of all the objects in the coded image, augmented by the number of rows index that are below zero.

ECodedImage2::GetLayerCount

Returns the number of layers that are encoded.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetLayerCount() const
```

ECodedImage2::RenderMask

Creates a Flexible Mask from a specified layer of the encoded image.

Namespace: Euresys::Open_eVision

[C++]

```
void RenderMask(
 EROIBW8& result
 )

void RenderMask(
 EROIBW8& result,
 OEV_UINT32 layerIndex,
 int offsetX,
 int offsetY
 )
```

```
void RenderMask(
 EROIBW8& result,
 OEV_UINT32 layerIndex
 )
void RenderMask(
 EROIBW8& result,
 int offsetX,
 int offsetY
 )
```

Parameters

result

The image in which the generated mask will be stored.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will serve as a source for the mask generation.

offsetX

The X-offset that must be applied to bring the zero X-coordinate in the coded image on the first column of the result image (defaults to zero).

offsetY

The Y-offset that must be applied to bring the zero Y-coordinate in the coded image on the first row of the result image (defaults to zero).

Remarks

The size of the result image will not be changed: It must be properly sized beforehand.

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

ECodedImage2::GetStartY

Returns the lowest row index, for all the runs of all the objects in the coded image.

Namespace: Euresys::Open_eVision

[C++]

```
int GetStartY() const
```

Remarks

The returned value will always be zero if the continuous mode is not activated.

ECodedImage2::ToRegion

Converts the encoded image to an ERegion.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion ToRegion(
 )
```

ECodedImage2::GetWidth

Returns the width of the coded image.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetWidth() const
```

Remarks

This width corresponds in any case to the width of the source image.

4.63. ECodeGrid Class

Represents a grid of Codes

Namespace: Euresys::Open_eVision

Methods

ECodeGrid	Creates an ECodeGrid object.
GetCellEnabled	Returns true if Cell is enabled and false otherwise
GetNumCols	Returns the number of columns in the grid
GetNumRows	Returns the number of rows in the grid
GetResults	Returns the detected codes
operator=	Assignment operator
SetEnableAll	Enable/Disable all cells
SetEnableCell	Enable/Disable Cell
SetEnableColumn	Enable/Disable Column
SetEnableRow	Enable/Disable Row

ECodeGrid::ECodeGrid

Creates an ECodeGrid object.

Namespace: Euresys::Open_eVision

[C++]

```
void ECodeGrid(
 )
void ECodeGrid(
 const ECodeGrid& other
 )
void ECodeGrid(
 OEV_UINT32 numCols,
 OEV_UINT32 numRows
 )
```

Parameters

other

The reference ECodeGrid instance to copy this one from.

numCols

The number of columns in the grid.

numRows

The number of rows in the grid.

ECodeGrid::SetEnableAll

Enable/Disable all cells

Namespace: Euresys::Open_eVision

[C++]

```
void SetEnableAll(bool enable)
```

Remarks

By default, all grid cells are enabled.

ECodeGrid::GetCellEnabled

Returns true if Cell is enabled and false otherwise

Namespace: Euresys::Open_eVision

[C++]

```
bool GetCellEnabled(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
```

Parameters

column

-

row

-

Remarks

By default, all grid cells are enabled.

ECodeGrid::GetResults

Returns the detected codes

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::ECode> GetResults(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
std::vector<Euresys::Open_eVision::ECode> GetResults(
 )
```

Parameters

column

The column of a cell.

row

The row of a cell.

ECodeGrid::GetNumCols

Returns the number of columns in the grid

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumCols() const
```

ECodeGrid::GetNumRows

Returns the number of rows in the grid

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumRows() const
```

ECodeGrid::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
ECodeGrid& operator=(
 const ECodeGrid& other
 )
```

Parameters

other

The ECodeGrid instance to assign.

ECodeGrid::SetEnableCell

Enable/Disable Cell

Namespace: Euresys::Open_eVision

```
[C++]
void SetEnableCell(
 OEV_UINT32 column,
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

column

-

row

-

enable

-

Remarks

By default, all grid cells are enabled.

ECodeGrid::SetEnableColumn

Enable/Disable Column

Namespace: Euresys::Open_eVision

```
[C++]
void SetEnableColumn(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

ECodeGrid::SetEnableRow

Enable/Disable Row

Namespace: Euresys::Open_eVision

```
[C++]
void SetEnableRow(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

4.64. ECodeReader Class

An ECodeReader instance can detect and decode various types of codes.

Namespace: Euresys::Open_eVision

Methods

ECodeReader	Creates an ECodeReader object.
GetBarCodeReader	Gets the underlying EBarCodeReader instance
GetEnabledCodeTypes	The enabled code types
GetMatrixCodeReader	Gets the underlying EMatrixCodeReader instance
GetMaxNumCodesPerType	The maximum number of codes that the reader should try to find.
GetQRCodeReader	Gets the underlying EQRCodeReader instance
GetTimeOut	The timeout for the ECodeReader::Read method in microseconds.
Load	Load the configuration for this ECodeReader instance.
operator=	Assignment operator
Read	Tries to locate, decode and read data codes in the given ROI
Save	Save the configuration for this ECodeReader instance.
SetEnabledCodeTypes	The enabled code types
SetMaxNumCodesPerType	The maximum number of codes that the reader should try to find.
SetTimeOut	The timeout for the ECodeReader::Read method in microseconds.

ECodeReader::GetBarCodeReader

Gets the underlying EBarCodeReader instance

Namespace: Euresys::Open_eVision

[C++]

```
EBarCodeReader& GetBarCodeReader()
```

ECodeReader::ECodeReader

Creates an ECodeReader object.

Namespace: Euresys::Open_eVision

[C++]

```
void ECodeReader(
 )
```

```
void ECodeReader(
 const ECodeReader& other
 )
```

Parameters

other

Another ECodeReader object to be copied in the new ECodeReader object.

ECodeReader::GetEnabledCodeTypes

ECodeReader::SetEnabledCodeTypes

The enabled code types

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::ECodeType> GetEnabledCodeTypes()
void SetEnabledCodeTypes(std::vector<Euresys::Open_eVision::ECodeType> types)
```

Remarks

All code types enabled by default

ECodeReader::Load

Load the configuration for this ECodeReader instance.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The path from which to load the configuration.

serializer

The serializer.

ECodeReader::GetMatrixCodeReader

Gets the underlying EMatrixCodeReader instance

Namespace: Euresys::Open_eVision

```
[C++]
EMatrixCodeReader& GetMatrixCodeReader()
```

ECodeReader::GetMaxNumCodesPerType

ECodeReader::SetMaxNumCodesPerType

The maximum number of codes that the reader should try to find.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMaxNumCodesPerType() const
void SetMaxNumCodesPerType(OEV_UINT32 max)
```

Remarks

By default, this parameter is set to 1.

ECodeReader::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
ECodeReader& operator=(
 const ECodeReader& other
 )
```

Parameters

other

ECodeReader object to be copied.

ECodeReader::GetQRCodeReader

Gets the underlying EQRCodeReader instance

Namespace: Euresys::Open_eVision

```
[C++]
EQRCodeReader& GetQRCodeReader()
```

ECodeReader::Read

Tries to locate, decode and read data codes in the given ROI

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::ECode> Read(
 const EROIBW8& field
 )

std::vector<Euresys::Open_eVision::ECode> Read(
 const EROIBW8& field,
 const ERegion& region
 )

ECodeGrid Read(
 const EROIBW8& roi,
 const ERectangleRegion& area,
 int numCellsX,
 int numCellsY,
 float extension
 )

ECodeGrid Read(
 const EROIBW8& roi,
 const ERectangleRegion& area,
 const ECodeGrid& grid,
 float extension
 )
```

Parameters

field

-

region

Region into the search field where the data matrix codes have to be found.

roi

The ROI in which the data matrix codes have to be found.

area

Rectangular Region used as the full grid area

numCellsX

Number of grid cells in the X direction

numCellsY

Number of grid cells in the Y direction

extension

Extension of the grid cells to allow cell overlap. For instance, 0.0f means no extension and 0.1f means a 10% cell size extension.

grid

Grid with cell disabling capabilities

Remarks

The grid overload allows you to disable some cells of the grid if those cells are not supposed to contain codes. See the ECodeGrid class documentation for more information.

ECodeReader::Save

Save the configuration for this ECodeReader instance.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The path to which to save the configuration.

serializer

The serializer.

ECodeReader::GetTimeOut

ECodeReader::SetTimeOut

The timeout for the ECodeReader::Read method in microseconds.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT64 GetTimeOut() const

void SetTimeOut(OEV_UINT64 timeOut)
```

Remarks

If the processing time of one of these methods becomes longer than the set time-out period, the process is stopped.
The ECodeReader::Read method will return all the codes it has decoded up to that point.
Note that the time-out period is not exact: the process is stopped at the first checkpoint after the time-out period has elapsed.

4.65. EColorLookup Class

Describes a color lookup table, that is used to speed-up complex conversions between color systems.

Namespace: Euresys::Open_eVision

Methods

AdjustGainOffset	Sets a color transformation such that a gain and offset is applied separately to each color component of a target color system.
Calibrate	Sets a color transformation to recalibrate.
ConvertFromRgb	Sets a color transformation from the EColorSystem_Rgb representation to another system, as defined by EColorSystem.
ConvertToRgb	Sets a color transformation from any color system, as defined by EColorSystem, to the EColorSystem_Rgb representation.
EColorLookup	Constructs a color lookup table.
GetColorSystemIn	Input color system.
GetColorSystemOut	Output color system.
GetIndexBits	Number of bits used for indexing the lookup table.
GetInterpolation	Interpolation mode.
SetIndexBits	Number of bits used for indexing the lookup table.
SetInterpolation	Interpolation mode.
Transform	Transforms a quantized color image/pixel to another quantized color image/pixel, using the previously initialized color lookup table.
WhiteBalance	Initializes a color lookup table that can be used for color adjustment, including a global gain, gamma pre-compensation [cancellation] and white balancing.

EColorLookup::AdjustGainOffset

Sets a color transformation such that a gain and offset is applied separately to each color component of a target color system.

Namespace: Euresys::Open_eVision

```
[C++]
void AdjustGainOffset(
 Euresys::Open_eVision::EColorSystem colorSystem,
 float gain0,
 float offset0,
 float gain1,
 float offset1,
 float gain2,
 float offset2
 )
```

Parameters

colorSystem

Target color system, as defined by EColorSystem.

gain0

Gain to be applied to color component 0.

offset0

Offset to be applied to color component 0.

gain1

Gain to be applied to color component 1.

offset1

Offset to be applied to color component 1.

gain2

Gain to be applied to color component 2.

offset2

Offset to be applied to color component 2.

Remarks

The input and output color systems are both EColorSystem_Rgb. To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as EColorLookup::Transform.

Note. The offsets are specified as unquantized values.

EColorLookup::Calibrate

Sets a color transformation to recalibrate.

Namespace: Euresys::Open_eVision

[C++]

```
void Calibrate(
 EC24 Color0,
 float x0,
 float y0,
 float z0,
 EC24 Color1,
 float x1,
 float y1,
 float z1,
 EC24 Color2,
 float x2,
 float y2,
 float z2
 )
void Calibrate(
 EC24 Color0,
 float x0,
 float y0,
 float z0,
 EC24 Color1,
 float x1,
 float y1,
 float z1,
 EC24 Color2,
 float x2,
 float y2,
 float z2,
 EC24 Color3,
 float x3,
 float y3,
 float z3
 )
```

Parameters

Color0
Measured quantized values of a pixel of color 0.

x0
CIE XYZ tri-stimulus unquantized values corresponding to color 0.

y0
CIE XYZ tri-stimulus unquantized values corresponding to color 0.

z0
CIE XYZ tri-stimulus unquantized values corresponding to color 0.

Color1
Measured quantized values of a pixel of color 1.

x1
CIE XYZ tri-stimulus unquantized values corresponding to color 1.

y1
CIE XYZ tri-stimulus unquantized values corresponding to color 1.

z1
CIE XYZ tri-stimulus unquantized values corresponding to color 1.
Color2
Measured quantized values of a pixel of color 2.
x2
CIE XYZ tri-stimulus unquantized values corresponding to color 2.
y2
CIE XYZ tri-stimulus unquantized values corresponding to color 2.
z2
CIE XYZ tri-stimulus unquantized values corresponding to color 2.
Color3
Measured quantized values of a pixel of color 3.
x3
CIE XYZ tri-stimulus unquantized values corresponding to color 3.
y3
CIE XYZ tri-stimulus unquantized values corresponding to color 3.
z3
CIE XYZ tri-stimulus unquantized values corresponding to color 3.
color
-
x
-
y
-
z
-

Remarks

The first prototype uses 3 reference colors. The second uses 4 reference colors. To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as EColorLookup::Transform.

EColorLookup::GetColorSystemIn

Input color system.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EColorSystem GetColorSystemIn()
```

Remarks

The EColorLookup objects keep track of the color system transformation, for consistency. When applying a transformation, the source image color system (usually EColorSystem_Rgb) must match the *input color system*; the destination image will be automatically be typed with the *output color system*. In case of a mismatch, an error message is issued. These two values are set by the lookup table initialization functions. An uninitialized lookup table has both color systems set to EColorSystem_NoColor.

EColorLookup::GetColorSystemOut

Output color system.

Namespace: Euresys::Open_eVision

```
[C++]

Euresys::Open_eVision::EColorSystem GetColorSystemOut()
```

Remarks

The EColorLookup objects keep track of the color system transformation, for consistency. When applying a transformation, the source image color system (usually EColorSystem_Rgb) must match the *input color system*; the destination image will be automatically be typed with the *output color system*. In case of a mismatch, an error message is issued. These two values are set by the lookup table initialization functions. An uninitialized lookup table has both color systems set to EColorSystem_NoColor.

EColorLookup::ConvertFromRgb

Sets a color transformation from the Rgb representation to another system, as defined by EColorSystem.

Namespace: Euresys::Open_eVision

```
[C++]

void ConvertFromRgb(
 Euresys::Open_eVision::EColorSystem colorSystem
 )
```

Parameters

colorSystem

Color system, as defined by EColorSystem.

Remarks

The input and output color systems are respectively EColorSystem_Rgb and colorSystem. To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as EColorLookup::Transform.

EColorLookup::ConvertToRgb

Sets a color transformation from any color system, as defined by EColorSystem, to the Rgb representation.

Namespace: Euresys::Open_eVision

```
[C++]
void ConvertToRgb(
 Euresys::Open_eVision::EColorSystem colorSystem
 )
```

Parameters

colorSystem

Color system, as defined by EColorSystem.

Remarks

The input and output color systems are respectively colorSystem and EColorSystem_Rgb. To apply some transform to a color image, you initialize a color lookup once for all and use it at will in a transformation operation such as EColorLookup::Transform.

EColorLookup::EColorLookup

Constructs a color lookup table.

Namespace: Euresys::Open_eVision

```
[C++]
void EColorLookup(
 )
```

EColorLookup::GetIndexBits

EColorLookup::SetIndexBits

Number of bits used for indexing the lookup table.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetIndexBits()
void SetIndexBits(OEV_UINT32 un32IndexBits)
```

Remarks

Before filling in a lookup table, it is necessary to decide how many table entries it requires. The EColorLookup::IndexBits property indicates how many (high-order) bits of the input components are used. The relation between EColorLookup::IndexBits, the number of table entries and the corresponding table size are given below:

IndexBits	Number of entries	Table size (bytes)
4	$2^{(3x4)} = 4096$	14739
5	$2^{(3x5)} = 32768$	107811
6	$2^{(3x6)} = 262144$	823875

The larger the number of entries, the more accuracy is obtained. After EColorLookup::IndexBits has been changed, the lookup table needs to be recomputed.

Note. Be aware that each time a color lookup table is filled, all the entries are recomputed. When EColorLookup::IndexBits equals 6, this may take a very long time. Such large lookup tables should be computed once only. Different combinations of EColorLookup::IndexBits and Interpolation provide a trade-off between accuracy and speed for the table pre-computation and table use.

EColorLookup::GetInterpolation

EColorLookup::SetInterpolation

Interpolation mode.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetInterpolation()

void SetInterpolation(bool bInterpolation)
```

Remarks

When applying a lookup table to transform pixel values, tri-linear interpolation can be used: * when interpolation is not used, the table is looked up at the entry closest to the pixel value. This gives an accuracy equal to the value of the IndexBits property. On the other hand, table lookup is very fast; * when interpolation is used, the table is looked up at eight neighboring entries and an adequate average is computed. This gives full accuracy (8 bits) if the transformation is smooth enough. On the other hand, table lookup is slower.

Note. The interpolation mode may be modified at any time without the need to reinitialize the lookup table.

EColorLookup::Transform

Transforms a quantized color image/pixel to another quantized color image/pixel, using the previously initialized color lookup table.

Namespace: Euresys::Open_eVision

[C++]

```
void Transform(
 EC24 sourceImageColor,
 EC24& destinationImageColor
 )

void Transform(
 const EROIC24* sourceImage,
 EROIC24* destinationImage
 )
```

Parameters

sourceImageColor

Input color image.

destinationImageColor

Output color image.

sourceImage

Input color image.

destinationImage

Output color image.

EColorLookup::WhiteBalance

Initializes a color lookup table that can be used for color adjustment, including a global gain, gamma pre-compensation [cancellation] and white balancing.

Namespace: Euresys::Open_eVision

[C++]

```
void WhiteBalance(
 float gain,
 float gamma,
 float balanceRed,
 float balanceGreen,
 float balanceBlue
 )
```

Parameters

gain

Global gain to be applied to all three color components. By default, the image intensity remains unchanged.

gamma

Gamma exponent. Setting this parameter will cancel the gamma pre-compensation feature of the camera, or apply it. By default, the no gamma pre-compensation is assumed (linear response). The gamma exponent can be chosen among the predefined values EasyColor::CompensateNtscGamma /EasyColor::CompensatePalGamma/EasyColor::CompensateSmpteGamma (pre-compensation) or EasyColor::NtscGamma/EasyColor::PalGamma/EasyColor::SmpteGamma (pre-compensation cancellation), or be user-defined.

balanceRed

Color values to be used for white balance. These parameters should be set to the measured average values of a white (or gray) pixels area, allowing the white balance to adjust pre-component gains appropriately. Use function EasyImage::PixelAverage to obtain them. By default, no white balancing is performed.

balanceGreen

Color values to be used for white balance. These parameters should be set to the measured average values of a white (or gray) pixels area, allowing the white balance to adjust pre-component gains appropriately. Use function EasyImage::PixelAverage to obtain them. By default, no white balancing is performed.

balanceBlue

Color values to be used for white balance. These parameters should be set to the measured average values of a white (or gray) pixels area, allowing the white balance to adjust pre-component gains appropriately. Use function EasyImage::PixelAverage to obtain them. By default, no white balancing is performed.

Remarks

To apply some transform to a color image, you initialize the color lookup once for all and use it at will with EColorLookup::Transform or EasyColor::TransformBayer operation.

4.66. EColorRangeThresholdSegmenter Class

Segments an image using a double threshold on a color image.

Remarks

This segmenter is applicable to EROIC24 RGB color images. It produces coded images with two layers: The White layer (usually, with index 1) contains the unmasked pixels that belong to the cube of the RGB space that spans the low threshold point to the high threshold point; and the Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class:ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetHighThreshold	Value of the high threshold.

GetLowThreshold	Value of the low threshold.
Load	Load the EColorRangeThresholdSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EColorRangeThresholdSegmenter configuration. The given ESerializer must have been created for writing.
SetHighThreshold	Value of the high threshold.
SetLowThreshold	Value of the low threshold.

EColorRangeThresholdSegmenter::GetHighThreshold

EColorRangeThresholdSegmenter::SetHighThreshold

Value of the high threshold.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
EC24 GetHighThreshold() const

void SetHighThreshold(EC24 threshold)
```

EColorRangeThresholdSegmenter::Load

Load the EColorRangeThresholdSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EColorRangeThresholdSegmenter::GetLowThreshold

EColorRangeThresholdSegmenter::SetLowThreshold

Value of the low threshold.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
EC24 GetLowThreshold() const
void SetLowThreshold(EC24 threshold)
```

EColorRangeThresholdSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool operator==(
 const EColorRangeThresholdSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to.

EColorRangeThresholdSegmenter::Save

Save the EColorRangeThresholdSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.67. EColorSingleThresholdSegmenter Class

Segments an image using a single threshold on a color image.

Remarks

This segmenter is applicable to EROIC24 RGB color images. It produces coded images with two layers: The White layer (usually, with index 1) contains the unmasked pixels that belong to the cube of the RGB space defined by the threshold point and the white point (255,255,255); and the Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class:ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetThreshold	Value of the threshold.
Load	Load the EColorSingleThresholdSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EColorSingleThresholdSegmenter configuration. The given ESerializer must have been created for writing.
SetThreshold	Value of the threshold.

EColorSingleThresholdSegmenter::Load

Load the EColorSingleThresholdSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EColorSingleThresholdSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool operator==(
 const EColorSingleThresholdSegmenter& other
 )
```

Parameters

other
Other segmenter to compare to.

EColorSingleThresholdSegmenter::Save

Save the EColorSingleThresholdSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EColorSingleThresholdSegmenter::GetThreshold

EColorSingleThresholdSegmenter::SetThreshold

Value of the threshold.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
EC24 GetThreshold() const
void SetThreshold(EC24 threshold)
```

4.68. EColorVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EColorVector member, and then add elements one at time at the tail by calling the EColorVector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EColorVector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
EColorVector	Constructs a vector.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EColorVector object into the current EColorVector object
SetElement	Modifies the vector element at the given index by the given value.

EColorVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EColor element
 )
```

Parameters

element

The element to be added.

EColorVector::EColorVector

Constructs a vector.

Namespace: Euresys::Open_eVision

```
[C++]
void EColorVector(
 )

void EColorVector(
 OEV_UINT32 un32MaxElements
 )

void EColorVector(
 const EColorVector& other
 )
```

Parameters

un32MaxElements

-

other

EColorVector object to be copied

EColorVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
EColor GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EColorVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EColorVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EColor& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EColorVector (excluded) of the element to be accessed.

EColorVector::operator=

Copies all the data from another EColorVector object into the current EColorVector object

Namespace: Euresys::Open_eVision

[C++]

```
EColorVector& operator=(
 const EColorVector& other
 )
```

Parameters

other

EColorVector object to be copied

EColorVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EColorVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EColor value
 )
```

Parameters

index

Index, between 0 and EColorVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.69. EConverter Class

Conversion functions between bit depth formats of various 3D classes.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	Converts EDepthMap or EZMap between various formats.

EConverter::Convert

Converts EDepthMap or EZMap between various formats.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Convert(
 const EDepthMap8& DepthMapIn,
 EDepthMap16& DepthMapOut,
 Euresys::Open_eVision::Easy3D::EMapConversionMode ConversionMode
 )
```

```
void Convert(
 const EDepthMap8& DepthMapIn,
 EDepthMap32f& DepthMapOut
 )

void Convert(
 const EDepthMap16& DepthMapIn,
 EDepthMap8& DepthMapOut,
 Euresys::Open_eVision::Easy3D::EMapConversionMode ConversionMode
 )

void Convert(
 const EDepthMap16& DepthMapIn,
 EDepthMap32f& DepthMapOut
 )

void Convert(
 const EDepthMap32f& DepthMapIn,
 EDepthMap8& DepthMapOut
 )

void Convert(
 const EDepthMap32f& DepthMapIn,
 EDepthMap16& DepthMapOut
 )

void Convert(
 const EZMap8& ZMapIn,
 EZMap16& ZMapOut,
 Euresys::Open_eVision::Easy3D::EMapConversionMode ConversionMode
 )

void Convert(
 const EZMap8& ZMapIn,
 EZMap32f& ZMapOut
 )

void Convert(
 const EZMap16& ZMapIn,
 EZMap8& ZMapOut,
 Euresys::Open_eVision::Easy3D::EMapConversionMode ConversionMode
 )

void Convert(
 const EZMap16& ZMapIn,
 EZMap32f& ZMapOut
 )

void Convert(
 const EZMap32f& ZMapIn,
 EZMap8& ZMapOut
 )

void Convert(
 const EZMap32f& MapIn,
 EZMap16& MapOut
 )
```

Parameters

DepthMapIn

The input DepthMap (8, 16 or 32 bits)

DepthMapOut

The output DepthMap (8, 16 or 32 bits)

ConversionMode

The conversion mode from EMapConversionMode defines how to transform the pixel value from a format (8, 16 or 32 bits) to another.
EMapConversionMode_MaxDynamic maximizes the used range.
EMapConversionMode_Shift converts by bit shifting.
By default, the mode EMapConversionMode_MaxDynamic is selected.

ZMapIn

The input ZMap (8, 16 or 32 bits)

ZMapOut

The output ZMap (8, 16 or 32 bits)

MapIn

-

MapOut

-

Remarks

Conversion from or to 32bits only supports EMapConversionMode_MaxDynamic mode. The undefined values are converted to the new map undefined value. For 8 and 16 bits images, the minimum defined value is 1, so the conversion of 32 bits images to 8 or 16 bits images adapts the dynamic range between 1 and the maximum value. For conversion to 32 bits float image, the used dynamic range is between 0 and FLOATMAX.

4.70. EDataAugmentation Class

An EDataAugmentation object is responsible for storing the data augmentation parameters and generating new transformed images from existing ones. Deep learning algorithms are not invariant to the location, scale, or rotation of the elements of interest in the image. Thus, if the final application requires the deep learning algorithm to be invariant to those characteristics, the dataset must contain images covering the spectrum of those characteristics. Data augmentation can be used to avoid capturing in the original dataset the whole spectrum of those characteristics by automatically generating new versions of the images in the dataset that are shifted, scaled or rotated to cover this spectrum. When generating a new image, the EDataAugmentation will randomly pick transformation value within the specified limits. When using data augmentation, one must carefully chose the spectrum of transformations such that the element of interest are still visible in the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

CopyTo	Copies itself to the EDataAugmentation object other.
EDataAugmentation	Creates an EDataAugmentation object.

Generate	Generates a new image from a generic EBaseROI image. The caller is responsible for calling freeing the memory of the returned image by calling delete on the image.
GetEnableHorizontalFlip	Sets whether to use horizontally flipped versions of input images.
GetEnableVerticalFlip	Sets whether to use vertically flipped versions of input images.
GetGaussianNoiseMaximumStandardDeviation	The Gaussian noise maximum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EDataAugmentation and EDataAugmentation.(also called the normal distribution). Its value must be superior or equal to EDataAugmentation.
GetGaussianNoiseMinimumStandardDeviation	The Gaussian noise minimum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EDataAugmentation and EDataAugmentation.(also called the normal distribution). Its value must be betwteen 0 and EDataAugmentation.
GetMaxBrightnessOffset	Sets the maximum absolute brightness offset. It must be between 0 and 1.
GetMaxContrastGain	Sets the maximum contrast gain. Its value must be strictly positive and over EDataAugmentation::MinContrastGain.
GetMaxGamma	Sets the maximum gamma for gamma correction. Its value must be higher than EDataAugmentation::MinGamma.
GetMaxHorizontalShear	Sets the maximum absolute horizontal shear, represented as an angle from the vertical direction. Its value must be between 0 and 90 degrees.
GetMaxHorizontalShift	Sets the maximum horizontal shift allowed.
GetMaxHueOffset	Sets the maximum absolute hue offset. Its value must be between 0 and 180 degrees.
GetMaxRotationAngle	Set the maximum rotation angle allowed. Its value must be between 0 and 180 degrees.
GetMaxSaturationGain	Sets the maximum saturation gain. Its value must be over or equal to EDataAugmentation::MinSaturationGain.
GetMaxScale	Sets the maximum scaling allowed. Its value must be strictly positive and over EDataAugmentation::MinScale.
GetMaxStainColor	The maximum color value from which to draw a color to fill in the stain (we use a gaussian distribution of with a mean betweeen EDataAugmentation::MinStainColor and EDataAugmentation::MaxStainColor) Its value must be bigger than EDataAugmentation::MinStainColor.
GetMaxVerticalShear	Sets the maximum absolute vertical shear, represented as an angle from the horizontal direction. Its value must be between 0 and 90 degrees.
GetMaxVerticalShift	Sets the maximum vertical shift allowed.
GetMinContrastGain	Sets the minimum contrast gain. Its value must be strictly positive and below EDataAugmentation::MaxContrastGain.

GetMinGamma	Sets the minimum gamma for gamma correction. Its value must be strictly positive and below EDataAugmentation::MaxGamma.
GetMinSaturationGain	Sets the minimum saturation gain. Its value must be strictly positive.
GetMinScale	Sets the minimum scaling allowed. Its value must be strictly positive and below EDataAugmentation::MaxScale.
GetMinStainColor	The minimum color value from which to draw a color to fill in the stain (we use a gaussian distribution of with a mean betweeen EDataAugmentation::MinStainColor and EDataAugmentation::MaxStainColor) Its value must be inferior to 255 and EDataAugmentation::MaxStainColor.
GetSaltAndPepperNoiseMaximumDensity	The maximum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EDataAugmentation and EDataAugmentation) pixels to its minimum or maximum value. Its value must be betwteen EDataAugmentation and 1.
GetSaltAndPepperNoiseMinimumDensity	The minimum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EDataAugmentation and EDataAugmentation) pixels to its minimum or maximum value. Its value must be betwteen 0 and EDataAugmentation.
GetSpeckleNoiseMaximumStandardDeviation	The speckle noise maximum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution of deviation between EDataAugmentation and EDataAugmentation. Its value must be strictly higher than EDataAugmentation.
GetSpeckleNoiseMinimumStandardDeviation	The speckle noise minimum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution of deviation between EDataAugmentation and EDataAugmentation. Its value must be strictly positive and lower than EDataAugmentation.
GetStainBlur	Stain blur. The stain blur is the half kernel size of a Gaussian filter that is applied on the stain to smooth its edges and therefore the transition between the original image and the stain.
GetStainColorVariation	The variation of the color used to fill in the ellipse (We use an gaussian distribution of standart mean deviation stainColorVariation).
GetStainDisruptionMaxAnchorPoints	The maximum number of anchor points to use while disrupting the ellipse.
GetStainDisruptionMaxIntensity	The maximun offset applied to the coordinates (x,y) of a pixel from the ellipse contour.
GetStainEllipseMaxRadius	The maximum radius of the ellipse used to produce the stain. Its value must be superior to EDataAugmentation::StainEllipseMinRadius.
GetStainEllipseMinRadius	The minimum radius of the ellipse used to produce the stain. Its value must be inferior to EDataAugmentation::StainEllipseMaxRadius.

GetStainInterpolationSiteFactor	The factor to compute the number of interpolation sites to use from the number of anchor points.
GetStainProbability	Probability of applying the stain data augmentation (default value: 0, i.e. no stain).
HasDataAugmentations	Whether the EDataAugmentation object contains augmentations.
Load	Loads an EDataAugmentation object. The given ESerializer must have been created for reading.
operator=	Copies the EDataAugmentation object other to this object.
operator==	Equality operator.
Save	Saves an EDataAugmentation object. The given ESerializer must have been created for writing.
SetEnableHorizontalFlip	Sets whether to use horizontally flipped versions of input images.
SetEnableVerticalFlip	Sets whether to use vertically flipped versions of input images.
SetGaussianNoiseMaximumStandardDeviation	The Gaussian noise maximum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EDataAugmentation and EDataAugmentation.(also called the normal distribution). Its value must be superior or equal to EDataAugmentation.
SetGaussianNoiseMinimumStandardDeviation	The Gaussian noise minimum standard deviation. The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EDataAugmentation and EDataAugmentation.(also called the normal distribution). Its value must be betwteen 0 and EDataAugmentation.
SetMaxBrightnessOffset	Sets the maximum absolute brightness offset. It must be between 0 and 1.
SetMaxContrastGain	Sets the maximum contrast gain. Its value must be strictly positive and over EDataAugmentation::MinContrastGain.
SetMaxGamma	Sets the maximum gamma for gamma correction. Its value must be higher than EDataAugmentation::MinGamma.
SetMaxHorizontalShear	Sets the maximum absolute horizontal shear, represented as an angle from the vertical direction. Its value must be between 0 and 90 degrees.
SetMaxHorizontalShift	Sets the maximum horizontal shift allowed.
SetMaxHueOffset	Sets the maximum absolute hue offset. Its value must be between 0 and 180 degrees.
SetMaxRotationAngle	Set the maximum rotation angle allowed. Its value must be between 0 and 180 degrees.
SetMaxSaturationGain	Sets the maximum saturation gain. Its value must be over or equal to EDataAugmentation::MinSaturationGain.
SetMaxScale	Sets the maximum scaling allowed. Its value must be strictly positive and over EDataAugmentation::MinScale.

SetMaxStainColor	The maximum color value from which to draw a color to fill in the stain (we use a gaussian distribution of with a mean betweeen EDataAugmentation::MinStainColor and EDataAugmentation::MaxStainColor) Its value must be bigger than EDataAugmentation::MinStainColor.
SetMaxVerticalShear	Sets the maximum absolute vertical shear, represented as an angle from the horizontal direction. Its value must be between 0 and 90 degrees.
SetMaxVerticalShift	Sets the maximum vertical shift allowed.
SetMinContrastGain	Sets the minimum contrast gain. Its value must be strictly positive and below EDataAugmentation::MaxContrastGain.
SetMinGamma	Sets the minimum gamma for gamma correction. Its value must be strictly positive and below EDataAugmentation::MaxGamma.
SetMinSaturationGain	Sets the minimum saturation gain. Its value must be strictly positive.
SetMinScale	Sets the minimum scaling allowed. Its value must be strictly positive and below EDataAugmentation::MaxScale.
SetMinStainColor	The minimum color value from which to draw a color to fill in the stain (we use a gaussian distribution of with a mean betweeen EDataAugmentation::MinStainColor and EDataAugmentation::MaxStainColor) Its value must be inferior to 255 and EDataAugmentation::MaxStainColor.
SetSaltAndPepperNoiseMaximumDensity	The maximum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EDataAugmentation and EDataAugmentation) pixels to its minimum or maximum value. Its value must be betwteen EDataAugmentation and 1.
SetSaltAndPepperNoiseMinimumDensity	The minimum density of the salt and pepper noise. The salt and pepper noise sets the value of a number of randomly selected (between EDataAugmentation and EDataAugmentation) pixels to its minimum or maximum value. Its value must be betwteen 0 and EDataAugmentation.
SetSpeckleNoiseMaximumStandardDeviation	The speckle noise maximum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution of deviation between EDataAugmentation and EDataAugmentation. Its value must be strictly higher than EDataAugmentation.
SetSpeckleNoiseMinimumStandardDeviation	The speckle noise minimum standard deviation. The speckle noise is a multiplicative noise sampled from a Gamma distribution of deviation between EDataAugmentation and EDataAugmentation. Its value must be strictly positive and lower than EDataAugmentation.
SetStainBlur	Stain blur. The stain blur is the half kernel size of a Gaussian filter that is applied on the stain to smooth its edges and therefore the transition between the original image and the stain.

SetStainColorVariation	The variation of the color used to fill in the ellipse (We use an gaussian distribution of standart mean deviation stainColorVariation).
SetStainDisruptionMaxAnchorPoints	The maximum number of anchor points to use while disrupting the ellipse.
SetStainDisruptionMaxIntensity	The maximun offset applied to the coordinates (x,y) of a pixel from the ellipse contour.
SetStainEllipseMaxRadius	The maximum radius of the ellipse used to produce the stain. Its value must be superior to EDataAugmentation::StainEllipseMinRadius.
SetStainEllipseMinRadius	The minimum radius of the ellipse used to produce the stain. Its value must be inferior to EDataAugmentation::StainEllipseMaxRadius.
SetStainInterpolationSiteFactor	The factor to compute the number of interpolation sites to use from the number of anchor points.
SetStainProbability	Probability of applying the stain data augmentation (default value: 0, i.e. no stain).

EDataAugmentation::CopyTo

Copies itself to the EDataAugmentation object other.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void CopyTo(
 EDataAugmentation& other
 )
```

Parameters

other

-

EDataAugmentation::EDataAugmentation

Creates an EDataAugmentation object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EDataAugmentation(
 )

void EDataAugmentation(
 const EDataAugmentation& other
 )
```

Parameters

other

-

EDataAugmentation::GetEnableHorizontalFlip

EDataAugmentation::SetEnableHorizontalFlip

Sets whether to use horizontally flipped versions of input images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableHorizontalFlip() const
void SetEnableHorizontalFlip(bool enable)
```

EDataAugmentation::GetEnableVerticalFlip

EDataAugmentation::SetEnableVerticalFlip

Sets whether to use vertically flipped versions of input images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableVerticalFlip() const
void SetEnableVerticalFlip(bool enable)
```

EDataAugmentation::GetGaussianNoiseMaximumStandardDeviation

EDataAugmentation::SetGaussianNoiseMaximumStandardDeviation

The Gaussian noise maximum standard deviation.
The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EDataAugmentation and EDataAugmentation.(also called the normal distribution).
Its value must be superior or equal to EDataAugmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetGaussianNoiseMaximumStandardDeviation() const
void SetGaussianNoiseMaximumStandardDeviation(float gaussianMaximumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EDataAugmentation::GetGaussianNoiseMinimumStandardDeviation

EDataAugmentation::SetGaussianNoiseMinimumStandardDeviation

The Gaussian noise minimum standard deviation.
The Gaussian noise is an additive noise sampled from a Gaussian distribution of deviation between EDataAugmentation and EDataAugmentation.(also called the normal distribution).
Its value must be betwteen 0 and EDataAugmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetGaussianNoiseMinimumStandardDeviation() const

void SetGaussianNoiseMinimumStandardDeviation(float gaussianMinimumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EDataAugmentation::Generate

Generates a new image from a generic EBaseROI image. The caller is responsible for calling freeing the memory of the returned image by calling delete on the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EBaseROI* Generate(
 const EBaseROI* img,
 Euresys::Open_eVision::EasyDeepLearning::EDLDataAugmentationType type
 )

void Generate(
 const EROIBW8& source,
 EROIBW8& dest,
 Euresys::Open_eVision::EasyDeepLearning::EDLDataAugmentationType type
 )

void Generate(
 const EROIBW16& source,
 EROIBW16& dest,
 Euresys::Open_eVision::EasyDeepLearning::EDLDataAugmentationType type
 )
```

```
void Generate(
 const EROIC24& source,
 EROIC24& dest,
 Euresys::Open_eVision::EasyDeepLearning::EDLDataAugmentationType type
 )
```

Parameters

img
The image to transform.

type
The type of transformation to generate.

source
The image to transform.

dest
The transformed image.

Remarks

If the data augmentation fails, the method will throw an exception.

EDataAugmentation::HasDataAugmentations

Whether the EDataAugmentation object contains augmentations.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasDataAugmentations(
 )
```

EDataAugmentation::Load

Loads an EDataAugmentation object. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EDataAugmentation::GetMaxBrightnessOffset

EDataAugmentation::SetMaxBrightnessOffset

Sets the maximum absolute brightness offset. It must be between 0 and 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxBrightnessOffset() const
void SetMaxBrightnessOffset(float offset)
```

Remarks

Brightness transformation is performed by adding a value taken between -EDataAugmentation::MaxBrightnessOffset and +EDataAugmentation::MaxBrightnessOffset to each pixel of the normalized image.

EDataAugmentation::GetMaxContrastGain

EDataAugmentation::SetMaxContrastGain

Sets the maximum contrast gain. Its value must be strictly positive and over EDataAugmentation::MinContrastGain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxContrastGain() const
void SetMaxContrastGain(float val)
```

Remarks

Contrast transformation is performed by multiplying each mean-centered pixel value by a gain value taken between EDataAugmentation::MinContrastGain and EDataAugmentation::MaxContrastGain. A contrast transformation does not change the overall brightness of an image.

EDataAugmentation::GetMaxGamma

EDataAugmentation::SetMaxGamma

Sets the maximum gamma for gamma correction. Its value must be higher than EDataAugmentation::MinGamma.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxGamma() const
void SetMaxGamma(float gamma)
```

Remarks

Gamma correction transformation is performed by raising each normalized pixel value to the power of gamma with gamma taken between EDataAugmentation::MinGamma and EDataAugmentation::MaxGamma.

EDataAugmentation::GetMaxHorizontalShear

EDataAugmentation::SetMaxHorizontalShear

Sets the maximum absolute horizontal shear, represented as an angle from the vertical direction. Its value must be between 0 and 90 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxHorizontalShear() const
void SetMaxHorizontalShear(float hShear)
```

EDataAugmentation::GetMaxHorizontalShift

EDataAugmentation::SetMaxHorizontalShift

Sets the maximum horizontal shift allowed.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetMaxHorizontalShift() const
void SetMaxHorizontalShift(int maxShift)
```

EDataAugmentation::GetMaxHueOffset

EDataAugmentation::SetMaxHueOffset

Sets the maximum absolute hue offset. Its value must be between 0 and 180 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxHueOffset() const
void SetMaxHueOffset(float hueOffset)
```

Remarks

The hue is represented as an angle between 0 and 360 degrees. The hue transformation is performed by rotating the hue of each pixel by a value between -EDataAugmentation::MaxHueOffset and +EDataAugmentation::MaxHueOffset. This transformation only works for color images.

EDataAugmentation::GetMaxRotationAngle

EDataAugmentation::SetMaxRotationAngle

Set the maximum rotation angle allowed. Its value must be between 0 and 180 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxRotationAngle() const
void SetMaxRotationAngle(float maxAngle)
```

EDataAugmentation::GetMaxSaturationGain

EDataAugmentation::SetMaxSaturationGain

Sets the maximum saturation gain. Its value must be over or equal to EDataAugmentation::MinSaturationGain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxSaturationGain() const
void SetMaxSaturationGain(float saturationGain)
```

Remarks

The saturation transformation is performed by multiplying the saturation of each pixel by a value between EDataAugmentation::MinSaturationGain and EDataAugmentation::MaxSaturationGain.

EDataAugmentation::GetMaxScale

EDataAugmentation::SetMaxScale

Sets the maximum scaling allowed. Its value must be strictly positive and over EDataAugmentation::MinScale.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxScale() const
void SetMaxScale(float maxScale)
```

EDataAugmentation::GetMaxStainColor

EDataAugmentation::SetMaxStainColor

The maximum color value from which to draw a color to fill in the stain (we use a gaussian distribution of with a mean betweeen EDataAugmentation::MinStainColor and EDataAugmentation::MaxStainColor)
Its value must be bigger than EDataAugmentation::MinStainColor.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT8 GetMaxStainColor()
void SetMaxStainColor(OEV_UINT8 maxStainColor)
```

EDataAugmentation::GetMaxVerticalShear

EDataAugmentation::SetMaxVerticalShear

Sets the maximum absolute vertical shear, represented as an angle from the horizontal direction. Its value must be between 0 and 90 degrees.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxVerticalShear() const
void SetMaxVerticalShear(float vShear)
```

EDataAugmentation::GetMaxVerticalShift

EDataAugmentation::SetMaxVerticalShift

Sets the maximum vertical shift allowed.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetMaxVerticalShift() const
void SetMaxVerticalShift(int maxShift)
```

EDataAugmentation::GetMinContrastGain

EDataAugmentation::SetMinContrastGain

Sets the minimum contrast gain. Its value must be strictly positive and below EDataAugmentation::MaxContrastGain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinContrastGain() const
void SetMinContrastGain(float val)
```

Remarks

Contrast transformation is performed by multiplying each mean-centered pixel value by a gain value taken between EDataAugmentation::MinContrastGain and EDataAugmentation::MaxContrastGain. A contrast transformation does not change the overall brightness of an image.

EDataAugmentation::GetMinGamma

EDataAugmentation::SetMinGamma

Sets the minimum gamma for gamma correction. Its value must be strictly positive and below EDataAugmentation::MaxGamma.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinGamma() const
void SetMinGamma(float gamma)
```

Remarks

Gamma correction transformation is performed by raising each normalized pixel value to the power of gamma with gamma taken between EDataAugmentation::MinGamma and EDataAugmentation::MaxGamma.

EDataAugmentation::GetMinSaturationGain

EDataAugmentation::SetMinSaturationGain

Sets the minimum saturation gain. Its value must be strictly positive.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinSaturationGain() const
void SetMinSaturationGain(float saturationGain)
```

Remarks

The saturation transformation is performed by multiplying the saturation of each pixel by a value between EDataAugmentation::MinSaturationGain and EDataAugmentation::MaxSaturationGain.

EDataAugmentation::GetMinScale

EDataAugmentation::SetMinScale

Sets the minimum scaling allowed. Its value must be strictly positive and below EDataAugmentation::MaxScale.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinScale() const
void SetMinScale(float minScale)
```

EDataAugmentation::GetMinStainColor

EDataAugmentation::SetMinStainColor

The minimum color value from which to draw a color to fill in the stain (we use a gaussian distribution of with a mean betweeen EDataAugmentation::MinStainColor and EDataAugmentation::MaxStainColor)
Its value must be inferior to 255 and EDataAugmentation::MaxStainColor.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT8 GetMinStainColor()
void SetMinStainColor(OEV_UINT8 minStainColor)
```

EDataAugmentation::operator=

Copies the EDataAugmentation object other to this object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDataAugmentation& operator=(
 const EDataAugmentation& other
 )
```

Parameters

other

-

EDataAugmentation::operator==

Equality operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool operator==(
 const EDataAugmentation& other
 )
```

Parameters

other

Other object to compare to

EDataAugmentation::GetSaltAndPepperNoiseMaximumDensity

EDataAugmentation::SetSaltAndPepperNoiseMaximumDensity

The maximum density of the salt and pepper noise.
The salt and pepper noise sets the value of a number of randomly selected (between EDataAugmentation and EDataAugmentation) pixels to its minimum or maximum value.
Its value must be betwteen EDataAugmentation and 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetSaltAndPepperNoiseMaximumDensity() const
void SetSaltAndPepperNoiseMaximumDensity(float saltAndPepperMaximumDensity)
```

Remarks

This noise is computed after all the other noises.

EDataAugmentation::GetSaltAndPepperNoiseMinimumDensity

EDataAugmentation::SetSaltAndPepperNoiseMinimumDensity

The minimum density of the salt and pepper noise.
The salt and pepper noise sets the value of a number of randomly selected (between EDataAugmentation and EDataAugmentation) pixels to its minimum or maximum value.
Its value must be betwteen 0 and EDataAugmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetSaltAndPepperNoiseMinimumDensity() const
void SetSaltAndPepperNoiseMinimumDensity(float saltAndPepperMinimumDensity)
```

Remarks

This noise is computed after all the other noises.

EDataAugmentation::Save

Saves an EDataAugmentation object. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EDataAugmentation::GetSpeckleNoiseMaximumStandardDeviation

EDataAugmentation::SetSpeckleNoiseMaximumStandardDeviation

The speckle noise maximum standard deviation.
The speckle noise is a multiplicative noise sampled from a Gamma distribution of deviation between EDataAugmentation and EDataAugmentation.
Its value must be strictly higher than EDataAugmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSpeckleNoiseMaximumStandardDeviation() const
void SetSpeckleNoiseMaximumStandardDeviation(float speckleMaximumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EDataAugmentation::GetSpeckleNoiseMinimumStandardDeviation

EDataAugmentation::SetSpeckleNoiseMinimumStandardDeviation

The speckle noise minimum standard deviation.
The speckle noise is a multiplicative noise sampled from a Gamma distribution of deviation between EDataAugmentation and EDataAugmentation.
Its value must be strictly positive and lower than EDataAugmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSpeckleNoiseMinimumStandardDeviation() const
void SetSpeckleNoiseMinimumStandardDeviation(float speckleMinimumDeviation)
```

Remarks

This noise is computed before the salt and paper noise.

EDataAugmentation::GetStainBlur

EDataAugmentation::SetStainBlur

Stain blur.
The stain blur is the half kernel size of a Gaussian filter that is applied on the stain to smooth its edges and therefore the transition between the original image and the stain.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetStainBlur() const
void SetStainBlur(int blur)
```

EDataAugmentation::GetStainColorVariation

EDataAugmentation::SetStainColorVariation

The variation of the color used to fill in the ellipse (We use an gaussian distribution of standart mean deviation stainColorVariation).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetStainColorVariation()
void SetStainColorVariation(OEV_UINT32 stainColorVariation)
```

EDataAugmentation::GetStainDisruptionMaxAnchorPoints

EDataAugmentation::SetStainDisruptionMaxAnchorPoints

The maximum number of anchor points to use while disrupting the ellipse.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetStainDisruptionMaxAnchorPoints()
void SetStainDisruptionMaxAnchorPoints(OEV_UINT32 stainDisruptioMaxAnchorPoints)
```

EDataAugmentation::GetStainDisruptionMaxIntensity

EDataAugmentation::SetStainDisruptionMaxIntensity

The maximun offset applied to the coordinates (x,y) of a pixel from the ellipse contour.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetStainDisruptionMaxIntensity()
void SetStainDisruptionMaxIntensity(float stainDisruptionMaxIntensity)
```

EDataAugmentation::GetStainEllipseMaxRadius

EDataAugmentation::SetStainEllipseMaxRadius

The maximum radius of the ellipse used to produce the stain.
Its value must be superior to EDataAugmentation::StainEllipseMinRadius.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetStainEllipseMaxRadius()
void SetStainEllipseMaxRadius(OEV_UINT32 stainEllipseMaxRadius)
```

EDataAugmentation::GetStainEllipseMinRadius

EDataAugmentation::SetStainEllipseMinRadius

The minimum radius of the ellipse used to produce the stain.
Its value must be inferior to EDataAugmentation::StainEllipseMaxRadius.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetStainEllipseMinRadius()
void SetStainEllipseMinRadius(OEV_UINT32 stainEllipseMinRadius)
```

EDataAugmentation::GetStainInterpolationSiteFactor

EDataAugmentation::SetStainInterpolationSiteFactor

The factor to compute the number of interpolation sites to use from the number of anchor points.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetStainInterpolationSiteFactor()
void SetStainInterpolationSiteFactor(float interpolationSiteFactor)
```

EDataAugmentation::GetStainProbability

EDataAugmentation::SetStainProbability

Probability of applying the stain data augmentation (default value: 0, i.e. no stain).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetStainProbability()
void SetStainProbability(float stainProbability)
```

4.71. EDatasetSplit Class

The EDatasetSplit maps the image indexes of a EClassificationDataset to a dataset split type (EDatasetType).
The image indexes mapped in the object are consecutive and between 0 and EDatasetSplit::GetNumImages - 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

CopyTo	Copy the dataset split to the other instance.
EDatasetSplit	Creates a EDatasetSplit.
GetDatasetMapping	Mapping between image indexes and the corresponding dataset split type. The value at index i of the vector is the dataset split type for the image index i.
GetImageType	Dataset type for the given image index.

GetImageTypeLocked	Whether the image type is locked for the given image. This property is mainly used to lock training and validation images after the split was used in a training.
GetNumImages	Number of image indexes associated in the object.
GetNumLockedImages	Number of images for which the dataset type is locked.
GetSplit	Image indexes corresponding to the given type.
HasLockedImages	Whether the split contains images for which the dataset type is locked.
Load	Load the EDatasetSplit configuration. The given ESerializer must have been created for reading.
operator=	Assigment operator.
Save	Save the EDatasetSplit configuration. The given ESerializer must have been created for writing.
SetImageType	Dataset type for the given image index.
SetImageTypeLocked	Whether the image type is locked for the given image. This property is mainly used to lock training and validation images after the split was used in a training.
SetTypeLocked	Locks or unlocks all the images whose dataset type matches the given type. Note that the given type can be a combination of several dataset type (using the | operator).

EDatasetSplit::CopyTo

Copy the dataset split to the other instance.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void CopyTo(
 EDatasetSplit& other
 )
```

Parameters

other

Other instance of EDatasetSplit

EDatasetSplit::GetDatasetMapping

Mapping between image indexes and the corresponding dataset split type.
The value at index i of the vector is the dataset split type for the image index i.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDatasetType> GetDatasetMapping()
const
```

EDatasetSplit::EDatasetSplit

Creates a EDatasetSplit.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EDatasetSplit(
 )

void EDatasetSplit(
 const std::vector<Euresys::Open_eVision::EasyDeepLearning::EDatasetType>& mapping
 )

void EDatasetSplit(
 const EDatasetSplit& other
 )
```

Parameters

mapping

Vector where the value at index i is the dataset type for image i.

other

Other instance of EDatasetSplit

EDatasetSplit::GetImageType

Dataset type for the given image index.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDatasetType GetImageType(
 int imageIndex
 )
```

Parameters

imageIndex

Image index.

EDatasetSplit::GetImageTypeLocked

Whether the image type is locked for the given image.
This property is mainly used to lock training and validation images after the split was used in a training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetImageTypeLocked(
 int imageIndex
 )
```

Parameters

imageIndex

Image index.

EDatasetSplit::GetNumImages

Number of image indexes associated in the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumImages(
 )

int GetNumImages(
 Euresys::Open_eVision::EasyDeepLearning::EDatasetType type
 )
```

Parameters

type

Dataset split type

EDatasetSplit::GetSplit

Image indexes corresponding to the given type.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<int> GetSplit(
 Euresys::Open_eVision::EasyDeepLearning::EDatasetType type
 )
```

Parameters

type

-

EDatasetSplit::HasLockedImages

Whether the split contains images for which the dataset type is locked.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasLockedImages(
 )
```

EDatasetSplit::Load

Load the EDatasetSplit configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EDatasetSplit::GetNumLockedImages

Number of images for which the dataset type is locked.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumLockedImages() const
```

EDatasetSplit::operator=

Assigment operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDatasetSplit& operator=(
 const EDatasetSplit& other
 )
```

Parameters

other

Other instance of EDatasetSplit

EDatasetSplit::Save

Save the EDatasetSplit configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EDatasetSplit::SetImageType

Dataset type for the given image index.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetImageType(
 int imageIndex,
 Euresys::Open_eVision::EasyDeepLearning::EDatasetType type
 )
```

Parameters

imageIndex

Image index.

type

Dataset split type.

EDatasetSplit::SetImageTypeLocked

Whether the image type is locked for the given image.
This property is mainly used to lock training and validation images after the split was used in a training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetImageTypeLocked(
 int imageIndex,
 bool isLocked
 )
```

Parameters

imageIndex

Image index.

isLocked

Whether to lock the image

EDatasetSplit::SetTypeLocked

Locks or unlocks all the images whose dataset type matches the given type. Note that the given type can be a combination of several dataset type (using the | operator).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetTypeLocked(
 int type,
 bool isLocked
 )
```

Parameters

type

Dataset type to lock.

isLocked

Whether to lock the images

4.72. EDecimator Class

Decimation of a point cloud/ZMap/DepthMap.

Derived Class(es):EGridDecimatorERandomDecimator

Namespace: Euresys::Open_eVision::Easy3D

Methods

Decimate	Decimates a EPointCloud.
EDecimator	Creates an EDecimator object.
Load	Loads the decimator configuration. The given ESerializer must have been created for reading.
operator!=	test inequality between two EDecimator.
operator==	test equality between two EDecimator.
Save	Saves the decimator configuration. The given ESerializer must have been created for writing.

EDecimator::Decimate

Decimates a EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Decimate(
 const EPointCloud& cloudIn,
 EPointCloud& cloudOut
 )
```

Parameters

cloudIn

The input point cloud.

cloudOut

The output point cloud.

EDecimator::EDecimator

Creates an EDecimator object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EDecimator(
 )
```

EDecimator::Load

Loads the decimator configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EDecimator::operator!=

test inequality between two EDecimator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator!=(
 const EDecimator& other
 )
```

Parameters

other
the EDecimator to be compared with

EDecimator::operator==

test equality between two EDecimator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const EDecimator& other
 )
```

Parameters

other

the EDecimator to be compared with

EDecimator::Save

Saves the decimator configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.73. EDeepLearningBenchmark Class

Deep Learning benchmark results.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

EDeepLearningBenchmark	-
GetBenchmarkHeight	Height of the images generated in the benchmark. It depends on EDeepLearningBenchmark::DatasetHeight and EDeepLearningBenchmarkSettings::InternalResizeDisabled).
GetBenchmarkSettings	Settings of the benchmark.

GetBenchmarkWidth	Width of the images generated in the benchmark. It depends on EDeepLearningBenchmark::DatasetWidth and EDeepLearningBenchmarkSettings::InternalResizeDisabled).
GetDatasetHeight	Height of the dataset used to run the benchmark.
GetDatasetWidth	Width of the dataset used to run the benchmark.
GetErrorCode	Error code thrown when computing the benchmark.
GetErrorDescription	Description of the error that occurred during the benchmark.
GetMaxLatency	Maximum latency (time between the acquisition of an image and the computation of its result).
GetMaxTimePerImage	Maximum time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).
GetMaxTimePerInference	Maximum time for a single inference call (in seconds).
GetMeanLatency	Mean latency.
GetMeanTimePerImage	Mean time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).
GetMeanTimePerInference	Mean time for a single inference call (in seconds).
GetMinLatency	Minimum latency (time between the acquisition of an image and the computation of its result).
GetMinTimePerImage	Minimum time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).
GetMinTimePerInference	Minimum time for a single inference call (in seconds).
GetNumDroppedInference	The number of dropped inference call. Dropped inference call happens when simulating a camera framerate and the processing speed isn't fast enough.
GetNumImagesByInference	The number of images that were processed together in one inference call to exploit the specified batch size. For EasyClassify and EasyLocate, this is equal to the batch size. For EasySegment, it depends on the number of patches that are extracted from input images.
GetNumInferences	Number of inference made in the benchmark. It is equal to the number of images divided by EDeepLearningBenchmark.
GetStdTimePerImage	Standard deviation of the time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).
GetStdTimePerInference	Standard deviation of the inference time for a single inference call (in seconds).
GetThroughput	Number of images per second that were processed by the Deep Learning tool.

GetTimePerInference	Inference time (in nanoseconds) for all the inference calls (an inference processes EDeepLearningBenchmark::NumImagesByInference images together). A time of 0 means that the images for this inference call were dropped (not processed).
IsValid	Whether the benchmark is ready and valid.
Load	Load a EDeepLearningBenchmark object. The given ESerializer must have been created for reading.
operator=	-
Save	Save the EDeepLearningBenchmark object. The given ESerializer must have been created for writing.

EDeepLearningBenchmark::GetBenchmarkHeight

Height of the images generated in the benchmark. It depends on EDeepLearningBenchmark::DatasetHeight and EDeepLearningBenchmarkSettings::InternalResizeDisabled).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBenchmarkHeight() const
```

EDeepLearningBenchmark::GetBenchmarkSettings

Settings of the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
const EDeepLearningBenchmarkSettings& GetBenchmarkSettings() const
```

EDeepLearningBenchmark::GetBenchmarkWidth

Width of the images generated in the benchmark. It depends on EDeepLearningBenchmark::DatasetWidth and EDeepLearningBenchmarkSettings::InternalResizeDisabled).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBenchmarkWidth() const
```

EDeepLearningBenchmark::GetDatasetHeight

Height of the dataset used to run the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetDatasetHeight() const
```

EDeepLearningBenchmark::GetDatasetWidth

Width of the dataset used to run the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetDatasetWidth() const
```

EDeepLearningBenchmark::EDeepLearningBenchmark

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EDeepLearningBenchmark(
 )

void EDeepLearningBenchmark(
 const EDeepLearningBenchmark& other
 )

void EDeepLearningBenchmark(
 const EDeepLearningBenchmarkSettings& settings
 )
```

Parameters

other

-

settings

-

EDeepLearningBenchmark::GetErrorCode

Error code thrown when computing the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EError GetErrorCode() const
```

EDeepLearningBenchmark::GetErrorDescription

Description of the error that occurred during the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetErrorDescription() const
```

EDeepLearningBenchmark::IsValid

Whether the benchmark is ready and valid.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsValid(
 )
```

EDeepLearningBenchmark::Load

Load a EDeepLearningBenchmark object. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& filePath
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

filePath

File path.

serializer

Pointer to the ESerializer created for reading.

EDeepLearningBenchmark::GetMaxLatency

Maximum latency (time between the acquisition of an image and the computation of its result).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMaxLatency() const
```

EDeepLearningBenchmark::GetMaxTimePerImage

Maximum time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMaxTimePerImage() const
```

EDeepLearningBenchmark::GetMaxTimePerInference

Maximum time for a single inference call (in seconds).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMaxTimePerInference() const
```

EDeepLearningBenchmark::GetMeanLatency

Mean latency.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMeanLatency() const
```

EDeepLearningBenchmark::GetMeanTimePerImage

Mean time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMeanTimePerImage() const
```

EDeepLearningBenchmark::GetMeanTimePerInference

Mean time for a single inference call (in seconds).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMeanTimePerInference() const
```

EDeepLearningBenchmark::GetMinLatency

Minimum latency (time between the acquisition of an image and the computation of its result).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMinLatency() const
```

EDeepLearningBenchmark::GetMinTimePerImage

Minimum time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMinTimePerImage() const
```

EDeepLearningBenchmark::GetMinTimePerInference

Minimum time for a single inference call (in seconds).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetMinTimePerInference() const
```

EDeepLearningBenchmark::GetNumDroppedInference

The number of dropped inference call.
Dropped inference call happens when simulating a camera framerate and the processing speed isn't fast enough.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumDroppedInference() const
```

EDeepLearningBenchmark::GetNumImagesByInference

The number of images that were processed together in one inference call to exploit the specified batch size.
For EasyClassify and EasyLocate, this is equal to the batch size.
For EasySegment, it depends on the number of patches that are extracted from input images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImagesByInference() const
```

EDeepLearningBenchmark::GetNumInferences

Number of inference made in the benchmark. It is equal to the number of images divided by EDeepLearningBenchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumInferences() const
```

EDeepLearningBenchmark::operator=

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningBenchmark& operator=(
 const EDeepLearningBenchmark& other
 )
```

Parameters

other

-

EDeepLearningBenchmark::Save

Save the EDeepLearningBenchmark object. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& filePath
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

filePath

File path.

serializer

Pointer to the ESerializer created for writing.

EDeepLearningBenchmark::GetStdTimePerImage

Standard deviation of the time in seconds for an image (equal to EDeepLearningBenchmark / EDeepLearningBenchmark::NumImagesByInference).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetStdTimePerImage() const
```

EDeepLearningBenchmark::GetStdTimePerInference

Standard deviation of the inference time for a single inference call (in seconds).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
double GetStdTimePerInference() const
```

EDeepLearningBenchmark::GetThroughput

Number of images per second that were processed by the Deep Learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
double GetThroughput() const
```

EDeepLearningBenchmark::GetTimePerInference

Inference time (in nanoseconds) for all the inference calls (an inference processes EDeepLearningBenchmark::NumImagesByInference images together). A time of 0 means that the images for this inference call were dropped (not processed).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::vector<double> GetTimePerInference() const
```

4.74. EDeepLearningBenchmarkSettings Class

Class representing the settings of a Deep Learning benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

EDeepLearningBenchmarkSettings	-
GetBatchSize	Batch size.
GetCameraSpeed	Camera speed to simulate. If equal to 0, the benchmark will produce images as fast as possible.
GetDevice	Device to use for the benchmark.
GetEngine	Deep Learning engine to use for the benchmark.
GetInferencePrecision	Inference precision to use for the benchmark.
GetInternalResizeDisabled	Whether to disable the internal resize operation performed by the Deep Learning tools. When disabled, the benchmark will generate images at the optimal size for the tool.
GetName	Name of the benchmark.
GetNumExternalThreads	Number of external threads.
GetNumImages	Number of images to use during the benchmark.

GetNumInternalThreads	Number of internal threads.
Load	Loads a EDeepLearningBenchmarkSettings object. The given ESerializer must have been created for reading.
operator=	-
Save	Saves the EDeepLearningBenchmarkSettings object. The given ESerializer must have been created for writing.
SetBatchSize	Batch size.
SetCameraSpeed	Camera speed to simulate. If equal to 0, the benchmark will produce images as fast as possible.
SetDevice	Device to use for the benchmark.
SetEngine	Deep Learning engine to use for the benchmark.
SetInferencePrecision	Inference precision to use for the benchmark.
SetInternalResizeDisabled	Whether to disable the internal resize operation performed by the Deep Learning tools. When disabled, the benchmark will generate images at the optimal size for the tool.
SetName	Name of the benchmark.
SetNumExternalThreads	Number of external threads.
SetNumImages	Number of images to use during the benchmark.
SetNumInternalThreads	Number of internal threads.

EDeepLearningBenchmarkSettings::GetBatchSize

EDeepLearningBenchmarkSettings::SetBatchSize

Batch size.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBatchSize() const
void SetBatchSize(int val)
```

EDeepLearningBenchmarkSettings::GetCameraSpeed

EDeepLearningBenchmarkSettings::SetCameraSpeed

Camera speed to simulate. If equal to 0, the benchmark will produce images as fast as possible.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetCameraSpeed() const
void SetCameraSpeed(float fps)
```

Remarks

Camera speed higher than 500 frame per seconds will lead to unreliable results.

EDeepLearningBenchmarkSettings::GetDevice

EDeepLearningBenchmarkSettings::SetDevice

Device to use for the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningDevice GetDevice() const
void SetDevice(const EDeepLearningDevice& dev)
```

EDeepLearningBenchmarkSettings::EDeepLearningBenchmarkSettings

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EDeepLearningBenchmarkSettings(
 )
void EDeepLearningBenchmarkSettings(
 const EDeepLearningBenchmarkSettings& other
 )
```

Parameters

other

-

EDeepLearningBenchmarkSettings::GetEngine

EDeepLearningBenchmarkSettings::SetEngine

Deep Learning engine to use for the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetEngine() const

void SetEngine(const std::string& engine)
```

EDeepLearningBenchmarkSettings::GetInferencePrecision

EDeepLearningBenchmarkSettings::SetInferencePrecision

Inference precision to use for the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningInferencePrecision
GetInferencePrecision() const

void SetInferencePrecision(Euresys::Open_
eVision::EasyDeepLearning::EDeepLearningInferencePrecision prec)
```

EDeepLearningBenchmarkSettings::GetInternalResizeDisabled

EDeepLearningBenchmarkSettings::SetInternalResizeDisabled

Whether to disable the internal resize operation performed by the Deep Learning tools. When disabled, the benchmark will generate images at the optimal size for the tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetInternalResizeDisabled() const

void SetInternalResizeDisabled(bool disabled)
```

EDeepLearningBenchmarkSettings::Load

Loads a EDeepLearningBenchmarkSettings object. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& filePath
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

filePath

File path.

serializer

Pointer to the ESerializer created for reading.

EDeepLearningBenchmarkSettings::GetName

EDeepLearningBenchmarkSettings::SetName

Name of the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetName() const

void SetName(const std::string& benchName)
```

EDeepLearningBenchmarkSettings::GetNumExternalThreads

EDeepLearningBenchmarkSettings::SetNumExternalThreads

Number of external threads.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumExternalThreads() const
```

```
void SetNumExternalThreads(int val)
```

EDeepLearningBenchmarkSettings::GetNumImages

EDeepLearningBenchmarkSettings::SetNumImages

Number of images to use during the benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumImages() const
void SetNumImages(int val)
```

EDeepLearningBenchmarkSettings::GetNumInternalThreads

EDeepLearningBenchmarkSettings::SetNumInternalThreads

Number of internal threads.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumInternalThreads() const
void SetNumInternalThreads(int val)
```

EDeepLearningBenchmarkSettings::operator=

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningBenchmarkSettings& operator=(
 const EDeepLearningBenchmarkSettings& other
 )
```

Parameters

other

-

EDeepLearningBenchmarkSettings::Save

Saves the EDeepLearningBenchmarkSettings object. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& filePath
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

filePath

File path.

serializer

Pointer to the ESerializer created for writing.

4.75. EDeepLearningDefectDetectionMetrics Class

Collection of metrics used to evaluate a defect detection problem where the detection is based on the thresholding of a score produced by the underlying deep learning tool, e.g. a EUnsupervisedSegmenter tool or a ESupervisedSegmenter tool.
These metrics are only valid when results for good and defective images are included in the metrics (see EDeepLearningDefectDetectionMetrics::IsDefectDetectionMetricsValid). The definition of what is considered a good or a defective image depends on the deep learning tool used.
The defect detection metrics are separated in two main categories:
- Metrics dependent on the ROC (Receiver Operating Characteristic) curve. They require at least one good and one defective sample to be defined.
- Metrics dependent on the Precision/Recall curve. They require at least one defective sample to be defined.

The metrics related to the ROC curve are:
- The accuracy (see EDeepLearningDefectDetectionMetrics::GetAccuracy).
- The confusion matrix (see EDeepLearningDefectDetectionMetrics::GetConfusion and EConfusionMatrixElement)
- The ROC curve (see EDeepLearningDefectDetectionMetrics::GetROCPoint)
- The area Under ROC curve (see EDeepLearningDefectDetectionMetrics::AreaUnderROCCurve)

The metrics related to the precision/recall curve are:
- The average precision (see EDeepLearningDefectDetectionMetrics::AveragePrecision)
- Precision/Recall curve (see EDeepLearningDefectDetectionMetrics::GetPrecisionRecallCurvePoint)
- Precision (see EDeepLearningDefectDetectionMetrics::GetPrecision)
- Recall (see EDeepLearningDefectDetectionMetrics::GetRecall)
- F-Score (see EDeepLearningDefectDetectionMetrics::GetFScore).

The ROC and Precision/Recall curve are both obtained by computing some metrics for different values of the EDeepLearningDefectDetectionMetrics::ClassificationThreshold. Thus, each point on the curves gives different metrics, except for the area under the ROC curve and the average precision.
By default, the value of the metrics corresponds to the classification threshold of the corresponding deep learning tool. However, you can specify an index to retrieve the value of the metrics for other value of the EDeepLearningDefectDetectionMetrics::ClassificationThreshold. Metrics based on the ROC curve are indexed between 0 and EDeepLearningDefectDetectionMetrics::NumberOfClassifiers-1 and metrics based on the Precision/Recall curve are indexed between 0 and EDeepLearningDefectDetectionMetrics::NumPrecisionRecallCurvePoint-1.

Derived Class(es):ESupervisedSegmenterMetricsEUnsupervisedSegmenterMetrics

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

EDeepLearningDefectDetectionMetrics	Constructs an empty EDeepLearningDefectDetectionMetrics object.

GetAccuracy	The accuracy of the segmenter. The accuracy is the number of images that were correctly classified over the total number of images that was used to evaluate the classifier.
GetAreaUnderROCCurve	The area under ROC curve (AUC) of the classifier (see EDeepLearningDefectDetectionMetrics::GetROCPoint). It's value is between 0 and 1. In the context of unsupervised segmentation, the AUC is equal to the probability that good images will have a lower reconstruction error than defective images. This metrics measure discrimination capacity of a model. It tells how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is.
GetAveragePrecision	Average precision. The average precision is the area under the precision-recall curve. The presicion is the ratio between the number of true positive and the number of predicted possitive and the recall, also called the true positive rate, is the ratio between the number of true positive and the number of positive sample.
GetBestAccuracy	Best achievable accuracy. The classification threshold corresponding to this accuracy is given by EDeepLearningDefectDetectionMetrics::BestAccuracyClassificationThreshold.
GetBestAccuracyClassificationThreshold	Classification threshold giving the best achievable accuracy (see EDeepLearningDefectDetectionMetrics::BestAccuracy).
GetBestBalancedAccuracy	Best achievable balanced accuracy. The classification threshold corresponding to this accuracy is given by EDeepLearningDefectDetectionMetrics::BestBalancedAccuracyClassificationThreshold.
GetBestBalancedAccuracyClassificationThreshold	Classification threshold giving the best achievable balanced accuracy (see EDeepLearningDefectDetectionMetrics::BestBalancedAccuracy).
GetBestFScore	Best F1-Score. The F1-Score is the harmonic mean of the precision and recall (true positive rate).
GetBestFScoreThreshold	Classification threshold that yields the best F1-Score. The F1-Score is the harmonic mean of the precision and recall (true positive rate).
GetBestWeightedAccuracy	Best achievable weighted accuracy. The weighted accuracy is the weighted average of the true positive rate and the true negative rate (which is equal to 1 minus the false positive rate). See EROCPoint. The classification threshold corresponding to this accuracy is given by EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracyClassificationThreshold.
GetBestWeightedAccuracyClassificationThreshold	Classification threshold giving the best achievable weighted accuracy (see EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracy).

GetClassificationThreshold	Classification threshold. By default, the classification threshold will be equal to the classification threshold of the last unsupervised segmenter used to produce the results that compose this metric. Some metrics such as EDeepLearningDefectDetectionMetrics::GetROCPoint, EDeepLearningDefectDetectionMetrics::GetAccuracy or EDeepLearningDefectDetectionMetrics depends on the classification threshold. By default, these methods will returns the metric corresponding to the classification threshold.
GetConfusion	Confusion value of one label with another. The confusion value of a label with another is the number of images belonging to this label that are classified as belonging to the other label. For a EDeepLearningDefectDetectionMetrics there are only 2 labels (good and defective) so the confusion matrix is only composed of 4 values which are called matrix element (see EDeepLearningDefectDetectionMetrics). The confusion matrix is computed for a given threshold (see EUnsupervisedSegmenter::ClassificationThreshold) which means an index can be passed to the method (see EDeepLearningDefectDetectionMetrics::NumberOfClassifiers).
GetFScore	The F1-Score for the current threshold. The F1-Score is the harmonic mean of EDeepLearningDefectDetectionMetrics and EDeepLearningDefectDetectionMetrics.
GetNumberOfClassifiers	Number of different possible classifiers. Each classifier is obtained by choosing a different classification threshold (see EUnsupervisedSegmenter::ClassificationThreshold and corresponds to a point in the ROC curve (see EDeepLearningDefectDetectionMetrics::GetROCPoint).
GetNumDefectiveSample	Number of defective sample added to the metric.
GetNumGoodSample	Number of good sample added to the metric.
GetNumPrecisionRecallCurvePoint	Number of point in the precision/recall curve.
GetPrecision	Precision for the current threshold. The precision is the proportion of detected defective instances that were correctly identified as such.
GetPrecisionRecallCurveIndex	Index in the precision/recall curve for the current EDeepLearningDefectDetectionMetrics::ClassificationThreshold. The value of the index is between 0 and EDeepLearningDefectDetectionMetrics::NumPrecisionRecallCurvePoint - 1. The index is -1 if the precision/recall curve is not defined.
GetPrecisionRecallCurvePoint	Get a point on the precision/recall curve.
GetRecall	Recall for the current threshold. It is the proportion of defective instances that were correctly identified as such. It is also called the true positive rate.

GetROCPoint	ROC (Receiver Operating Characteristic) point. A ROC point is a point from the ROC curve which is the plot of the true positive rate against the false positive rate (see EConfusionMatrixElement) obtained at various classification threshold (see EUnsupervisedSegmenter::ClassificationThreshold). The ROC points are stricly ordered by decreasing threshold order meaning the true positive rate and false positive rate (see EConfusionMatrixElement) are sorted in increasing order.
IsDefectDetectionMetricsValid	Whether the defect detection metrics are valid or not. Defect detection metrics are completely valid when the metrics has results for at least one good and one defective images. Some metrics might be valid with only defective or good results.
Load	Loads the defect detection metrics. The given ESerializer must have been created for reading.
operator=	Copy operator.
Save	Saves the defect detection metrics. The given ESerializer must have been created for writing.
SetClassificationThreshold	Classification threshold. By default, the classification threshold will be equal to the classification threshold of the last unsupervised segmenter used to produce the results that compose this metric. Some metrics such as EDeepLearningDefectDetectionMetrics::GetROCPoint, EDeepLearningDefectDetectionMetrics::GetAccuracy or EDeepLearningDefectDetectionMetrics depends on the classification threshold. By default, these methods will returns the metric corresponding to the classification threshold.

EDeepLearningDefectDetectionMetrics::GetAreaUnderROCCurve

The area under ROC curve (AUC) of the classifier (see EDeepLearningDefectDetectionMetrics::GetROCPoint). It's value is between 0 and 1.
In the context of unsupervised segmentation, the AUC is equal to the probability that good images will have a lower reconstruction error than defective images.
This metrics measure discrimination capacity of a model.
It tells how much the model is capable of distinguishing between classes. The higher the AUC, the better the model is.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAreaUnderROCCurve() const
```

EDeepLearningDefectDetectionMetrics::GetAveragePrecision

Average precision.
The average precision is the area under the precision-recall curve. The presicion is the ratio between the number of true positive and the number of predicted possitive and the recall, also called the true positive rate, is the ratio between the number of true positive and the number of positive sample.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAveragePrecision() const
```

EDeepLearningDefectDetectionMetrics::GetBestAccuracy

Best achievable accuracy.
The classification threshold corresponding to this accuracy is given by EDeepLearningDefectDetectionMetrics::BestAccuracyClassificationThreshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestAccuracy() const
```

EDeepLearningDefectDetectionMetrics::GetBestAccuracyClassificationThreshold

Classification threshold giving the best achievable accuracy (see EDeepLearningDefectDetectionMetrics::BestAccuracy).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestAccuracyClassificationThreshold() const
```

EDeepLearningDefectDetectionMetrics::GetBestBalancedAccuracy

Best achievable balanced accuracy.
The classification threshold corresponding to this accuracy is given by EDeepLearningDefectDetectionMetrics::BestBalancedAccuracyClassificationThreshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestBalancedAccuracy() const
```

EDeepLearningDefectDetectionMetrics::GetBestBalancedAccuracyClassificationThreshold

Classification threshold giving the best achievable balanced accuracy (see EDeepLearningDefectDetectionMetrics::BestBalancedAccuracy).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestBalancedAccuracyClassificationThreshold() const
```

EDeepLearningDefectDetectionMetrics::GetBestFScore

Best F1-Score.
The F1-Score is the harmonic mean of the precision and recall (true positive rate).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestFScore() const
```

EDeepLearningDefectDetectionMetrics::GetBestFScoreThreshold

Classification threshold that yields the best F1-Score.
The F1-Score is the harmonic mean of the precision and recall (true positive rate).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestFScoreThreshold() const
```

EDeepLearningDefectDetectionMetrics::GetClassificationThreshold

EDeepLearningDefectDetectionMetrics::SetClassificationThreshold

Classification threshold.
By default, the classification threshold will be equal to the classification threshold of the last unsupervised segmenter used to produce the results that compose this metric.
Some metrics such as EDeepLearningDefectDetectionMetrics::GetROCPoint, EDeepLearningDefectDetectionMetrics::GetAccuracy or EDeepLearningDefectDetectionMetrics depends on the classification threshold. By default, these methods will returns the metric corresponding to the classification threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetClassificationThreshold() const

void SetClassificationThreshold(float threshold)
```

Remarks

Modifying the classification threshold in this class doesn't modify the classification threshold of the unsupervised segmenter.

EDeepLearningDefectDetectionMetrics::EDeepLearningDefectDetectionMetrics

Constructs an empty EDeepLearningDefectDetectionMetrics object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EDeepLearningDefectDetectionMetrics(
 )

void EDeepLearningDefectDetectionMetrics(
 const EDeepLearningDefectDetectionMetrics& other
 )
```

Parameters

other

Other object

EDeepLearningDefectDetectionMetrics::GetAccuracy

The accuracy of the segmenter.
The accuracy is the number of images that were correctly classified over the total number of images that was used to evaluate the classifier.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]

float GetAccuracy(
 int index
 )
```

Parameters

index

The index of the classifier to use. If the index is equal to '-1', the index corresponding to EUnsupervisedSegmenter::ClassificationThreshold will be used.

EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracy

Best achievable weighted accuracy.
The weighted accuracy is the weighted average of the true positive rate and the true negative rate (which is equal to 1 minus the false positive rate). See EROCPoint.
The classification threshold corresponding to this accuracy is given by EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracyClassificationThreshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]

float GetBestWeightedAccuracy(
 float goodWeight,
 float badWeight
 )
```

Parameters

goodWeight

Weight for the good label

badWeight

Weight for the bad label

Remarks

When using a dataset as the source for the label weights, the good weight is the weight of the "good" label and the bad weight is the sum of the weights of all the other labels.

EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracyClassificationThreshold

Classification threshold giving the best achievable weighted accuracy (see EDeepLearningDefectDetectionMetrics::GetBestWeightedAccuracy).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedAccuracyClassificationThreshold(
 float goodWeight,
 float badWeight
 )
```

Parameters

goodWeight

Weight for the good label

badWeight

Weight for the bad label

EDeepLearningDefectDetectionMetrics::GetConfusion

Confusion value of one label with another.
The confusion value of a label with another is the number of images belonging to this label that are classified as belonging to the other label.
For a EDeepLearningDefectDetectionMetrics there are only 2 labels (good and defective) so the confusion matrix is only composed of 4 values which are called matrix element (see EDeepLearningDefectDetectionMetrics).
The confusion matrix is computed for a given threshold (see EUnsupervisedSegmenter::ClassificationThreshold) which means an index can be passed to the method (see EDeepLearningDefectDetectionMetrics::NumberOfClassifiers).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetConfusion(
 Euresys::Open_eVision::EasyDeepLearning::EConfusionMatrixElement element,
 int index
 )
```

Parameters

element

The element from which to obtain the confusion value

index

The index of the classifier to use. If the index is '-1', the index corresponding to EUnsupervisedSegmenter::ClassificationThreshold will be used.

EDeepLearningDefectDetectionMetrics::GetFScore

The F1-Score for the current threshold.
The F1-Score is the harmonic mean of EDeepLearningDefectDetectionMetrics and EDeepLearningDefectDetectionMetrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetFScore(
 int index
 )
```

Parameters

index

Index for the precision/recall curve point or -1 to use the index corresponding to the current EDeepLearningDefectDetectionMetrics::ClassificationThreshold.

EDeepLearningDefectDetectionMetrics::GetPrecision

Precision for the current threshold.
The precision is the proportion of detected defective instances that were correctly identified as such.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetPrecision(
 int index
 )
```

Parameters

index

Index for the precision/recall curve point or -1 to use the index corresponding to the current EDeepLearningDefectDetectionMetrics::ClassificationThreshold.

EDeepLearningDefectDetectionMetrics::GetPrecisionRecallCurvePoint

Get a point on the precision/recall curve.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EROCPoint GetPrecisionRecallCurvePoint(
 int index
 )
```

Parameters

index

Index for the precision/recall curve point or -1 to use the index corresponding to the current EDeepLearningDefectDetectionMetrics::ClassificationThreshold.

Remarks

The precision/recall curve is defined when there is at least one ground truth positive sample in the metric.

EDeepLearningDefectDetectionMetrics::GetRecall

Recall for the current threshold.
It is the proportion of defective instances that were correctly identified as such. It is also called the true positive rate.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetRecall(
 int index
 )
```

Parameters

index

Index for the precision/recall curve point or -1 to use the index corresponding to the current EDeepLearningDefectDetectionMetrics::ClassificationThreshold.

EDeepLearningDefectDetectionMetrics::GetROCPoint

ROC (Receiver Operating Characteristic) point.
A ROC point is a point from the ROC curve which is the plot of the true positive rate against the false positive rate (see EConfusionMatrixElement) obtained at various classification threshold (see EUnsupervisedSegmenter::ClassificationThreshold).
The ROC points are stricly ordered by decreasing threshold order meaning the true positive rate and false positive rate (see EConfusionMatrixElement) are sorted in increasing order.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EROCPoint GetROCPoint(
 int index
 )
```

Parameters

index

The index of the classifier to use. If the index is equal to '-1', the index corresponding to EUnsupervisedSegmenter::ClassificationThreshold will be used.

Remarks

Each ROC point corresponds to a different classifier (see EDeepLearningDefectDetectionMetrics::NumberOfClassifiers).
It means that the ROC curve is the perfect tool to choose a threshold depending on the false and true positive rate values that best suit your application.

EDeepLearningDefectDetectionMetrics::IsDefectDetectionMetricsValid

Whether the defect detection metrics are valid or not. Defect detection metrics are completely valid when the metrics has results for at least one good and one defective images. Some metrics might be valid with only defective or good results.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsDefectDetectionMetricsValid(
 )
```

EDeepLearningDefectDetectionMetrics::Load

Loads the defect detection metrics. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EDeepLearningDefectDetectionMetrics::GetNumberOfClassifiers

Number of different possible classifiers.
Each classifier is obtained by choosing a different classification threshold (see EUnsupervisedSegmenter::ClassificationThreshold and corresponds to a point in the ROC curve (see EDeepLearningDefectDetectionMetrics::GetROCPoint).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumberOfClassifiers() const
```

Remarks

The number of classifiers is equal to 2 plus the number of results added to the metrics that have a unique classification score (i.e. different from the classification score of all the other results): each unique classification score corresponds to a classification threshold.

EDeepLearningDefectDetectionMetrics::GetNumDefectiveSample

Number of defective sample added to the metric.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumDefectiveSample() const
```

EDeepLearningDefectDetectionMetrics::GetNumGoodSample

Number of good sample added to the metric.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumGoodSample() const
```

EDeepLearningDefectDetectionMetrics::GetNumPrecisionRecallCurvePoint

Number of point in the precision/recall curve.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumPrecisionRecallCurvePoint() const
```

Remarks

The precision/recall curve is defined when there is at least one ground truth positive sample in the metric. The number of point in the precision/recall curve is equal to the number of positive sample plus 1.

EDeepLearningDefectDetectionMetrics::operator=

Copy operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EDeepLearningDefectDetectionMetrics& operator=(
 const EDeepLearningDefectDetectionMetrics& other
 )
```

Parameters

other

Other object

EDeepLearningDefectDetectionMetrics::GetPrecisionRecallCurveIndex

Index in the precision/recall curve for the current EDeepLearningDefectDetectionMetrics::ClassificationThreshold.
The value of the index is between 0 and EDeepLearningDefectDetectionMetrics::NumPrecisionRecallCurvePoint - 1. The index is -1 if the precision/recall curve is not defined.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetPrecisionRecallCurveIndex() const
```

EDeepLearningDefectDetectionMetrics::Save

Saves the defect detection metrics. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.76. EDeepLearningDevice Class

Class representing a device that can run a neural network. A device is fully described by its EDeepLearningDevice::Name and EDeepLearningDevice::EngineName.
The other properties are information about the device and its capabilities.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

EDeepLearningDevice	Default constructor. The resulting device will be invalid.
GetDefaultPrecision	Default precision for this device. If supported by the device, the default precision will be FLOAT32.
GetDeviceId	A device index corresponding to the index of the device within the engine. With the default engine, the device index is the index of the NVIDIA GPU for example.
GetDeviceType	The device type.
GetDeviceTypeDescription	Textual description of the device type.
GetEngineName	The name of the engine supporting this device.
GetFullName	A full name of the device.
GetName	Name of the device.
GetSupportedPrecisions	Precisions supported by the device.
HasInferenceCapability	Whether the device can perform inference or not.
HasTrainingCapability	Whether the device can perform training or not.
IsPrecisionSupported	Whether the given precision is supported by the device or not.

IsValid	Whether this instance represents a usable device or not.
operator!=	Inequality operator.
operator=	Copy operator
operator==	Equality operator.

EDeepLearningDevice::GetDefaultPrecision

Default precision for this device. If supported by the device, the default precision will be FLOAT32.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningInferencePrecision
GetDefaultPrecision() const
```

EDeepLearningDevice::GetDeviceId

A device index corresponding to the index of the device within the engine.
With the default engine, the device index is the index of the NVIDIA GPU for example.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetDeviceId() const
```

EDeepLearningDevice::GetDeviceType

The device type.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDeviceType GetDeviceType() const
```

EDeepLearningDevice::GetDeviceTypeDescription

Textual description of the device type.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetDeviceTypeDescription() const
```

EDeepLearningDevice::EDeepLearningDevice

Default constructor. The resulting device will be invalid.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EDeepLearningDevice(
 )
void EDeepLearningDevice(
 const EDeepLearningDevice& other
 )
```

Parameters

other

Device to copy

EDeepLearningDevice::GetEngineName

The name of the engine supporting this device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetEngineName() const
```

EDeepLearningDevice::GetFullName

A full name of the device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetFullName() const
```

EDeepLearningDevice::HasInferenceCapability

Whether the device can perform inference or not.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasInferenceCapability(
 )
```

EDeepLearningDevice::HasTrainingCapability

Whether the device can perform training or not.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasTrainingCapability(
 )
```

EDeepLearningDevice::IsPrecisionSupported

Whether the given precision is supported by the device or not.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsPrecisionSupported(
 Euresys::Open_eVision::EasyDeepLearning::EDeepLearningInferencePrecision precision
 )
```

Parameters

precision

Precision to check

EDeepLearningDevice::IsValid

Whether this instance represents a usable device or not.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsValid(
 )
```

EDeepLearningDevice::GetName

Name of the device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetName() const
```

EDeepLearningDevice::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool operator!=(
 const EDeepLearningDevice& other
 )
```

Parameters

other

Device to compare with

EDeepLearningDevice::operator=

Copy operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningDevice& operator=(
 const EDeepLearningDevice& other
 )
```

Parameters

other

Device to copy

EDeepLearningDevice::operator==

Equality operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool operator==(
 const EDeepLearningDevice& other
 )
```

Parameters

other

Device to compare with

EDeepLearningDevice::GetSupportedPrecisions

Precisions supported by the device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningInferencePrecision>
GetSupportedPrecisions() const
```

4.77. EDeepLearningExecutionSettings Class

Execution settings for a Deep Learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

CopyTo	Copy the object to another one.
EDeepLearningExecutionSettings	Constructs a EDeepLearningExecutionSettings.
GetAvailableDevice	-
GetAvailableDeviceName	Name of the i-th device of the current engine.
GetAvailableDevices	Get all available device for the current engine.
GetAvailableDevicesForEngine	Available devices for the specified engine.
GetAvailableEngineName	Gets the name of the engine given its id.
GetAvailableEngines	Available engines.
GetBatchSize	Batch size. The batch size is the number of images that are processed together during training and batch inference. When using multi-GPUs processing (see EDeepLearningExecutionSettings), the batch size is the number of images that each GPU will process at once. A large batch size will increase the processing speed on GPU but also the memory requirements. The batch size must be bigger or equal to 1 and it is commonly chosen to be a power of 2.
GetDevice	Gets an active device. By default, it returns the first active device.
GetDevices	Devices with which to run or train the model. Using multiple GPUs is only supported by the default engine.
GetEngine	Engine with which to execute or train the neural network.

GetInferencePrecision	Inference precision. Use EDeepLearningDevice::IsPrecisionSupported to check whether a device support the given precision.
GetNumAvailableDevices	Available devices for the current EDeepLearningExecutionSettings::Engine.
GetNumAvailableEngines	Number of available engines.
GetNumDevices	Number of devices configured in this execution context.
GetOptimizeBatchSize	Indicates whether to optimize the batch size (see EDeepLearningExecutionSettings::BatchSize) to maximize the training and inference speed according to the engine, device and the available memory. Default value is true.
Load	Loads the EDeepLearningExecutionSettings. The given ESerializer must have been created for reading.
operator=	Assignement operator.
Save	Saves the EDeepLearningExecutionSettings. The given ESerializer must have been created for writing.
SetBatchSize	Batch size. The batch size is the number of images that are processed together during training and batch inference. When using multi-GPUs processing (see EDeepLearningExecutionSettings), the batch size is the number of images that each GPU will process at once. A large batch size will increase the processing speed on GPU but also the memory requirements. The batch size must be bigger or equal to 1 and it is commonly chosen to be a power of 2.
SetDevice	Sets the device with which to run or train the model.
SetDeviceById	Sets the device with which to run the model using its index in the list returned by EDeepLearningExecutionSettings::AvailableDevices.
SetDeviceByName	Sets the device with which to run the model using its name.
SetDevices	Devices with which to run or train the model. Using multiple GPUs is only supported by the default engine.
SetEngine	Engine with which to execute or train the neural network.
SetInferencePrecision	Inference precision. Use EDeepLearningDevice::IsPrecisionSupported to check whether a device support the given precision.
SetOptimizeBatchSize	Indicates whether to optimize the batch size (see EDeepLearningExecutionSettings::BatchSize) to maximize the training and inference speed according to the engine, device and the available memory. Default value is true.

EDeepLearningExecutionSettings::GetAvailableDevices

Get all available device for the current engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDevice>
GetAvailableDevices() const
```

EDeepLearningExecutionSettings::GetAvailableEngines

Available engines.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
static std::vector<std::string> GetAvailableEngines()
```

EDeepLearningExecutionSettings::GetBatchSize

EDeepLearningExecutionSettings::SetBatchSize

Batch size. The batch size is the number of images that are processed together during training and batch inference.
When using multi-GPUs processing (see EDeepLearningExecutionSettings), the batch size is the number of images that each GPU will process at once.
A large batch size will increase the processing speed on GPU but also the memory requirements.
The batch size must be bigger or equal to 1 and it is commonly chosen to be a power of 2.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBatchSize() const

void SetBatchSize(int batchSize)
```

Remarks

If EDeepLearningExecutionSettings::OptimizeBatchSize is 'true', then the value of this property will not be taken into account because it will be optimized automatically for training or inference according to the situation.

EDeepLearningExecutionSettings::CopyTo

Copy the object to another one.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void CopyTo(
 EDeepLearningExecutionSettings& other
 )
```

Parameters

other

-

EDeepLearningExecutionSettings::SetDeviceById

Sets the device with which to run the model using its index in the list returned by EDeepLearningExecutionSettings::AvailableDevices.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetDeviceById(int id)
```

EDeepLearningExecutionSettings::SetDeviceByName

Sets the device with which to run the model using its name.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetDeviceByName(const std::string& name)
```

EDeepLearningExecutionSettings::GetDevices

EDeepLearningExecutionSettings::SetDevices

Devices with which to run or train the model.
Using multiple GPUs is only supported by the default engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDevice> GetDevices()
const
```

```
void SetDevices(const std::vector<Euresys::Open_
eVision::EasyDeepLearning::EDeepLearningDevice>& devices)
```

EDeepLearningExecutionSettings::EDeepLearningExecutionSettings

Constructs a EDeepLearningExecutionSettings.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EDeepLearningExecutionSettings(
 )
void EDeepLearningExecutionSettings(
 const EDeepLearningExecutionSettings& other
 )
```

Parameters

other

-

EDeepLearningExecutionSettings::GetEngine

EDeepLearningExecutionSettings::SetEngine

Engine with which to execute or train the neural network.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetEngine() const
void SetEngine(const std::string& engine)
```

EDeepLearningExecutionSettings::GetAvailableDevice

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EDeepLearningDevice GetAvailableDevice(
 int i
 )
```

Parameters

i

-

EDeepLearningExecutionSettings::GetAvailableDeviceName

Name of the i-th device of the current engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetAvailableDeviceName(
 int i
 )
```

Parameters

i

Index of the device between 0 and EDeepLearningExecutionSettings - 1.

EDeepLearningExecutionSettings::GetAvailableDevicesForEngine

Available devices for the specified engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDevice>
GetAvailableDevicesForEngine(
 const std::string& engine
 )
```

Parameters

engine

Engine for which to get the devices

EDeepLearningExecutionSettings::GetAvailableEngineName

Gets the name of the engine given its id.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetAvailableEngineName(
 int id
 )
```

Parameters

id

Index of the available engine between 0 and EDeepLearningExecutionSettings::NumAvailableEngines - 1.

EDeepLearningExecutionSettings::GetDevice

Gets an active device. By default, it returns the first active device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]

EDeepLearningDevice GetDevice(
 int i
 )
```

Parameters

i

Index of the active devices between 0 and EDeepLearningExecutionSettings.

EDeepLearningExecutionSettings::GetInferencePrecision

EDeepLearningExecutionSettings::SetInferencePrecision

Inference precision.
Use EDeepLearningDevice::IsPrecisionSupported to check whether a device support the given precision.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]

Euresys::Open_eVision::EasyDeepLearning::EDeepLearningInferencePrecision
GetInferencePrecision() const

void SetInferencePrecision(Euresys::Open_
eVision::EasyDeepLearning::EDeepLearningInferencePrecision prec)
```

EDeepLearningExecutionSettings::Load

Loads the EDeepLearningExecutionSettings. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& filePath
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

filePath

File path.

serializer

Pointer to the ESerializer created for reading.

Remarks

Loading a EDeepLearningExecutionSettings can trigger exceptions if the engine and/or device is not available on the machine.

EDeepLearningExecutionSettings::GetNumAvailableDevices

Available devices for the current EDeepLearningExecutionSettings::Engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumAvailableDevices() const
```

EDeepLearningExecutionSettings::GetNumAvailableEngines

Number of available engines.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
static int GetNumAvailableEngines()
```

EDeepLearningExecutionSettings::GetNumDevices

Number of devices configured in this execution context.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumDevices() const
```

EDeepLearningExecutionSettings::operator=

Assignement operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningExecutionSettings& operator=(
 const EDeepLearningExecutionSettings& other
 )
```

Parameters

other

-

EDeepLearningExecutionSettings::GetOptimizeBatchSize

EDeepLearningExecutionSettings::SetOptimizeBatchSize

Indicates whether to optimize the batch size (see EDeepLearningExecutionSettings::BatchSize) to maximize the training and inference speed according to the engine, device and the available memory.
Default value is true.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool GetOptimizeBatchSize() const

void SetOptimizeBatchSize(bool optimize)
```

EDeepLearningExecutionSettings::Save

Saves the EDeepLearningExecutionSettings. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Save(
 const std::string& filePath
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

filePath
File path.

serializer
Pointer to the ESerializer created for writing.

EDeepLearningExecutionSettings::SetDevice

Sets the device with which to run or train the model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetDevice(
 const EDeepLearningDevice& device
 )
```

Parameters

device
-

Remarks

For CPU device, the number of thread is initialized from Easy

4.78. EDeepLearningProject Class

Deep learning project.
A deep learning project manages a directory (EDeepLearningProject::ProjectDirectory) in which it stores:
- A main project file (EDeepLearningProject::ProjectFile) containing the dataset, the dataset splits, data augmentation objects, and meta data about the tools;
- A image directory to store the images of the dataset (EDeepLearningProject::ImageDirectory);
- A directory for each tool named after the corresponding tool name (EDeepLearningProject::GetToolName).

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

AddBenchmark	Adds a benchmark for the given tool.
AddDataAugmentation	Adds an empty data augmentation.
AddSplit	Adds a split.
AddTool	Adds a new tool with the given name.
CopyTo	Copy the project.

EDeepLearningProject	Creates or opens a project.
GetBenchmark	Gets a benchmark for a given tool.
GetCreationDate	Creation date of the project represented as the number of seconds since the UNIX epoch.
GetDataAugmentation	Gets the data augmentation object for the given index.
GetDataAugmentationCreationDate	Creation date of the data augmentation represented as the number of seconds since the UNIX epoch.
GetDataAugmentationCreationISODate	Creation date of the data augmentation formatted as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).
GetDataAugmentationName	Name of the data augmentation.
GetDataset	Gets the dataset contained in the project.
GetFileStructureUpdateDescription	Description of the file structure update.
GetGoodLabel	Good label for EasySegment Unsupervised.
GetImageDirectory	Image directory. Calling this method will create the image directory if it doesn't exist.
GetImportImageIntoProjectDirectory	Whether to import the images into the EDeepLearningProject::ImageDirectory. (Default value: true)
GetISOCreationDate	Creation date of the project as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).
GetName	Name of the project.
GetNumBenchmarks	Number of benchmarks for the given tool.
GetNumDataAugmentations	Number of data augmentation.
GetNumSplits	Number of dataset splits in the project.
GetNumTools	Number of tools.
GetProjectDirectory	Project directory. The project directory can only be set on an empty project. Setting the project directory will create the directory if it doesn't exist.
GetProjectFile	Path to the project file. The path to the project file is only defined for a project that has non-empty EDeepLearningProject::Name and EDeepLearningProject::ProjectDirectory.
GetResultFilename	Result filename for the given image.
GetSplit	Gets the dataset split object for the given index.
GetSplitCreationDate	Creation date of the split represented as the number of seconds since the UNIX epoch.
GetSplitCreationISODate	Creation date of the split formatted as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).
GetSplitName	Name of the split.

GetToolCopy	Loads and returns the i-th tool of the project. The user is responsible for deleting the returned object.
GetToolCreationDate	Creation date of the tool represented as the number of seconds since the UNIX epoch.
GetToolDataAugmentation	Data augmentation associated with a tool.
GetToolISOCreationDate	Creation date of the tool represented as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).
GetToolName	Tool name.
GetToolSplit	Dataset split associated with a tool.
GetType	Type of the tools in the projects.
HasFileStructureUpdates	Whether the project can be updated to a new folder and file structure.
ImportTool	Imports an existing tool. Note that the imported tool will be copied into the project directory. The original file will no be used by the project. The labels' name, weight, and color in the imported tool will be taken from the dataset.
IsToolNameValid	Checks that the tool name is valid (not used by any other tool).
Load	Loads a project.
operator=	Assignment operator.
RemoveBenchmark	Removes a benchmark.
RemoveDataAugmentation	Removes the given data augmentation. The data augmentation at index 0 can't be removed.
RemoveSplit	Removes a split.
RemoveTool	Removes the given tool from the project. This will also remove all files and directories associated with the tool (tool directory, result and metrics files).
Save	Saves a project and set the project directory to the parent directory of projectFile.
SaveProject	Saves an opened project.
SetBenchmark	Replaces a benchmark.
SetDataAugmentationName	Name of the data augmentation.
SetGoodLabel	Good label for EasySegment Unsupervised.
SetImportImageIntoProjectDirectory	Whether to import the images into the EDeepLearningProject::ImageDirectory. (Default value: true)
SetName	Name of the project.
SetProjectDirectory	Project directory. The project directory can only be set on an empty project. Setting the project directory will create the directory if it doesn't exist.

SetSplitName	Name of the split.
SetToolDataAugmentation	Data augmentation associated with a tool.
SetToolName	Tool name. Changing the name of a tool will move the directory in which the tool is saved.
SetToolSplit	Dataset split associated with a tool.
SetType	Type of the tools in the projects.
UnsetGoodLabel	Remove the good label for EasySegment Unsupervised.
UpdateProjectFileStructure	Updates the project file structure.

EDeepLearningProject::AddBenchmark

Adds a benchmark for the given tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddBenchmark(
 int toolId,
 const EDeepLearningBenchmark& benchmark
 )
```

Parameters

toolId

Index of the tool

benchmark

Benchmark to add to the project

EDeepLearningProject::AddDataAugmentation

Adds an empty data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddDataAugmentation(
 const std::string& name
 )

void AddDataAugmentation(
 const std::string& name,
 const EDataAugmentation& dataAugmentation
 )
```

Parameters

name

Name for the data augmentation

dataAugmentation

Instance of EDataAugmentation class

EDeepLearningProject::AddSplit

Adds a split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddSplit(
 const std::string& name,
 const EDatasetSplit& split
 )
```

Parameters

name

Name of the split

split

Instance of EDatasetSplit class.

EDeepLearningProject::AddTool

Adds a new tool with the given name.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddTool(
 const std::string& name
 )
```

Parameters

name

Name of the tool.

EDeepLearningProject::CopyTo

Copy the project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void CopyTo(
 EDeepLearningProject& other
 )
```

Parameters

other

Other project

EDeepLearningProject::GetCreationDate

Creation date of the project represented as the number of seconds since the UNIX epoch.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_INT64 GetCreationDate() const
```

EDeepLearningProject::GetDataset

Gets the dataset contained in the project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
const EClassificationDataset& GetDataset() const
```

EDeepLearningProject::EDeepLearningProject

Creates or opens a project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EDeepLearningProject(
 const std::string& projectFile
 )

void EDeepLearningProject(
 const std::string& projectDir,
 const std::string& projectName,
 Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType type
 )

void EDeepLearningProject(
 )
```

```
void EDeepLearningProject(
 const EDeepLearningProject& other
 )
```

Parameters

projectFile
Main project file top open

projectDir
Directory for a new project

projectName
Name of the new project project

type
Type of the new project

other
Other project for the copy constructor

EDeepLearningProject::GetFileStructureUpdateDescription

Description of the file structure update.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetFileStructureUpdateDescription() const
```

EDeepLearningProject::GetBenchmark

Gets a benchmark for a given tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
const EDeepLearningBenchmark& GetBenchmark(
 int toolId,
 int benchId
 )

EDeepLearningBenchmark& GetBenchmark(
 int toolId,
 int benchId
 )
```

Parameters

toolId
Index of the tool

benchId
Index of the benchmark

EDeepLearningProject::GetDataAugmentation

Gets the data augmentation object for the given index.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
const EDataAugmentation& GetDataAugmentation(
 int dataAugmentationId
 )
```

Parameters

dataAugmentationId
Index of the data augmentation

EDeepLearningProject::GetDataAugmentationCreationDate

Creation date of the data augmentation represented as the number of seconds since the UNIX epoch.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_INT64 GetDataAugmentationCreationDate(
 int dataAugmentationId
 )
```

Parameters

dataAugmentationId
Index of the data augmentation

EDeepLearningProject::GetDataAugmentationCreationISODate

Creation date of the data augmentation formatted as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetDataAugmentationCreationISODate(
 int dataAugmentationId
 )
```

Parameters

dataAugmentationId

Index of the data augmentation

EDeepLearningProject::GetDataAugmentationName

Name of the data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetDataAugmentationName(
 int dataAugmentationId
 )
```

Parameters

dataAugmentationId

Index of the data augmentation

EDeepLearningProject::GetNumBenchmarks

Number of benchmarks for the given tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumBenchmarks(
 int toolId
 )
```

Parameters

toolId

Index of the tool

EDeepLearningProject::GetResultFilename

Result filename for the given image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetResultFilename(
 int imageId
 )
```

Parameters

imageId
Index of the image in the dataset

EDeepLearningProject::GetSplit

Gets the dataset split object for the given index.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
const EDatasetSplit& GetSplit(
 int splitId
 )
```

Parameters

splitId
Index of the split

EDeepLearningProject::GetSplitCreationDate

Creation date of the split represented as the number of seconds since the UNIX epoch.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_INT64 GetSplitCreationDate(
 int splitId
 )
```

Parameters

splitId
Index of the split

EDeepLearningProject::GetSplitCreationISODate

Creation date of the split formatted as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetSplitCreationISODate(
 int splitId
 )
```

Parameters

splitId

Index of the split

EDeepLearningProject::GetSplitName

Name of the split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetSplitName(
 int splitId
 )
```

Parameters

splitId

Index of the split

EDeepLearningProject::GetToolCopy

Loads and returns the i-th tool of the project.
The user is responsible for deleting the returned object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningTool* GetToolCopy(
 int toolId,
 bool includeTrainingModel
 )
```

Parameters

toolId

Index of the tool

includeTrainingModel

Whether to include the training model

EDeepLearningProject::GetToolCreationDate

Creation date of the tool represented as the number of seconds since the UNIX epoch.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_INT64 GetToolCreationDate(
 int toolId
 )
```

Parameters

toolId

Index of the tool

EDeepLearningProject::GetToolDataAugmentation

Data augmentation associated with a tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetToolDataAugmentation(
 int toolId
 )
```

Parameters

toolId

Index of the tool

EDeepLearningProject::GetToolISOCreationDate

Creation date of the tool represented as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetToolISOCreationDate(
 int toolId
 )
```

Parameters

toolId

Index of the tool

EDeepLearningProject::GetToolName

Tool name.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetToolName(
 int toolId
 )
```

Parameters

toolId

Index of the tool

EDeepLearningProject::GetToolSplit

Dataset split associated with a tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetToolSplit(
 int toolId
 )
```

Parameters

toolId

Index of the tool

EDeepLearningProject::GetGoodLabel

EDeepLearningProject::SetGoodLabel

Good label for EasySegment Unsupervised.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetGoodLabel() const
void SetGoodLabel(const std::string& newGoodLabel)
```

EDeepLearningProject::HasFileStructureUpdates

Whether the project can be updated to a new folder and file structure.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasFileStructureUpdates(
 )
```

EDeepLearningProject::GetImageDirectory

Image directory.
Calling this method will create the image directory if it doesn't exist.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetImageDirectory() const
```

EDeepLearningProject::GetImportImageIntoProjectDirectory

EDeepLearningProject::SetImportImageIntoProjectDirectory

Whether to import the images into the EDeepLearningProject::ImageDirectory. (Default value: true)

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool GetImportImageIntoProjectDirectory() const
void SetImportImageIntoProjectDirectory(bool importImages)
```

Remarks

This is a hint for Deep Learning Studio and this property will not be enforced when adding images through the API.
Importing the images into the project directory allows relocating a project to another computer.

EDeepLearningProject::ImportTool

Imports an existing tool. Note that the imported tool will be copied into the project directory. The original file will no be used by the project.
The labels' name, weight, and color in the imported tool will be taken from the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void ImportTool(
 const std::string& name,
 const std::string& path
 )
void ImportTool(
 const std::string& name,
 EDeepLearningTool* tool
 )
```

Parameters

name
Name for the imported tool.

path
Path towards the tool file.

tool
Tool object to import.

EDeepLearningProject::GetISOCreationDate

Creation date of the project as an ISO 8601 date (YYYY-MM-DD HH:MM:SS).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetISOCreationDate() const
```

EDeepLearningProject::IsToolNameValid

Checks that the tool name is valid (not used by any other tool).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsToolNameValid(
 const std::string& toolName
 )
```

Parameters

toolName
Tool name

EDeepLearningProject::Load

Loads a project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Load(
 const std::string& projectFile
 )
```

Parameters

projectFile

Project file to load

EDeepLearningProject::GetName

EDeepLearningProject::SetName

Name of the project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetName() const
void SetName(const std::string& projectName)
```

EDeepLearningProject::GetNumDataAugmentations

Number of data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumDataAugmentations() const
```

Remarks

A project always contains an empty data augmentation at index 0.

EDeepLearningProject::GetNumSplits

Number of dataset splits in the project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumSplits() const
```

EDeepLearningProject::GetNumTools

Number of tools.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumTools() const
```

EDeepLearningProject::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EDeepLearningProject& operator=(
 const EDeepLearningProject& other
 )
```

Parameters

other

Other project

EDeepLearningProject::GetProjectDirectory

EDeepLearningProject::SetProjectDirectory

Project directory.
The project directory can only be set on an empty project. Setting the project directory will create the directory if it doesn't exist.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetProjectDirectory() const

void SetProjectDirectory(const std::string& projectDirectory)
```

EDeepLearningProject::GetProjectFile

Path to the project file.
The path to the project file is only defined for a project that has non-empty EDeepLearningProject::Name and EDeepLearningProject::ProjectDirectory.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetProjectFile() const
```

EDeepLearningProject::RemoveBenchmark

Removes a benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveBenchmark(
 int toolId,
 int benchmarkId
 )
```

Parameters

toolId
Index of the tool

benchmarkId
Index of the benchmark to remove

EDeepLearningProject::RemoveDataAugmentaton

Removes the given data augmentation. The data augmentation at index 0 can't be removed.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveDataAugmentaton(
 int dataAugmentationId
 )
```

Parameters

dataAugmentationId
Index of the data augmentation to remove

EDeepLearningProject::RemoveSplit

Removes a split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void RemoveSplit(
 int splitId
 )
```

Parameters

splitId

Index of the split

EDeepLearningProject::RemoveTool

Removes the given tool from the project. This will also remove all files and directories associated with the tool (tool directory, result and metrics files).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void RemoveTool(
 int toolId
 )
```

Parameters

toolId

Index of the tool to remove

EDeepLearningProject::Save

Saves a project and set the project directory to the parent directory of projectFile.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Save(
 const std::string& projectFile
 )
```

Parameters

projectFile

Project file to save

EDeepLearningProject::SaveProject

Saves an opened project.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SaveProject(
 )
```

EDeepLearningProject::SetBenchmark

Replaces a benchmark.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetBenchmark(
 int toolId,
 int benchmarkId,
 const EDeepLearningBenchmark& benchmark
 )
```

Parameters

toolId
Index of the tool

benchmarkId
Index of the benchmark to replace

benchmark
New benchmark

EDeepLearningProject::SetDataAugmentationName

Name of the data augmentation.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetDataAugmentationName(
 int dataAugmentationId,
 const std::string& name
 )
```

Parameters

dataAugmentationId

Index of the data augmentation

name

New name for the data augmentation

EDeepLearningProject::SetSplitName

Name of the split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetSplitName(
 int splitId,
 const std::string& newName
 )
```

Parameters

splitId

Index of the split

newName

New name for the split

EDeepLearningProject::SetToolDataAugmentation

Data augmentation associated with a tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetToolDataAugmentation(
 int toolId,
 int dataAugmentationId
 )
```

Parameters

toolId

Index of the tool

dataAugmentationId

Index of the data augmentation

EDeepLearningProject::SetToolName

Tool name.
Changing the name of a tool will move the directory in which the tool is saved.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetToolName(
 int toolId,
 const std::string& name
 )
```

Parameters

toolId
Index of the tool

name
New name for the tool

EDeepLearningProject::SetToolSplit

Dataset split associated with a tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetToolSplit(
 int toolId,
 int splitId
 )
```

Parameters

toolId
Index of the tool

splitId
Index of the split

EDeepLearningProject::GetType

EDeepLearningProject::SetType

Type of the tools in the projects.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetType() const
void SetType(Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType type)
```

EDeepLearningProject::UnsetGoodLabel

Remove the good label for EasySegment Unsupervised.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UnsetGoodLabel(
 )
```

EDeepLearningProject::UpdateProjectFileStructure

Updates the project file structure.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UpdateProjectFileStructure(
 )
```

4.79. EDeepLearningTool Class

EDeepLearningTool represents the common operations of deep learning tools.

The class is responsible for handling CPU/GPU settings and training.
Computing the result for an image is called inference and is the responsability of the actual deep learning tools (see EClassifier).

Derived Class(es):EClassifierELocatorBaseESupervisedSegmenterEUnsupervisedSegmenter

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Create	Factory method to open a deep learning tool. Returns the corresponding deep learning tool that must be deleted by the caller.
GetActiveDevice	Gets the active device. By default, it returns the first active device.

GetActiveDevices	Devices with which to run or train the model. Only multiple GPUs with the default engine is supported.
GetAutoSavePeriod	Auto-save period. The auto-save period is the number of training iterations between each automatic save of the model. In order for the model to be automatically saved during the training, the tool must have been loaded or saved to a file. The path where the tool will be saved is either indicated by EDeepLearningTool::Path or the three paths EDeepLearningTool::SettingsPath, EDeepLearningTool::InferenceModelPath, and EDeepLearningTool::TrainingModelPath. If one of these path is empty, the corresponding part of the tool will not be saved automatically during training. By default, auto save is disabled and the period is equal to 0.
GetAvailableDevice	-
GetAvailableDeviceName	Name of the i-th device of the current engine.
GetAvailableDevices	Get all available device for the current engine.
GetAvailableDevicesForEngine	Available devices for the specified engine.
GetAvailableEngineName	Gets the name of the engine given its id.
GetAvailableEngines	Available engines.
GetBatchSize	Batch size. The batch size is the number of images that are processed together during training and batch inference. When using multi-GPUs processing (see EDeepLearningTool::GPUIndexes), the batch size is the number of images that each GPU will process at once. A large batch size will increase the processing speed on GPU but also the memory requirements. The batch size must be bigger or equal to 1 and it is commonly chosen to be a power of 2.
GetBatchSizeForMaximumInferenceSpeed	Computes the batch size that will maximize the inference speed on NVIDIA GPU. For all other devices, it will return 1.
GetBestIteration	Iteration at which the minimum validation error was reached. After training, the tool is in the state it was at this best iteration.
GetCurrentTrainingFinishedIterations	Number of iterations that are already finished in the current training.
GetCurrentTrainingNumIterations	Total number of training iterations that will be performed during the current training of the deep learning tool. After the end of the training, this number is the actual number of iterations performed during the training which can be lower than the number of iterations given to the EDeepLearningTool::Train method (for example if the user called EDeepLearningTool::StopTraining).
GetCurrentTrainingProgression	Current training progression as a percentage (between 0 and 1).

GetDeterministicTrainingRandomSeed	Random seed to use when EDeepLearningTool::EnableDeterministicTraining is set to true. The default value is 42. When EDeepLearningTool::EnableDeterministicTraining, the random seed is generated using a random non-deterministic source.
GetEnableDeterministicTraining	Whether to use determinitic training algorithm that allows to repeatable training results. Default value: false.
GetEnableGPU	Deprecated: Use EDeepLearningTool. Enable the use of a GPU for training and inference of the deep learning tool. If the active engine is not the default engine, EDeepLearningTool::EnableGPU will be false even if the active device has a GPU device type. Setting EDeepLearningTool::EnableGPU to any value when the active engine is not the default one will throw an exception.
GetEngine	Sets the engine with which to execute or train the neural network.
GetExecutionSettings	Execution settings of the tool (engine, device(s), inference precision, batch size settings).
GetGPUIndexes	Deprecated: use EDeepLearningTool or EDeepLearningTool::ActiveDevices. Indexes of the GPUs to use for computations. Using multiple GPUs is only possible when we can process multiple images at once, i.e. during training (EDeepLearningTool::Train) or batch inference. By default, all the detected GPUs will be used.
GetImageCacheSize	Size in byte of the image cache. The cache is used during training to store reformatted and normalized images. A correctly sized cache can reduce the hard drive accesses and the preprocessing time for each image.
GetInferenceModelPath	Path of the inference model when loaded or saved using EDeepLearningTool::SaveInferenceModel, EDeepLearningTool::LoadInferenceModel or EDeepLearningTool::SerializeInferenceModel.
GetInferencePrecision	Inference precision. Use EDeepLearningDevice::IsPrecisionSupported to check whether a device support the given precision.
GetLabel	Gets the label from its index. If the tool is not trained, the method may throw an exception.
GetLabelColor	Label color. The label colors affect how the results will be displayed. The default label colors are the one specified in the dataset used for training.
GetLabelOpacity	Gets the label opacity (between 0 for fully transparent and 255 for opaque color). The label opacities affect with the label colors how the results will be displayed. The default label opacities are the one specified in the dataset used for training.
GetLabelWeight	Gets the label weight. If the tool is not trained, the label weights are not defined. After training, they are set to the value of the training dataset label weights.

GetNumActiveDevices	Gets the number of devices with which to run or train the model.
GetNumAvailableDevices	Available devices for the current EDeepLearningTool::Engine.
GetNumAvailableEngines	Number of available engines.
GetNumGPUs	Deprecated: use EDeepLearningTool with a device type of GPU. Number of detected NVIDIA GPU.
GetNumLabels	Number of labels of this tool. If the tool is not trained, the method will throw an exception.
GetNumPatchesForImage	Number of patches that will be extracted from an input image to perform inference. For EasyClassify and EasyLocate, this will always be equal to 1. For EasySegment, the number of patches will depend on the scale, patch size and sampling density parameters.
GetNumTrainedIterations	Number of iterations that were performed to train this deep learning tool.
GetOptimalNumImagesForBatchSize	Optimal number of images to process together for inference given the batch size and EDeepLearningTool::GetNumPatchesForImage. In practice, for EasyClassify and EasyLocate, this is the same as the batch size. For EasySegment, the value will depend on the scale, patch size and sampling density parameters.
GetOptimizeBatchSize	Indicates whether to optimize the batch size (see EDeepLearningTool::BatchSize) to maximize the training and inference speed according to the engine, the device and the available memory. Default value is true.
GetPath	Path of the tool when loaded or saved using EDeepLearningTool::Save, EDeepLearningTool::Load or EDeepLearningTool.
GetSettingsPath	Path of the settings when loaded or saved using EDeepLearningTool::SaveSettings, EDeepLearningTool::LoadSettings or EDeepLearningTool::SerializeSettings.
GetToolType	Type of the deep learning tool.
GetTrainingModelPath	Path of the training model when loaded or saved using EDeepLearningTool::SaveTrainingModel, EDeepLearningTool::LoadTrainingModel or EDeepLearningTool::SerializeTrainingModel.
HasInferenceModel	Whether the inference model is loaded or created when the tool is not yet trained.
HasTrainingModel	Whether the training model is loaded or created when the tool is not yet trained.
InitializeInference	Initializes the neural network for inference (i.e. making predictions with the neural network). You need to pass an image or a vector of images with the same characteristics (image resolution, image type and number of image) as the image or vector of images that you will use to make inference with this tool.

IsTrained	Tells whether the deep learning tool has been trained.
IsTraining	Indicates whether the object is currently training.
Load	Loads a deep learning tool. The given ESerializer must have been created for reading.
LoadInferenceModel	Loads the tool's inference model. The inference model is required to apply the tool on new images.
LoadSettings	Loads the settings of a tool.
LoadTrainingModel	Loads the tool's training model. The training model is required to continue a training.
Save	Saves the settings, inference and training model of the deep learning tool. The given ESerializer must have been created for writing.
SaveInferenceModel	Saves the tool's inference model to a file.
SaveSettings	Saves the settings of the tool to a file.
SaveTrainingModel	Saves the tool's training model to a file.
SerializeInferenceModel	Serializes the inference model of the tool.
SerializeSettings	Serializes the tool settings.
SerializeTrainingModel	Serializes the training model of the tool. The training model is necessary to continue training the tool.
SetActiveDevice	Sets the device with which to run or train the model.
SetActiveDeviceById	Sets the device with which to run the model using its index in the list returned by EDeepLearningTool::AvailableDevices.
SetActiveDeviceByName	Sets the device with which to run the model using its name.
SetActiveDevices	Devices with which to run or train the model. Only multiple GPUs with the default engine is supported.
SetActiveDevicesById	Sets the devices with which to run the model using their index in the list returned by EDeepLearningTool::AvailableDevices.
SetAutoSavePeriod	Auto-save period. The auto-save period is the number of training iterations between each automatic save of the model. In order for the model to be automatically saved during the training, the tool must have been loaded or saved to a file. The path where the tool will be saved is either indicated by EDeepLearningTool::Path or the three paths EDeepLearningTool::SettingsPath, EDeepLearningTool::InferenceModelPath, and EDeepLearningTool::TrainingModelPath. If one of these path is empty, the corresponding part of the tool will not be saved automatically during training. By default, auto save is disabled and the period is equal to 0.

SetBatchSize	Batch size. The batch size is the number of images that are processed together during training and batch inference. When using multi-GPUs processing (see EDeepLearningTool::GPUIndexes), the batch size is the number of images that each GPU will process at once. A large batch size will increase the processing speed on GPU but also the memory requirements. The batch size must be bigger or equal to 1 and it is commonly chosen to be a power of 2.
SetDeterministicTrainingRandomSeed	Random seed to use when EDeepLearningTool::EnableDeterministicTraining is set to true. The default value is 42. When EDeepLearningTool::EnableDeterministicTraining, the random seed is generated using a random non-deterministic source.
SetEnableDeterministicTraining	Whether to use determinitic training algorithm that allows to repeatable training results. Default value: false.
SetEnableGPU	Deprecated: Use EDeepLearningTool. Enable the use of a GPU for training and inference of the deep learning tool. If the active engine is not the default engine, EDeepLearningTool::EnableGPU will be false even if the active device has a GPU device type. Setting EDeepLearningTool::EnableGPU to any value when the active engine is not the default one will throw an exception.
SetEngine	Sets the engine with which to execute or train the neural network.
SetExecutionSettings	Execution settings of the tool (engine, device(s), inference precision, batch size settings).
SetGPUIndexes	Deprecated: use EDeepLearningTool or EDeepLearningTool::ActiveDevices. Indexes of the GPUs to use for computations. Using multiple GPUs is only possible when we can process multiple images at once, i.e. during training (EDeepLearningTool::Train) or batch inference. By default, all the detected GPUs will be used.
SetImageCacheSize	Size in byte of the image cache. The cache is used during training to store reformatted and normalized images. A correctly sized cache can reduce the hard drive accesses and the preprocessing time for each image.
SetInferencePrecision	Inference precision. Use EDeepLearningDevice::IsPrecisionSupported to check whether a device support the given precision.
SetLabel	Changes a label predicted by the tool.
SetLabelColor	Changes a label color.
SetLabelOpacity	Changes a label opacity (between 0 for fully transparent and 255 for opaque color).
SetLabelWeight	Sets the label weight. If the tool is not trained, the label weights are not defined. After training, they are set to the value of the training dataset label weights.

SetOptimizeBatchSize	Indicates whether to optimize the batch size (see EDeepLearningTool::BatchSize) to maximize the training and inference speed according to the engine, the device and the available memory. Default value is true.
StopTraining	Stops training and returns the last completed iteration. If the parameter 'wait' is set to true, the method will wait for the training thread to completely stop. Otherwise, the method will return immediately.
Train	Trains the EDeepLearningTool with the given dataset for the specified number of iterations. At the end of the training, the deep learning tool is in the state it was at the iteration that gave the minimum validation error. See EDeepLearningTool::BestIteration.
UnloadInferenceModel	Unloads the inference model.
UnloadModels	Unloads the inference and training model.
UnloadTrainingModel	Unloads the training model.
WaitForIterationCompletion	Waits until an iteration is complete. A call to this method will block the calling thread until a training iteration in the training thread is finished. This method returns the number of trained iterations.
WaitForTrainingCompletion	Waits until the training is complete or the timeout is expired. A call to this method will block the calling thread for the shortest time between the timeout and the time it takes for the training to complete. A negative timeout means that the method will wait until the training is complete. The default value is set to -1. The method returns the number of trained iterations.

EDeepLearningTool::SetActiveDeviceById

Sets the device with which to run the model using its index in the list returned by EDeepLearningTool::AvailableDevices.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetActiveDeviceById(int id)
```

EDeepLearningTool::SetActiveDeviceByName

Sets the device with which to run the model using its name.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetActiveDeviceByName(const std::string& name)
```

EDeepLearningTool::GetActiveDevices

EDeepLearningTool::SetActiveDevices

Devices with which to run or train the model.
Only multiple GPUs with the default engine is supported.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDevice>
GetActiveDevices() const

void SetActiveDevices(const std::vector<Euresys::Open_
eVision::EasyDeepLearning::EDeepLearningDevice>& devices)
```

EDeepLearningTool::SetActiveDevicesById

Sets the devices with which to run the model using their index in the list returned by EDeepLearningTool::AvailableDevices.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetActiveDevicesById(const std::vector<int>& ids)
```

EDeepLearningTool::GetAutoSavePeriod

EDeepLearningTool::SetAutoSavePeriod

Auto-save period. The auto-save period is the number of training iterations between each automatic save of the model.
In order for the model to be automatically saved during the training, the tool must have been loaded or saved to a file. The path where the tool will be saved is either indicated by EDeepLearningTool::Path or the three paths EDeepLearningTool::SettingsPath, EDeepLearningTool::InferenceModelPath, and EDeepLearningTool::TrainingModelPath. If one of these path is empty, the corresponding part of the tool will not be saved automatically during training.
By default, auto save is disabled and the period is equal to 0.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetAutoSavePeriod() const
void SetAutoSavePeriod(int period)
```

EDeepLearningTool::GetAvailableDevices

Get all available device for the current engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDevice>
GetAvailableDevices() const
```

EDeepLearningTool::GetAvailableEngines

Available engines.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
static std::vector<std::string> GetAvailableEngines()
```

EDeepLearningTool::GetBatchSize

EDeepLearningTool::SetBatchSize

Batch size. The batch size is the number of images that are processed together during training and batch inference.
When using multi-GPUs processing (see EDeepLearningTool::GPUIndexes), the batch size is the number of images that each GPU will process at once.
A large batch size will increase the processing speed on GPU but also the memory requirements.
The batch size must be bigger or equal to 1 and it is commonly chosen to be a power of 2.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBatchSize() const
void SetBatchSize(int size)
```

Remarks

If EDeepLearningTool::OptimizeBatchSize is 'true', then the value of this property will be optimized automatically for training or inference according to the situation.
See also EDeepLearningTool::BatchSizeForMaximumInferenceSpeed.

EDeepLearningTool::GetBatchSizeForMaximumInferenceSpeed

Computes the batch size that will maximize the inference speed on NVIDIA GPU. For all other devices, it will return 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBatchSizeForMaximumInferenceSpeed() const
```

Remarks

This value is given as an indication and should not necessarily be used in practice.
You must choose a tradeoff between the overall inference speed (also called the throughput), which is limited by this value, and the time it takes to compute the result of a whole batch (also called the latency), which is minimized by making inference image per image (i.e. a batch size of 1).
The tradeoff depends on your particular application.
Throw EError_DeepLearningToolNotTrained if the tool is not trained.

EDeepLearningTool::GetBestIteration

Iteration at which the minimum validation error was reached. After training, the tool is in the state it was at this best iteration.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetBestIteration() const
```

EDeepLearningTool::Create

Factory method to open a deep learning tool.
Returns the corresponding deep learning tool that must be deleted by the caller.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EDeepLearningTool* Create(
 const std::string& path
 )
```

```
EDeepLearningTool* Create(
 ESerializer* serializer
 )
```

Parameters

path

Path to a deep learning tool.

serializer

A serializer created for reading.

EDeepLearningTool::GetCurrentTrainingFinishedIterations

Number of iterations that are already finished in the current training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetCurrentTrainingFinishedIterations() const
```

EDeepLearningTool::GetCurrentTrainingNumIterations

Total number of training iterations that will be performed during the current training of the deep learning tool.
After the end of the training, this number is the actual number of iterations performed during the training which can be lower than the number of iterations given to the EDeepLearningTool::Train method (for example if the user called EDeepLearningTool::StopTraining).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetCurrentTrainingNumIterations() const
```

EDeepLearningTool::GetCurrentTrainingProgression

Current training progression as a percentage (between 0 and 1).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetCurrentTrainingProgression() const
```

EDeepLearningTool::GetDeterministicTrainingRandomSeed

EDeepLearningTool::SetDeterministicTrainingRandomSeed

Random seed to use when EDeepLearningTool::EnableDeterministicTraining is set to true. The default value is 42.
When EDeepLearningTool::EnableDeterministicTraining, the random seed is generated using a random non-deterministic source.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetDeterministicTrainingRandomSeed() const
void SetDeterministicTrainingRandomSeed(int seed)
```

EDeepLearningTool::GetEnableDeterministicTraining

EDeepLearningTool::SetEnableDeterministicTraining

Whether to use determinitic training algorithm that allows to repeatable training results.
Default value: false.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableDeterministicTraining() const
void SetEnableDeterministicTraining(bool enable)
```

Remarks

Deterministic training is not guarenteed when:
- using multi-core processing is enabled (see Easy::MaxNumberOfProcessingThreads) regardless of the value of EDeepLearningTool::EnableGPU).
- performing several training on the same tool in succession.

EDeepLearningTool::GetEnableGPU

EDeepLearningTool::SetEnableGPU

This property is deprecated.

Deprecated: Use EDeepLearningTool.
Enable the use of a GPU for training and inference of the deep learning tool. If the active engine is not the default engine, EDeepLearningTool::EnableGPU will be false even if the active device has a GPU device type. Setting EDeepLearningTool::EnableGPU to any value when the active engine is not the default one will throw an exception.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetEnableGPU() const
void SetEnableGPU(bool enable)
```

EDeepLearningTool::GetEngine

EDeepLearningTool::SetEngine

Sets the engine with which to execute or train the neural network.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetEngine() const
void SetEngine(const std::string& engineName)
```

EDeepLearningTool::GetExecutionSettings

EDeepLearningTool::SetExecutionSettings

Execution settings of the tool (engine, device(s), inference precision, batch size settings).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EDeepLearningExecutionSettings GetExecutionSettings() const
void SetExecutionSettings(const EDeepLearningExecutionSettings& settings)
```

EDeepLearningTool::GetActiveDevice

Gets the active device. By default, it returns the first active device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningDevice GetActiveDevice(
 int i
 )
```

Parameters

i

Index of the active devices between 0 and EDeepLearningTool::NumActiveDevices.

EDeepLearningTool::GetAvailableDevice

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EDeepLearningDevice GetAvailableDevice(
 int i
 )
```

Parameters

i

-

EDeepLearningTool::GetAvailableDeviceName

Name of the i-th device of the current engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetAvailableDeviceName(
 int i
 )
```

Parameters

i

Index of the device between 0 and EDeepLearningTool - 1.

EDeepLearningTool::GetAvailableDevicesForEngine

Available devices for the specified engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::EDeepLearningDevice>
GetAvailableDevicesForEngine(
 const std::string& engine
 )
```

Parameters

engine

Engine for which to get the devices

EDeepLearningTool::GetAvailableEngineName

Gets the name of the engine given its id.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetAvailableEngineName(
 int id
 )
```

Parameters

id

Index of the available engine between 0 and EDeepLearningTool::NumAvailableEngines - 1.

EDeepLearningTool::GetLabel

Gets the label from its index. If the tool is not trained, the method may throw an exception.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetLabel(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the label

EDeepLearningTool::GetLabelColor

Label color. The label colors affect how the results will be displayed.
The default label colors are the one specified in the dataset used for training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ERGBColor GetLabelColor(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the label between 0 and EDeepLearningTool::NumLabels

EDeepLearningTool::GetLabelOpacity

Gets the label opacity (between 0 for fully transparent and 255 for opaque color). The label opacities affect with the label colors how the results will be displayed.
The default label opacities are the one specified in the dataset used for training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT8 GetLabelOpacity(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the label between 0 and EDeepLearningTool::NumLabels

EDeepLearningTool::GetLabelWeight

Gets the label weight. If the tool is not trained, the label weights are not defined. After training, they are set to the value of the training dataset label weights.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetLabelWeight(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the label

EDeepLearningTool::GetNumPatchesForImage

Number of patches that will be extracted from an input image to perform inference.
For EasyClassify and EasyLocate, this will always be equal to 1.
For EasySegment, the number of patches will depend on the scale, patch size and sampling density parameters.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumPatchesForImage(
 const EBaseROI& sampleImage
 )

int GetNumPatchesForImage(
 int imageWidth,
 int imageHeight
 )
```

Parameters

sampleImage

Image for which to get the number of patch

imageWidth

Width of the image for which to get the number of patch

imageHeight

Height of the image for which to get the number of patch

EDeepLearningTool::GetOptimalNumImagesForBatchSize

Optimal number of images to process together for inference given the batch size and EDeepLearningTool::GetNumPatchesForImage.
In practice, for EasyClassify and EasyLocate, this is the same as the batch size.
For EasySegment, the value will depend on the scale, patch size and sampling density parameters.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetOptimalNumImagesForBatchSize(
 const EBaseROI& sampleImage
 )

int GetOptimalNumImagesForBatchSize(
 int imageWidth,
 int imageHeight
 )
```

Parameters

sampleImage
Image for which to get the optimal number of images
imageWidth
Width of the image for which to get the optimal number of images
imageHeight
Height of the image for which to get the optimal number of images

EDeepLearningTool::GetGPUIndexes

EDeepLearningTool::SetGPUIndexes

This property is deprecated.

Deprecated: use EDeepLearningTool or EDeepLearningTool::ActiveDevices.
Indexes of the GPUs to use for computations.
Using multiple GPUs is only possible when we can process multiple images at once, i.e. during training (EDeepLearningTool::Train) or batch inference.
By default, all the detected GPUs will be used.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<OEV_UINT32> GetGPUIndexes() const
void SetGPUIndexes(const std::vector<OEV_UINT32>& indexes)
```

Remarks

The GPU are indexed from 0 to EDeepLearningTool::NumGPUs - 1.

EDeepLearningTool::HasInferenceModel

Whether the inference model is loaded or created when the tool is not yet trained.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasInferenceModel(
 )
```

EDeepLearningTool::HasTrainingModel

Whether the training model is loaded or created when the tool is not yet trained.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasTrainingModel(
 )
```

EDeepLearningTool::GetImageCacheSize

EDeepLearningTool::SetImageCacheSize

Size in byte of the image cache. The cache is used during training to store reformatted and normalized images. A correctly sized cache can reduce the hard drive accesses and the preprocessing time for each image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT64 GetImageCacheSize() const

void SetImageCacheSize(OEV_UINT64 size)
```

EDeepLearningTool::GetInferenceModelPath

Path of the inference model when loaded or saved using EDeepLearningTool::SaveInferenceModel, EDeepLearningTool::LoadInferenceModel or EDeepLearningTool::SerializeInferenceModel.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetInferenceModelPath() const
```

EDeepLearningTool::GetInferencePrecision

EDeepLearningTool::SetInferencePrecision

Inference precision.
Use EDeepLearningDevice::IsPrecisionSupported to check whether a device support the given precision.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningInferencePrecision
GetInferencePrecision() const
```

```
void SetInferencePrecision(Euresys::Open_
eVision::EasyDeepLearning::EDeepLearningInferencePrecision prec)
```

EDeepLearningTool::InitializeInference

Initializes the neural network for inference (i.e. making predictions with the neural network). You need to pass an image or a vector of images with the same characteristics (image resolution, image type and number of image) as the image or vector of images that you will use to make inference with this tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void InitializeInference(
 int width,
 int height,
 Euresys::Open_eVision::EImageType imageType,
 int numImages
 )

void InitializeInference(
 const EBaseROI& img
 )

void InitializeInference(
 std::vector<const Euresys::Open_eVision::EBaseROI*>& imgList
 )

void InitializeInference(
 std::vector<Euresys::Open_eVision::EImageBW8>& imgList
 )

void InitializeInference(
 std::vector<Euresys::Open_eVision::EImageBW16>& imgList
 )

void InitializeInference(
 std::vector<Euresys::Open_eVision::EImageC24>& imgList
 )
```

Parameters

width
Width of the images for which to prepare the tool for inference

height
Height of the images for which to prepare the tool for inference

imageType
Type of the images for which to prepare the tool for inference

numImages
Width of the images for which to prepare the tool for inference

img
Image for which to prepare the tool for inference

imgList
Vector of images for which to prepare the tool for inference

EDeepLearningTool::IsTrained

Tells whether the deep learning tool has been trained.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsTrained(
 )
```

EDeepLearningTool::IsTraining

Indicates whether the object is currently training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsTraining(
 )
```

EDeepLearningTool::Load

Loads a deep learning tool. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Load(
 const std::string& filePath
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

filePath
File path.

serializer
Pointer to the ESerializer created for reading.

EDeepLearningTool::LoadInferenceModel

Loads the tool's inference model. The inference model is required to apply the tool on new images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void LoadInferenceModel(
 const std::string& filePath
 )
```

Parameters

filePath

File path.

EDeepLearningTool::LoadSettings

Loads the settings of a tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void LoadSettings(
 const std::string& filePath
 )
```

Parameters

filePath

File path.

EDeepLearningTool::LoadTrainingModel

Loads the tool's training model. The training model is required to continue a training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void LoadTrainingModel(
 const std::string& filePath
 )
```

Parameters

filePath

File path.

EDeepLearningTool::GetNumActiveDevices

Gets the number of devices with which to run or train the model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumActiveDevices() const
```

EDeepLearningTool::GetNumAvailableDevices

Available devices for the current EDeepLearningTool::Engine.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumAvailableDevices() const
```

EDeepLearningTool::GetNumAvailableEngines

Number of available engines.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
static int GetNumAvailableEngines()
```

EDeepLearningTool::GetNumGPUs

This property is deprecated.

Deprecated: use EDeepLearningTool with a device type of GPU.
Number of detected NVIDIA GPU.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumGPUs() const
```

EDeepLearningTool::GetNumLabels

Number of labels of this tool. If the tool is not trained, the method will throw an exception.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumLabels() const
```

Remarks

Some type of tools such as EUnsupervisedSegmenter may have no label at all. For these tools, there is a separate API to control the predicted labels.

EDeepLearningTool::GetNumTrainedIterations

Number of iterations that were performed to train this deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumTrainedIterations() const
```

Remarks

This number of iteration may result from the addition of the iterations performed in several calls to EDeepLearningTool::Train.
An iteration can also be called an epoch.

EDeepLearningTool::GetOptimizeBatchSize

EDeepLearningTool::SetOptimizeBatchSize

Indicates whether to optimize the batch size (see EDeepLearningTool::BatchSize) to maximize the training and inference speed according to the engine, the device and the available memory.
Default value is true.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetOptimizeBatchSize() const

void SetOptimizeBatchSize(bool optimize)
```

EDeepLearningTool::GetPath

Path of the tool when loaded or saved using EDeepLearningTool::Save, EDeepLearningTool::Load or EDeepLearningTool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetPath() const
```

EDeepLearningTool::Save

Saves the settings, inference and training model of the deep learning tool.
The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& filePath,
 bool includeTrainingModel
 )

void Save(
 ESerializer* serializer,
 bool includeTrainingModel
 )
```

Parameters

filePath

File path.

includeTrainingModel

Whether to save the training model. The training model is required to continue training the tool.

serializer

Pointer to the ESerializer created for writing.

EDeepLearningTool::SaveInferenceModel

Saves the tool's inference model to a file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SaveInferenceModel(
 const std::string& filePath
 )
```

Parameters

filePath

File path.

EDeepLearningTool::SaveSettings

Saves the settings of the tool to a file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SaveSettings(
 const std::string& filePath
 )
```

Parameters

filePath

File path.

EDeepLearningTool::SaveTrainingModel

Saves the tool's training model to a file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SaveTrainingModel(
 const std::string& filePath
 )
```

Parameters

filePath

File path.

EDeepLearningTool::SerializeInferenceModel

Serializes the inference model of the tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SerializeInferenceModel(
 ESerializer* serializer
 )
```

Parameters

serializer

Pointer to ESerializer

EDeepLearningTool::SerializeSettings

Serializes the tool settings.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer

Pointer to ESerializer

EDeepLearningTool::SerializeTrainingModel

Serializes the training model of the tool.
The training model is necessary to continue training the tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SerializeTrainingModel(
 ESerializer* serializer
 )
```

Parameters

serializer

Pointer to ESerializer

EDeepLearningTool::SetActiveDevice

Sets the device with which to run or train the model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetActiveDevice(
 const EDeepLearningDevice& device
 )
```

Parameters

device

-

Remarks

For CPU device, the number of thread is initialized from Easy

EDeepLearningTool::SetLabel

Changes a label predicted by the tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetLabel(
 OEV_UINT32 index,
 const std::string& label
 )
```

Parameters

index

Index of the label

label

New label

Remarks

Be careful when changing one of the label of a trained tool. It can create incompatibilities with previous metrics or results computed with this tool or with the dataset used to trained the tool.
Some tools may not use this label API. For example EUnsupervisedSegmenter uses EUnsupervisedSegmenter::GoodLabel instead.

EDeepLearningTool::SetLabelColor

Changes a label color.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetLabelColor(
 OEV_UINT32 index,
 const ERGBColor& c
 )
```

Parameters

index

Index of the label between 0 and EDeepLearningTool::NumLabels

c

New color for the specified label

EDeepLearningTool::SetLabelOpacity

Changes a label opacity (between 0 for fully transparent and 255 for opaque color).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetLabelOpacity(
 OEV_UINT32 index,
 OEV_UINT8 opacity
 )
```

Parameters

index

Index of the label between 0 and EDeepLearningTool::NumLabels

opacity

New opacity for the specified label

EDeepLearningTool::SetLabelWeight

Sets the label weight. If the tool is not trained, the label weights are not defined. After training, they are set to the value of the training dataset label weights.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SetLabelWeight(
 OEV_UINT32 index,
 float weight
 )
```

Parameters

index

Index of the label

weight

Weight

EDeepLearningTool::GetSettingsPath

Path of the settings when loaded or saved using EDeepLearningTool::SaveSettings, EDeepLearningTool::LoadSettings or EDeepLearningTool::SerializeSettings.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetSettingsPath() const
```

EDeepLearningTool::StopTraining

Stops training and returns the last completed iteration. If the parameter 'wait' is set to true, the method will wait for the training thread to completely stop. Otherwise, the method will return immediately.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int StopTraining(
 bool wait
 )
```

Parameters

wait

Whether to wait for the training to completely stop

EDeepLearningTool::GetToolType

Type of the deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetToolType() const
```

EDeepLearningTool::Train

Trains the EDeepLearningTool with the given dataset for the specified number of iterations. At the end of the training, the deep learning tool is in the state it was at the iteration that gave the minimum validation error. See EDeepLearningTool::BestIteration.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Train(
 EClassificationDataset& dataset,
 int iterations
 )

void Train(
 EClassificationDataset& dataset,
 const EDataAugmentation& dataAugmentation,
 int iterations
 )

void Train(
 EClassificationDataset& trainingDataset,
 EClassificationDataset& validationDataset,
 int iterations
 )

void Train(
 EClassificationDataset& trainingDataset,
 EClassificationDataset& validationDataset,
 const EDataAugmentation& dataAugmentation,
 int iterations
 )
```

Parameters

dataset

EClassificationDataset with which to train and validate the deep learning tool

iterations

Number of iterations for training.

dataAugmentation

Data augmentation to use during training

trainingDataset

EClassificationDataset with which to train the deep learning tool

validationDataset

EClassificationDataset with which to validate the deep learning tool

EDeepLearningTool::GetTrainingModelPath

Path of the training model when loaded or saved using EDeepLearningTool::SaveTrainingModel, EDeepLearningTool::LoadTrainingModel or EDeepLearningTool::SerializeTrainingModel.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetTrainingModelPath() const
```

EDeepLearningTool::UnloadInferenceModel

Unloads the inference model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UnloadInferenceModel(
 )
```

EDeepLearningTool::UnloadModels

Unloads the inference and training model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UnloadModels(
 )
```

EDeepLearningTool::UnloadTrainingModel

Unloads the training model.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void UnloadTrainingModel(
 )
```

EDeepLearningTool::WaitForIterationCompletion

Waits until an iteration is complete. A call to this method will block the calling thread until a training iteration in the training thread is finished. This method returns the number of trained iterations.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int WaitForIterationCompletion(
 )
```

EDeepLearningTool::WaitForTrainingCompletion

Waits until the training is complete or the timeout is expired. A call to this method will block the calling thread for the shortest time between the timeout and the time it takes for the training to complete.
A negative timeout means that the method will wait until the training is complete.
The default value is set to -1.
The method returns the number of trained iterations.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int WaitForTrainingCompletion(
 int timeout
 )
```

Parameters

timeout

Timeout in second

4.80. EDepthMap Class

Represents a generic DepthMap type interface.

Derived Class(es):EDepthMap16EDepthMap32fEDepthMap8

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. Overwrites if exists.
Clear	Clears the depth map: replaces all pixels with undefined value
ClearMetadata	Deletes all metadata.
ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Pixel coordinates.
ConvertCoordinatesPixelToMap	Converts Pixel coordinates to 3D Map coordinates.
DeleteMetadata	Deletes an existing metadata. Throws an exception if it does not exist.
Draw	Draws a DepthMap in a device context.
DrawImage	Displays the internal image buffer
GetAxisSystemType	Manage the axis coordinate system.
GetBufferPtr	Retrieves the pointer of the pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer of the pixel buffer.

GetHeight	Access depth map Height.
GetMetadata	Returns string value of the given metadata. Throws an exception if it does not exist.
GetRowPitch	Returns the buffer row pitch.
GetType	Returns the image type.
GetWidth	Access depth map Width.
GetZResolution	Access the Z Resolution (depth units per grey scale value).
GetZValue	Gets the Z value of a pixel.
IsVoid	Tests if the EDepthMap object has a size of zero.
Load	Restores the EDepthMap stored in the given Open eVision file.
LoadImage	Restores the EDepthMap image stored in the given image file.
LoadImageAndMetadata	Loads the image and the metadata from a file in JSON format.
LoadMetadata	Loads the metadata from a file in JSON format.
ModifyMetadata	Changes the value of an existing metadata. Throws an exception if the metadata does not exist.
Save	Saves the EDepthMap object to the given Open eVision file.
SaveImage	Saves the EDepthMap image to the given image file.
SaveImageAndMetadata	Saves the image and the metadata to a JSON format file.
SaveJpeg	Saves the EDepthMap object to the given image file, in JPEG format.
SaveJpeg2K	Saves the EDepthMap object to the given image file, in JPEG 2000 format.
SaveMetadata	Saves the metadata to a file in JSON format
SetAxisSystemType	Manage the axis coordinate system.
SetBufferPtr	Sets the pointer to an externally allocated buffer.
SetHeight	Access depth map Height.
SetSize	Sets the width and height of a DepthMap.
SetWidth	Access depth map Width.
SetZResolution	Access the Z Resolution (depth units per grey scale value).

EDepthMap::AddMetadata

Adds a metadata key (name) and value. Overwrites if exists.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EDepthMap::GetAxisSystemType

EDepthMap::SetAxisSystemType

Manage the axis coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EAxisSystemType GetAxisSystemType() const
void SetAxisSystemType(Euresys::Open_eVision::Easy3D::EAxisSystemType baseType)
```

EDepthMap::Clear

Clears the depth map: replaces all pixels with undefined value

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Clear(
 )
```

EDepthMap::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ClearMetadata(
 )
```

EDepthMap::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D

The Map X coordinate.

y3D

The Map Y coordinate.

xBuffer

The returned Pixel X coordinate.

yBuffer

The returned Pixel Y coordinate.

EDepthMap::ConvertCoordinatesPixelToMap

Converts Pixel coordinates to 3D Map coordinates.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EDepthMap::DeleteMetadata

Deletes an existing metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key
The name of an existing metadata.

EDepthMap::Draw

Draws a DepthMap in a device context.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the depthmap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A DepthMap can be drawn (its pixels rendered) using a device context.
EDepthMap::Draw takes the axis coordinate system in account. See EDepthMap.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context.
EDepthMap::DrawImage does not takes the axis coordinate system in account. It displays the internal image buffer without flipping the axis. See EDepthMap::Draw method for an alternative drawing method.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap::GetBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )

void* GetBufferPtr(
 int x,
 int y
 )
```

```
const void* GetBufferPtr(
 )

const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EDepthMap::GetCheckedBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )

const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function checks the value of the parameters.

EDepthMap::GetMetadata

Returns string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EDepthMap::GetZValue

Gets the Z value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap::GetHeight

EDepthMap::SetHeight

Access depth map Height.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetHeight() const

void SetHeight(int height)
```

EDepthMap::IsVoid

Tests if the EDepthMap object has a size of zero.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsVoid(
 )
```

Remarks

Returns true if the depthmap size is zero.

EDepthMap::Load

Restores the EDepthMap stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
```

Parameters

path

Full path of the file.

Remarks

When loading, the depth map is resized if needed.
This function restores the depth map attributes.

EDepthMap::LoadImage

Restores the EDepthMap image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the depth map is resized if need be.
This function does not restore the depth map attributes, only the image associated with the EDepthMap is updated.

EDepthMap::LoadImageAndMetadata

Loads the image and the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata
 )
```

Parameters

pathImage

Full path to the image file.

pathMetadata

Full path to the metadata file.

EDepthMap::LoadMetadata

Loads the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EDepthMap::ModifyMetadata

Changes the value of an existing metadata.
Throws an exception if the metadata does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of an existing metadata.

value

The value for the given metadata.

EDepthMap::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRowPitch() const
```

EDepthMap::Save

Saves the EDepthMap object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

Remarks

This function saves the EDepthMap in a Open eVision file.
This function stores the depth map attributes.

EDepthMap::SaveImage

Saves the EDepthMap image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This function saves the image associated to EDepthMap in a standard image file and thus does not store depth map attributes.

EDepthMap::SaveImageAndMetadata

Saves the image and the metadata to a JSON format file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

pathImage

The full path to the destination image file.

pathMetadata

The full path to the destination metadata file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

EDepthMap::SaveJpeg

Saves the EDepthMap object to the given image file, in JPEG format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveJpeg(
 const std::string& path,
 int quality
 )
```

Parameters

path
The full path of the destination file.

quality
JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

EDepthMap::SaveJpeg2K

Saves the EDepthMap object to the given image file, in JPEG 2000 format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveJpeg2K(
 const std::string& path,
 int quality
 )
```

Parameters

path
The full path of the destination file.

quality
JPEG 2000 quality, between 1 and 512.
The default value is 16.

EDepthMap::SaveMetadata

Saves the metadata to a file in JSON format

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EDepthMap::SetBufferPtr

Sets the pointer to an externally allocated buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EDepthMap::SetBufferPtr.

EDepthMap::SetSize

Sets the width and height of a DepthMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetSize(
 int width,
 int height
 )

void SetSize(
 const EDepthMap& other
 )
```

Parameters

width

The new requested DepthMap width.

height

The new requested DepthMap height.

other

The other DepthMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new buffer (deallocate the old buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new buffer and setting its size creates a 4-byte aligned buffer, by default.

EDepthMap::GetType

Returns the image type.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::EImageType GetType() const
```

EDepthMap::GetWidth

EDepthMap::SetWidth

Access depth map Width.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetWidth() const

void SetWidth(int width)
```

EDepthMap::GetZResolution

EDepthMap::SetZResolution

Access the Z Resolution (depth units per grey scale value).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.81. EDepthMap16 Class

Represents a EDepthMap with an 16-bit pixel internal representation.

Base Class:EDepthMap
Derived Class(es):EGrabberDepthMap16
Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. Overwrites if exists.
AsEImage	Casts the EDepthMap16 to an EImageBW16 to use with the Open eVision 2D tools.
Clear	Clears the depth map: replaces all pixels with undefined value
ClearMetadata	Deletes all metadata.
ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Pixel coordinates.
ConvertCoordinatesPixelToMap	Converts Pixel coordinates to 3D Map coordinates.
CopyMetadataTo	Copies all metadata to another EDepthMap16.
DeleteMetadata	Deletes an existing metadata. Throws an exception if it does not exist.
Draw	Draws a DepthMap in a device context.
DrawImage	Displays the internal image buffer
EDepthMap16	Creates a 16 bits EDepthMap.
FillUndefinedPixels	Fills undefined pixels, used to fill the "holes" in the depth map.
FillUndefinedPixelsWithMedian	Fills undefined pixels, used to fill the "holes" in the depth map using a median rectangular kernel of odd size.
GetAxisSystemType	Manage the axis coordinate system.

GetBufferPtr	Retrieves the pointer of the pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer of the pixel buffer.
GetHeight	Access depth map Height.
GetMetadata	Returns string value of the given metadata. Throws an exception if it does not exist.
GetPixel	Gets the value of a pixel.
GetRowPitch	Returns the buffer row pitch.
GetType	Returns the image type.
GetUndefinedValue	Returns the Undefined value. That value is used to mark pixels with no valid depth value.
GetWidth	Access depth map Width.
GetZResolution	Access the Z Resolution (depth units per grey scale value).
GetZValue	Gets Z value of a pixel.
IsVoid	Tests if the EDepthMap16 object has a size of zero.
Load	Restores the EDepthMap16 stored in the given Open eVision file.
LoadImage	Restores the EDepthMap16 image stored in the given image file.
LoadImageAndMetadata	Loads the image and the metadata from a file in JSON format.
LoadMetadata	Loads the metadata from a file in JSON format.
ModifyMetadata	Changes the value of an existing metadata. Throws an exception if the metadata does not exist.
operator=	Assignment operator.
Save	Saves the EDepthMap16 object to the given Open eVision file.
SaveImage	Saves the EDepthMap16 image to the given image file.
SaveImageAndMetadata	Saves the image and the metadata to a JSON format file.
SaveJpeg	Saves the EDepthMap16 object to the given image file, in JPEG format.
SaveJpeg2K	Saves the EDepthMap16 object to the given image file, in JPEG 2000 format.
SaveMetadata	Saves the metadata to a file in JSON format
SetAxisSystemType	Manage the axis coordinate system.
SetBufferPtr	Sets the pointer to an externally allocated buffer.
SetHeight	Access depth map Height.
SetPixel	Sets the value of a pixel.
SetSize	Sets the width and height of a DepthMap.
SetWidth	Access depth map Width.
SetZResolution	Access the Z Resolution (depth units per grey scale value).

EDepthMap16::AddMetadata

Adds a metadata key (name) and value. Overwrites if exists.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key
The name of the metadata. Names are unique.

value
The value for the given metadata.

EDepthMap16::AsEImage

Casts the EDepthMap16 to an EImageBW16 to use with the Open eVision 2D tools.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EImageBW16& AsEImage(
 )
```

EDepthMap16::GetAxisSystemType

EDepthMap16::SetAxisSystemType

Manage the axis coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::EAxisSystemType GetAxisSystemType() const
void SetAxisSystemType(Euresys::Open_eVision::Easy3D::EAxisSystemType baseType)
```

EDepthMap16::Clear

Clears the depth map: replaces all pixels with undefined value

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Clear(
 )
```

EDepthMap16::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearMetadata(
 )
```

EDepthMap16::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D
The Map X coordinate.

y3D
The Map Y coordinate.

xBuffer
The returned Pixel X coordinate.

yBuffer
The returned Pixel Y coordinate.

EDepthMap16::ConvertCoordinatesPixelToMap

Converts Pixel coordinates to 3D Map coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EDepthMap16::CopyMetadataTo

Copies all metadata to another EDepthMap16.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void CopyMetadataTo(
 EDepthMap16& other
 )
```

Parameters

other
The destination EDepthMap16.

EDepthMap16::DeleteMetadata

Deletes an existing metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EDepthMap16::Draw

Draws a DepthMap in a device context.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the depthmap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A DepthMap can be drawn (its pixels rendered) using a device context.
EDepthMap16::Draw takes the axis coordinate system in account. See EDepthMap16.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap16::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context. EDepthMap16::DrawImage does not takes the axis coordinate system in account. It displays the internal image buffer without flipping the axis. See EDepthMap16::Draw method for an alternative drawing method.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap16::EDepthMap16

Creates a 16 bits EDepthMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EDepthMap16(
 )
void EDepthMap16(
 int width,
 int height
 )
void EDepthMap16(
 const EDepthMap16& other
 )
```

Parameters

width
The width of the new depth map.
height
The height of the new depth map.
other
Another depth map.

EDepthMap16::FillUndefinedPixels

Fills undefined pixels, used to fill the "holes" in the depth map.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillUndefinedPixels(
 EDepthMap16& outMap,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

outMap

The destination depth map.

direction

Direction in which the undefined pixels are filled in a depthmap from EFillUndefinedPixelsDirection.

method

Which values used to fill the undefined pixels in a depthmap from EFillUndefinedPixelsMethod.

EDepthMap16::FillUndefinedPixelsWithMedian

Fills undefined pixels, used to fill the "holes" in the depth map using a median rectangular kernel of odd size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillUndefinedPixelsWithMedian(
 EDepthMap16& outMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

outMap

The destination depth map

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 2).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth).

EDepthMap16::GetBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )

void* GetBufferPtr(
 int x,
 int y
 )

const void* GetBufferPtr(
 )

const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EDepthMap16::GetCheckedBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )

const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function checks the value of the parameters.

EDepthMap16::GetMetadata

Returns string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EDepthMap16::GetPixel

Gets the value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth16 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap16::GetZValue

Gets Z value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap16::GetHeight

EDepthMap16::SetHeight

Access depth map Height.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetHeight() const

void SetHeight(int height)
```

EDepthMap16::IsVoid

Tests if the EDepthMap16 object has a size of zero.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsVoid(
 )
```

Remarks

Returns true if the depthmap size is zero.

EDepthMap16::Load

Restores the EDepthMap16 stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path of the file.

serializer

-

Remarks

When loading, the depth map is resized if needed.
This function restores the depth map attributes.

EDepthMap16::LoadImage

Restores the EDepthMap16 image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the depth map is resized if need be.
This function does not restore the depth map attributes, only the image associated with the EDepthMap16 is updated.

EDepthMap16::LoadImageAndMetadata

Loads the image and the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata
 )
```

Parameters

pathImage
Full path to the image file.

pathMetadata
Full path to the metadata file.

EDepthMap16::LoadMetadata

Loads the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path
Full path to the file.

EDepthMap16::ModifyMetadata

Changes the value of an existing metadata.
Throws an exception if the metadata does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key
The name of an existing metadata.

value
The value for the given metadata.

EDepthMap16::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepthMap16& operator=(
 const EDepthMap16& other
 )
```

Parameters

other

The source EDepthMap16.

EDepthMap16::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRowPitch() const
```

EDepthMap16::Save

Saves the EDepthMap16 object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This function saves the EDepthMap16 in a Open eVision file.
This function stores the depth map attributes.

EDepthMap16::SaveImage

Saves the EDepthMap16 image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This function saves the image associated to EDepthMap16 in a standard image file and thus does not store depth map attributes.

EDepthMap16::SaveImageAndMetadata

Saves the image and the metadata to a JSON format file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

pathImage

The full path to the destination image file.

pathMetadata

The full path to the destination metadata file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

EDepthMap16::SaveJpeg

Saves the EDepthMap16 object to the given image file, in JPEG format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveJpeg(
 const std::string& path,
 int quality
 )
```

Parameters

path

The full path of the destination file.

quality

JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

EDepthMap16::SaveJpeg2K

Saves the EDepthMap16 object to the given image file, in JPEG 2000 format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveJpeg2K(
 const std::string& path,
 int quality
 )
```

Parameters

path

The full path of the destination file.

quality

JPEG 2000 quality, between 1 and 512.
The default value is 16.

EDepthMap16::SaveMetadata

Saves the metadata to a file in JSON format

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EDepthMap16::SetBufferPtr

Sets the pointer to an externally allocated buffer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EDepthMap16::SetBufferPtr.

EDepthMap16::SetPixel

Sets the value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetPixel(
 EDepth16 value,
 int x,
 int y
 )
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMap16::SetSize

Sets the width and height of a DepthMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetSize(
 int width,
 int height
 )
void SetSize(
 const EDepthMap& other
 )
```

Parameters

width

The new requested DepthMap width.

height

The new requested DepthMap height.

other

The other DepthMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new buffer (deallocate the old buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new buffer and setting its size creates a 4-byte aligned buffer, by default.

EDepthMap16::GetType

Returns the image type.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::EImageType GetType() const
```

EDepthMap16::GetUndefinedValue

Returns the Undefined value. That value is used to mark pixels with no valid depth value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth16 GetUndefinedValue() const
```

EDepthMap16::GetWidth

EDepthMap16::SetWidth

Access depth map Width.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetWidth() const
void SetWidth(int width)
```

EDepthMap16::GetZResolution

EDepthMap16::SetZResolution

Access the Z Resolution (depth units per grey scale value).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.82. EDepthMap32f Class

Represents a EDepthMap with an 32-bit pixel internal representation.

Base Class:EDepthMap

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. Overwrites if exists.
AsEImage	Casts the EDepthMap32f to a 32 bits gray scale image
Clear	Clears the depth map: replaces all pixels with undefined value
ClearMetadata	Deletes all metadata.
ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Pixel coordinates.
ConvertCoordinatesPixelToMap	Converts Pixel coordinates to 3D Map coordinates.
CopyMetadataTo	Copies all metadata to another EDepthMap32f.
DeleteMetadata	Deletes an existing metadata. Throws an exception if it does not exist.
Draw	Draws a DepthMap in a device context.
DrawImage	Displays the internal image buffer
EDepthMap32f	Creates a 32 bits EDepthMap.
FillUndefinedPixels	Fills undefined pixels, used to fill the "holes" in the depth map.
FillUndefinedPixelsWithMedian	Fills undefined pixels, used to fill the "holes" in the depth map using a median rectangular kernel of odd size.
GetAxisSystemType	Manage the axis coordinate system.
GetBufferPtr	Retrieves the pointer of the pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer of the pixel buffer.
GetHeight	Access depth map Height.
GetMetadata	Returns string value of the given metadata. Throws an exception if it does not exist.
GetPixel	Gets the value of a pixel.
GetRowPitch	Returns the buffer row pitch.
GetType	Returns the image type.
GetUndefinedValue	Returns the Undefined value. That value is used to mark pixels with no valid depth value.
GetWidth	Access depth map Width.

GetZResolution	Access the Z Resolution (depth units per grey scale value).
GetZValue	Gets Z value of a pixel.
IsVoid	Tests if the EDepthMap32f object has a size of zero.
Load	Restores the EDepthMap32f stored in the given Open eVision file.
LoadImage	Restores the EDepthMap32f image stored in the given image file.
LoadImageAndMetadata	Loads the image and the metadata from a file in JSON format.
LoadMetadata	Loads the metadata from a file in JSON format.
ModifyMetadata	Changes the value of an existing metadata. Throws an exception if the metadata does not exist.
operator=	Assignment operator.
Save	Saves the EDepthMap32f object to the given Open eVision file.
SaveImage	Saves the EDepthMap32f image to the given image file.
SaveImageAndMetadata	Saves the image and the metadata to a JSON format file.
SaveJpeg	Saves the EDepthMap32f object to the given image file, in JPEG format.
SaveJpeg2K	Saves the EDepthMap32f object to the given image file, in JPEG 2000 format.
SaveMetadata	Saves the metadata to a file in JSON format
SetAxisSystemType	Manage the axis coordinate system.
SetBufferPtr	Sets the pointer to an externally allocated buffer.
SetHeight	Access depth map Height.
SetPixel	Sets the value of a pixel.
SetSize	Sets the width and height of a DepthMap.
SetWidth	Access depth map Width.
SetZResolution	Access the Z Resolution (depth units per grey scale value).

EDepthMap32f::AddMetadata

Adds a metadata key (name) and value. Overwrites if exists.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EDepthMap32f::AsEImage

Casts the EDepthMap32f to a 32 bits gray scale image

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EImageBW32f& AsEImage(
 )
```

EDepthMap32f::GetAxisSystemType

EDepthMap32f::SetAxisSystemType

Manage the axis coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EAxisSystemType GetAxisSystemType() const
void SetAxisSystemType(Euresys::Open_eVision::Easy3D::EAxisSystemType baseType)
```

EDepthMap32f::Clear

Clears the depth map: replaces all pixels with undefined value

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Clear(
 )
```

EDepthMap32f::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ClearMetadata(
 )
```

EDepthMap32f::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D
The Map X coordinate.

y3D
The Map Y coordinate.

xBuffer
The returned Pixel X coordinate.

yBuffer
The returned Pixel Y coordinate.

EDepthMap32f::ConvertCoordinatesPixelToMap

Converts Pixel coordinates to 3D Map coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EDepthMap32f::CopyMetadataTo

Copies all metadata to another EDepthMap32f.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void CopyMetadataTo(
 EDepthMap32f& other
 )
```

Parameters

other
The destination EDepthMap32f.

EDepthMap32f::DeleteMetadata

Deletes an existing metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key
The name of an existing metadata.

EDepthMap32f::Draw

Draws a DepthMap in a device context.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the depthmap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A DepthMap can be drawn (its pixels rendered) using a device context. EDepthMap32f::Draw takes the axis coordinate system in account. See EDepthMap32f.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap32f::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context.
EDepthMap32f::DrawImage does not takes the axis coordinate system in account. It displays the internal image buffer without flipping the axis. See EDepthMap32f::Draw method for an alternative drawing method.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap32f::EDepthMap32f

Creates a 32 bits EDepthMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EDepthMap32f(
 )

void EDepthMap32f(
 int width,
 int height
 )
```

```
void EDepthMap32f(
 const EDepthMap32f& other
 )
```

Parameters

width

The width of the new depth map.

height

The height of the new depth map.

other

Another depth map.

EDepthMap32f::FillUndefinedPixels

Fills undefined pixels, used to fill the "holes" in the depth map.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillUndefinedPixels(
 EDepthMap32f& outMap,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

outMap

The destination depth map

direction

Direction in which the undefined pixels are filled in a depthmap from EFillUndefinedPixelsDirection.

method

Which values used to fill the undefined pixels in a depthmap from EFillUndefinedPixelsMethod.

EDepthMap32f::FillUndefinedPixelsWithMedian

Fills undefined pixels, used to fill the "holes" in the depth map using a median rectangular kernel of odd size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillUndefinedPixelsWithMedian(
 EDepthMap32f& outMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

outMap

The destination depth map

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 2).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth).

EDepthMap32f::GetBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )
void* GetBufferPtr(
 int x,
 int y
 )
const void* GetBufferPtr(
 )
const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EDepthMap32f::GetCheckedBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )
const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function checks the value of the parameters.

EDepthMap32f::GetMetadata

Returns string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EDepthMap32f::GetPixel

Gets the value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth32f GetPixel(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap32f::GetZValue

Gets Z value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap32f::GetHeight

EDepthMap32f::SetHeight

Access depth map Height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetHeight() const

void SetHeight(int height)
```

EDepthMap32f::IsVoid

Tests if the EDepthMap32f object has a size of zero.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsVoid(
 )
```

Remarks

Returns true if the depthmap size is zero.

EDepthMap32f::Load

Restores the EDepthMap32f stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path of the file.

serializer

-

Remarks

When loading, the depth map is resized if needed.
This function restores the depth map attributes.

EDepthMap32f::LoadImage

Restores the EDepthMap32f image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the depth map is resized if need be.
This function does not restore the depth map attributes, only the image associated with the EDepthMap32f is updated.

EDepthMap32f::LoadImageAndMetadata

Loads the image and the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata
 )
```

Parameters

pathImage

Full path to the image file.

pathMetadata

Full path to the metadata file.

EDepthMap32f::LoadMetadata

Loads the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EDepthMap32f::ModifyMetadata

Changes the value of an existing metadata.
Throws an exception if the metadata does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of an existing metadata.

value

The value for the given metadata.

EDepthMap32f::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepthMap32f& operator=(
 const EDepthMap32f& other
 )
```

Parameters

other

The source EDepthMap32f.

EDepthMap32f::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetRowPitch() const
```

EDepthMap32f::Save

Saves the EDepthMap32f object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This function saves the EDepthMap32f in a Open eVision file.
This function stores the depth map attributes.

EDepthMap32f::SaveImage

Saves the EDepthMap32f image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This function saves the image associated to EDepthMap32f in a standard image file and thus does not store depth map attributes.

EDepthMap32f::SaveImageAndMetadata

Saves the image and the metadata to a JSON format file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

pathImage

The full path to the destination image file.

pathMetadata

The full path to the destination metadata file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

EDepthMap32f::SaveJpeg

Saves the EDepthMap32f object to the given image file, in JPEG format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveJpeg(
 const std::string& path,
 int quality
 )
```

Parameters

path
The full path of the destination file.

quality
JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

EDepthMap32f::SaveJpeg2K

Saves the EDepthMap32f object to the given image file, in JPEG 2000 format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveJpeg2K(
 const std::string& path,
 int quality
 )
```

Parameters

path
The full path of the destination file.

quality
JPEG 2000 quality, between 1 and 512.
The default value is 16.

EDepthMap32f::SaveMetadata

Saves the metadata to a file in JSON format

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path
The full path to the destination file.

EDepthMap32f::SetBufferPtr

Sets the pointer to an externally allocated buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EDepthMap32f::SetBufferPtr.

EDepthMap32f::SetPixel

Sets the value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPixel(
 EDepth32f value,
 int x,
 int y
 )
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMap32f::SetSize

Sets the width and height of a DepthMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetSize(
 int width,
 int height
 )
void SetSize(
 const EDepthMap& other
 )
```

Parameters

width

The new requested DepthMap width.

height

The new requested DepthMap height.

other

The other DepthMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new buffer (deallocate the old buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new buffer and setting its size creates a 4-byte aligned buffer, by default.

EDepthMap32f::GetType

Returns the image type.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::EImageType GetType() const
```

EDepthMap32f::GetUndefinedValue

Returns the Undefined value. That value is used to mark pixels with no valid depth value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepth32f GetUndefinedValue() const
```

EDepthMap32f::GetWidth

EDepthMap32f::SetWidth

Access depth map Width.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetWidth() const
void SetWidth(int width)
```

EDepthMap32f::GetZResolution

EDepthMap32f::SetZResolution

Access the Z Resolution (depth units per grey scale value).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetZResolution() const
void SetZResolution(float resolution)
```

4.83. EDepthMap8 Class

Represents a EDepthMap with an internal 8-bit pixel representation.

Base Class:EDepthMap
Derived Class(es):EGrabberDepthMap8
Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. Overwrites if exists.
AsEImage	Casts the EDepthMap8 to an EImageBW8 to use with the Open eVision 2D tools.
Clear	Clears the depth map: replaces all pixels with undefined value
ClearMetadata	Deletes all metadata.
ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Pixel coordinates.

ConvertCoordinatesPixelToMap	Converts Pixel coordinates to 3D Map coordinates.
CopyMetadataTo	Copies all metadata to another EDepthMap8.
DeleteMetadata	Deletes an existing metadata. Throws an exception if it does not exist.
Draw	Draws a DepthMap in a device context.
DrawImage	Displays the internal image buffer
EDepthMap8	Creates a 8 bits EDepthMap.
FillUndefinedPixels	Fills undefined pixels, used to fill the "holes" in the depth map.
FillUndefinedPixelsWithMedian	Fills undefined pixels, used to fill the "holes" in the depth map using a median rectangular kernel of odd size.
GetAxisSystemType	Manage the axis coordinate system.
GetBufferPtr	Retrieves the pointer of the pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer of the pixel buffer.
GetHeight	Access depth map Height.
GetMetadata	Returns string value of the given metadata. Throws an exception if it does not exist.
GetPixel	Gets the value of a pixel.
GetRowPitch	Returns the buffer row pitch.
GetType	Returns the image type.
GetUndefinedValue	Returns the Undefined value. That value is used to mark pixels with no valid depth value.
GetWidth	Access depth map Width.
GetZResolution	Access the Z Resolution (depth units per grey scale value).
GetZValue	Gets Z value of a pixel.
IsVoid	Tests if the EDepthMap8 object has a size of zero.
Load	Restores the EDepthMap8 stored in the given Open eVision file.
LoadImage	Restores the EDepthMap8 image stored in the given image file.
LoadImageAndMetadata	Loads the image and the metadata from a file in JSON format.
LoadMetadata	Loads the metadata from a file in JSON format.
ModifyMetadata	Changes the value of an existing metadata. Throws an exception if the metadata does not exist.
operator=	Assignment operator.
Save	Saves the EDepthMap8 object to the given Open eVision file.
SaveImage	Saves the EDepthMap8 image to the given image file.
SaveImageAndMetadata	Saves the image and the metadata to a JSON format file.

SaveJpeg	Saves the EDepthMap8 object to the given image file, in JPEG format.
SaveJpeg2K	Saves the EDepthMap8 object to the given image file, in JPEG 2000 format.
SaveMetadata	Saves the metadata to a file in JSON format
SetAxisSystemType	Manage the axis coordinate system.
SetBufferPtr	Sets the pointer to an externally allocated buffer.
SetHeight	Access depth map Height.
SetPixel	Sets the value of a pixel.
SetSize	Sets the width and height of a DepthMap.
SetWidth	Access depth map Width.
SetZResolution	Access the Z Resolution (depth units per grey scale value).

EDepthMap8::AddMetadata

Adds a metadata key (name) and value. Overwrites if exists.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EDepthMap8::AsEImage

Casts the EDepthMap8 to an EImageBW8 to use with the Open eVision 2D tools.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EImageBW8& AsEImage(
 )
```

EDepthMap8::GetAxisSystemType

EDepthMap8::SetAxisSystemType

Manage the axis coordinate system.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::EAxisSystemType GetAxisSystemType() const
void SetAxisSystemType(Euresys::Open_eVision::Easy3D::EAxisSystemType baseType)
```

EDepthMap8::Clear

Clears the depth map: replaces all pixels with undefined value

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Clear(
 )
```

EDepthMap8::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearMetadata(
 )
```

EDepthMap8::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D
The Map X coordinate.

y3D
The Map Y coordinate.

xBuffer
The returned Pixel X coordinate.

yBuffer
The returned Pixel Y coordinate.

EDepthMap8::ConvertCoordinatesPixelToMap

Converts Pixel coordinates to 3D Map coordinates.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EDepthMap8::CopyMetadataTo

Copies all metadata to another EDepthMap8.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void CopyMetadataTo(
 EDepthMap8& other
 )
```

Parameters

other

The destination EDepthMap8.

EDepthMap8::DeleteMetadata

Deletes an existing metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EDepthMap8::Draw

Draws a DepthMap in a device context.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the depthmap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A DepthMap can be drawn (its pixels rendered) using a device context.
EDepthMap8::Draw takes the axis coordinate system in account. See EDepthMap8.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap8::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context.
EDepthMap8::DrawImage does not takes the axis coordinate system in account. It displays the internal image buffer without flipping the axis. See EDepthMap8::Draw method for an alternative drawing method.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EDepthMap8::EDepthMap8

Creates a 8 bits EDepthMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EDepthMap8(
 )

void EDepthMap8(
 int width,
 int height
 )
```

```
void EDepthMap8(
 const EDepthMap8& other
 )
```

Parameters

width

The width of the new depth map.

height

The height of the new depth map.

other

Another depth map.

EDepthMap8::FillUndefinedPixels

Fills undefined pixels, used to fill the "holes" in the depth map.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillUndefinedPixels(
 EDepthMap8& outMap,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

outMap

The destination depth map.

direction

Direction in which the undefined pixels are filled in a depthmap from EFillUndefinedPixelsDirection.

method

Which values used to fill the undefined pixels in a depthmap from EFillUndefinedPixelsMethod.

EDepthMap8::FillUndefinedPixelsWithMedian

Fills undefined pixels, used to fill the "holes" in the depth map using a median rectangular kernel of odd size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillUndefinedPixelsWithMedian(
 EDepthMap8& outMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

outMap

The destination depth map

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 2).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth).

EDepthMap8::GetBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )
void* GetBufferPtr(
 int x,
 int y
 )
const void* GetBufferPtr(
 )
const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EDepthMap8::GetCheckedBufferPtr

Retrieves the pointer of the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )
const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

Remarks

This function checks the value of the parameters.

EDepthMap8::GetMetadata

Returns string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EDepthMap8::GetPixel

Gets the value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth8 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap8::GetZValue

Gets Z value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EDepthMap8::GetHeight

EDepthMap8::SetHeight

Access depth map Height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetHeight() const

void SetHeight(int height)
```

EDepthMap8::IsVoid

Tests if the EDepthMap8 object has a size of zero.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsVoid(
 )
```

Remarks

Returns true if the depthmap size is zero.

EDepthMap8::Load

Restores the EDepthMap8 stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path of the file.

serializer

-

Remarks

When loading, the depth map is resized if needed.
This function restores the depth map attributes.

EDepthMap8::LoadImage

Restores the EDepthMap8 image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the depth map is resized if need be.
This function does not restore the depth map attributes, only the image associated with the EDepthMap8 is updated.

EDepthMap8::LoadImageAndMetadata

Loads the image and the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata
 )
```

Parameters

pathImage

Full path to the image file.

pathMetadata

Full path to the metadata file.

EDepthMap8::LoadMetadata

Loads the metadata from a file in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EDepthMap8::ModifyMetadata

Changes the value of an existing metadata.
Throws an exception if the metadata does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of an existing metadata.

value

The value for the given metadata.

EDepthMap8::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepthMap8& operator=(
 const EDepthMap8& other
 )
```

Parameters

other

The source EDepthMap8.

EDepthMap8::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetRowPitch() const
```

EDepthMap8::Save

Saves the EDepthMap8 object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This function saves the EDepthMap8 in a Open eVision file.
This function stores the depth map attributes.

EDepthMap8::SaveImage

Saves the EDepthMap8 image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This function saves the image associated to EDepthMap8 in a standard image file and thus does not store depth map attributes.

EDepthMap8::SaveImageAndMetadata

Saves the image and the metadata to a JSON format file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

pathImage

The full path to the destination image file.

pathMetadata

The full path to the destination metadata file.

type

File format, as defined by EImageFileType.
If not specified, the file format is determined from the file extension.

EDepthMap8::SaveJpeg

Saves the EDepthMap8 object to the given image file, in JPEG format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveJpeg(
 const std::string& path,
 int quality
 )
```

Parameters

path

The full path of the destination file.

quality

JPEG quality, between 0 and 100 (100 is best quality). The default value is 75.

EDepthMap8::SaveJpeg2K

Saves the EDepthMap8 object to the given image file, in JPEG 2000 format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveJpeg2K(
 const std::string& path,
 int quality
 )
```

Parameters

path

The full path of the destination file.

quality

JPEG 2000 quality, between 1 and 512.
The default value is 16.

EDepthMap8::SaveMetadata

Saves the metadata to a file in JSON format

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EDepthMap8::SetBufferPtr

Sets the pointer to an externally allocated buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the image.

bitsPerRow

The total number of bits contained in a row, padding included. Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EDepthMap8::SetBufferPtr.

EDepthMap8::SetPixel

Sets the value of a pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPixel(
 EDepth8 value,
 int x,
 int y
 )
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EDepthMap8::SetSize

Sets the width and height of a DepthMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetSize(
 int width,
 int height
 )
void SetSize(
 const EDepthMap& other
 )
```

Parameters

width

The new requested DepthMap width.

height

The new requested DepthMap height.

other

The other DepthMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new buffer (deallocate the old buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new buffer and setting its size creates a 4-byte aligned buffer, by default.

EDepthMap8::GetType

Returns the image type.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::EImageType GetType() const
```

EDepthMap8::GetUndefinedValue

Returns the Undefined value. That value is used to mark pixels with no valid depth value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepth8 GetUndefinedValue() const
```

EDepthMap8::GetWidth

EDepthMap8::SetWidth

Access depth map Width.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetWidth() const
void SetWidth(int width)
```

EDepthMap8::GetZResolution

EDepthMap8::SetZResolution

Access the Z Resolution (depth units per grey scale value).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetZResolution() const
void SetZResolution(float resolution)
```

4.84. EDepthMapToMeshConverter Class

Performs the conversion from a EDepthMap to a EMesh, using the given calibration model.
A Depth Map is a grayscale image acquired by a laser triangulation system.
The calibration model defines how to transform a pixel from the Depth Map to a world space position.
The resulting 3D representation contains a EMesh representing the surface in the world space.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	Using the ECalibrationModel, the method 'Convert' performs the conversion from a EDepthMap to a EMesh.
EDepthMapToMeshConverter	Creates an EDepthMapToMeshConverter object.
GetCalibrationModel	Access the ECalibrationModel used for conversion.
Load	Loads the converter configuration. The given ESerializer must have been created for reading.

operator=	Assignment operator.
Save	Saves the converter configuration. The given ESerializer must have been created for writing.
SetCalibrationModel	Access the ECalibrationModel used for conversion.

EDepthMapToMeshConverter::GetCalibrationModel

EDepthMapToMeshConverter::SetCalibrationModel

Access the ECalibrationModel used for conversion.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const ECalibrationModel& GetCalibrationModel() const

void SetCalibrationModel(const ECalibrationModel& model)
```

EDepthMapToMeshConverter::Convert

Using the ECalibrationModel, the method 'Convert' performs the conversion from a EDepthMap to a EMesh.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Convert(
 const EDepthMap8& srcDepthMap,
 EMesh& obj
 )

void Convert(
 const EDepthMap16& srcDepthMap,
 EMesh& obj
 )

void Convert(
 const EDepthMap8& srcDepthMap,
 ERegion& region,
 EMesh& obj
 )

void Convert(
 const EDepthMap16& srcDepthMap,
 ERegion& region,
 EMesh& obj
 )
```

Parameters

srcDepthMap
The Depth Map to convert.

obj
The destination mesh.

region
The region of interest.

EDepthMapToMeshConverter::EDepthMapToMeshConverter

Creates an EDepthMapToMeshConverter object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EDepthMapToMeshConverter(
 )
void EDepthMapToMeshConverter(
 const EDepthMapToMeshConverter& other
 )
```

Parameters

other
Another EDepthMapToMeshConverter.

EDepthMapToMeshConverter::Load

Loads the converter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EDepthMapToMeshConverter::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepthMapToMeshConverter& operator=(
 const EDepthMapToMeshConverter& other
 )
```

Parameters

other

-

EDepthMapToMeshConverter::Save

Saves the converter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.85. EDepthMapToPointCloudConverter Class

Performs the conversion from a EDepthMap to a EPointCloud, using the given calibration model.
A Depth Map is a grayscale image acquired by a laser triangulation system.
The calibration model defines how to transform a pixel from the Depth Map to a world space position.
The resulting EPointCloud contains a point per defined pixel of the Depth Map.
Undefined pixels are discarded.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	Applies the ECalibrationModel to the Depth Map pixels and fill the EPointCloud with world positions.
EDepthMapToPointCloudConverter	Create a EDepthMapToPointCloudConverter.
GetCalibrationModel	Access the ECalibrationModel used for conversion.
Load	Loads the converter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the converter configuration. The given ESerializer must have been created for writing.
SetCalibrationModel	Access the ECalibrationModel used for conversion.

EDepthMapToPointCloudConverter::GetCalibrationModel

EDepthMapToPointCloudConverter::SetCalibrationModel

Access the ECalibrationModel used for conversion.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const ECalibrationModel& GetCalibrationModel() const

void SetCalibrationModel(const ECalibrationModel& model)
```

EDepthMapToPointCloudConverter::Convert

Applies the ECalibrationModel to the Depth Map pixels and fill the EPointCloud with world positions.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Convert(
 const EDepthMap8& srcDepthMap,
 EPointCloud& pc
 )

void Convert(
 const EDepthMap16& srcDepthMap,
 EPointCloud& pc
 )
```

```
void Convert(
 const EDepthMap32f& srcDepthMap,
 EPointCloud& pc
 )

void Convert(
 const EDepthMap8& srcDepthMap,
 ERegion& region,
 EPointCloud& pc
 )

void Convert(
 const EDepthMap16& srcDepthMap,
 ERegion& region,
 EPointCloud& pc
 )

void Convert(
 const EDepthMap32f& srcDepthMap,
 ERegion& region,
 EPointCloud& pc
 )
```

Parameters

srcDepthMap

The Depth Map to convert.

pc

The destination Point Cloud.

region

The region of interest, only pixels inside the given region are converted and added to the Point Cloud.

EDepthMapToPointCloudConverter::EDepthMapToPointCloudConverter

Create a EDepthMapToPointCloudConverter.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EDepthMapToPointCloudConverter(
 )

void EDepthMapToPointCloudConverter(
 const EDepthMapToPointCloudConverter& other
 )
```

Parameters

other

Another EDepthMapToPointCloudConverter.

EDepthMapToPointCloudConverter::Load

Loads the converter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EDepthMapToPointCloudConverter::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepthMapToPointCloudConverter& operator=(
 const EDepthMapToPointCloudConverter& other
 )
```

Parameters

other
Another EDepthMapToPointCloudConverter.

EDepthMapToPointCloudConverter::Save

Saves the converter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
```

```
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

4.86. EDrawableExtent Class

Drawable surface extent.

Namespace: Euresys::Open_eVision

Methods

DoesContainHorizontalLine	-
DoesContainPoint	-
DoesContainRectangle	-
EDrawableExtent	-
GetBottomExclusive	-
GetHeight	-
GetLeft	-
GetRightExclusive	-
GetTop	-
GetWidth	-
IsInfinite	-
operator=	-

EDrawableExtent::GetBottomExclusive

-

Namespace: Euresys::Open_eVision

[C++]

```
int GetBottomExclusive() const
```

EDrawableExtent::DoesContainHorizontalLine

-

Namespace: Euresys::Open_eVision

```
[C++]
bool DoesContainHorizontalLine(
 int y,
 int x1,
 int x2
 )
```

Parameters

y

-

x1

-

x2

-

EDrawableExtent::DoesContainPoint

-

Namespace: Euresys::Open_eVision

```
[C++]
bool DoesContainPoint(
 int x,
 int y
 )
```

Parameters

x

-

y

-

EDrawableExtent::DoesContainRectangle

-

Namespace: Euresys::Open_eVision

[C++]

```
bool DoesContainRectangle(
 int rectangleLeft,
 int rectangleTop,
 OEV_UINT32 rectangleWidth,
 OEV_UINT32 rectangleHeight
 )
```

Parameters

rectangleLeft

-

rectangleTop

-

rectangleWidth

-

rectangleHeight

-

EDrawableExtent::EDrawableExtent

-

Namespace: Euresys::Open_eVision

[C++]

```
void EDrawableExtent(
 )

void EDrawableExtent(
 int left,
 int top,
 OEV_UINT32 width,
 OEV_UINT32 height
 )

void EDrawableExtent(
 const EDrawableExtent& other
 )
```

Parameters

left

-

top

-

width

-

height

-

other

-

EDrawableExtent::GetHeight

-

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetHeight() const
```

EDrawableExtent::IsInfinite

-

Namespace: Euresys::Open_eVision

[C++]

```
int IsInfinite(
 )
```

EDrawableExtent::GetLeft

-

Namespace: Euresys::Open_eVision

[C++]

```
int GetLeft() const
```

EDrawableExtent::operator=

-

Namespace: Euresys::Open_eVision

```
[C++]
EDrawableExtent& operator=(
 const EDrawableExtent& other
 )
```

Parameters

other

-

EDrawableExtent::GetRightExclusive

-

Namespace: Euresys::Open_eVision

```
[C++]
int GetRightExclusive() const
```

EDrawableExtent::GetTop

-

Namespace: Euresys::Open_eVision

```
[C++]
int GetTop() const
```

EDrawableExtent::GetWidth

-

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetWidth() const
```

4.87. EDrawAdapter Class

Draw adapter interface used for drawing Open eVision objects and results.
The interface contains various primitives to draw images and various shapes (line, rectangle, ellipse, polygon). It draws the contours of shapes using a EPen and fills the inside of a shape with a EBrush. It has a default pen (EDrawAdapter::Pen) and default brush (EDrawAdapter::Brush) but they can be overridden using the optional pen and brush arguments of shape primitives. If no default pen and brush are set and no optional pen and brush are specified when calling a shape primitives, the shape will not be drawn. For the "Fill" primitives, no pen will results in no contours being drawn around the shape and no brush is equivalent to the corresponding "Draw" primitive.

Derived Class(es):EExternalDrawAdapterEGenericDrawAdapterEWindowsDrawAdapter

Namespace: Euresys::Open_eVision

Methods

Arc	Draws an arc defined by a rectangle, angle, and amplitude.
BackedText	Draws a text with a background.
DrawMask	Draws a mask with a brush.
DrawPoint	Draws a point at the given coordinate.
Ellipse	Draws an ellipse.
FilledEllipse	Fills an ellipse.
FilledPolygon	Fills a polygon.
FilledRectangle	Fills a rectangle.
FilledRotatedEllipse	Fills a rotated ellipse.
GetBrush	Default brush.
GetFont	Default font.
GetPen	Default pen.
GetTextSize	Size of the given text using the default font (EDrawAdapter::Font).
Image	Draws an image.
Line	Draws a line between two points.
Polygon	Draws a polygon.
Rectangle	Draws a rectangle.
RotatedEllipse	Draws a rotated ellipse.
SetBrush	Default brush.
SetFont	Default font.
SetPen	Default pen.
Text	Draws a text.

UseCurrentBrush	Use the current pen set in the drawing framework.
UseCurrentPen	Use the current pen set in the drawing framework.

EDrawAdapter::Arc

Draws an arc defined by a rectangle, angle, and amplitude.

Namespace: Euresys::Open_eVision

[C++]

```
void Arc(
 int orgX,
 int orgY,
 int width,
 int height,
 float startAngle,
 float amplitude,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

startAngle
Starting angle of the arc in radians

amplitude
Amplitude of the arc in radians

pen
Optional pen to use

EDrawAdapter::BackedText

Draws a text with a background.

Namespace: Euresys::Open_eVision

[C++]

```
void BackedText(
 const std::string& text,
 int x,
 int y,
 EBrush textBrush,
 EBrush backgroundBrush
 )
void BackedText(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush,
 EBrush backgroundBrush
 )
```

Parameters

text
Text to draw.

x
X position of the text.

y
Y position of the text.

textBrush
Optional brush to use for the color of the text.

backgroundBrush
Optional brush to use for the background.

orientation
Orientation of the text.

EDrawAdapter::GetBrush

EDrawAdapter::SetBrush

Default brush.

Namespace: Euresys::Open_eVision

[C++]

```
EBrush GetBrush() const
void SetBrush(const EBrush& brush)
```

EDrawAdapter::DrawMask

Draws a mask with a brush.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawMask(
 const EBaseROI& mask,
 const EBrush& brush
 )
void DrawMask(
 const EBaseROI& mask,
 float orgX,
 float orgY,
 float width,
 float height,
 const EBrush& brush
 )
```

Parameters

mask

Mask to draw

brush

Brush to draw the mask with.

orgX

X coordinate of the point where to draw the image.

orgY

Y coordinate of the point where to draw the image.

width

Width of the destination rectangle in which to draw the image.

height

Height of the destination rectangle in which to draw the image.

Remarks

The type of the mask iamge must be BW8, BW16, or BW32.
The default implentation converts the mask into a EImageC24A image using the brush and draws this image.

EDrawAdapter::DrawPoint

Draws a point at the given coordinate.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawPoint(
 int x1,
 int y1,
 EPen pen
 )
```

Parameters

x1
X coordinate

y1
Y coordinate

pen
Optional pen to use

EDrawAdapter::Ellipse

Draws an ellipse.

Namespace: Euresys::Open_eVision

```
[C++]
void Ellipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle containing the ellipse

orgY
Y origin of the rectangle containing the ellipse

width
Width of the rectangle containing the ellipse

height
Height of the rectangle containing the ellipse

pen
Optional pen to use

EDrawAdapter::FilledEllipse

Fills an ellipse.

Namespace: Euresys::Open_eVision

```
[C++]
void FilledEllipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle containing the ellipse

orgY
Y origin of the rectangle containing the ellipse

width
Width of the rectangle containing the ellipse

height
Height of the rectangle containing the ellipse

pen
Optional pen to use for drawing the contour of the ellipse

brush
Optional pen to use for drawing the inside of the ellipse

EDrawAdapter::FilledPolygon

Fills a polygon.

Namespace: Euresys::Open_eVision

```
[C++]
void FilledPolygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen,
 EBrush brush
 )
void FilledPolygon(
 const EPolygon& polygon,
 EPen pen,
 EBrush brush
 )
```

Parameters

points
Points of the polygon
pen
Optional pen to use for drawing the contour of the polygon
brush
Optional pen to use for drawing the inside of the polygon
polygon
Polygon

EDrawAdapter::FilledRectangle

Fills a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledRectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle
orgY
Y origin of the rectangle
width
Width of the rectangle
height
Height of the rectangle
pen
Optional pen to use for drawing the contour of the rectangle
brush
Optional brush to use for filling the inside of the rectangle

EDrawAdapter::FilledRotatedEllipse

Fills a rotated ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledRotatedEllipse(
 float centerX,
 float centerY,
 float radianAngle,
 float longAxis,
 float shortAxis,
 EPen pen,
 EBrush brush
 )
```

Parameters

centerX
X coordinate of the ellipse center

centerY
Y coordinate of the ellipse center

radianAngle
Angle of the rotated ellipse in radian

longAxis
Long axis length

shortAxis
Short axis length

pen
Pen to draw the ellipse with.

brush
-

Remarks

A default implementation based on EDrawAdapter already exists.

EDrawAdapter::GetFont

EDrawAdapter::SetFont

Default font.

Namespace: Euresys::Open_eVision

[C++]

```
EFont GetFont() const
void SetFont(const EFont& font)
```

EDrawAdapter::GetTextSize

Size of the given text using the default font (EDrawAdapter::Font).

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetTextSize(
 const std::string& text
 )
```

Parameters

text

Text

EDrawAdapter::Image

Draws an image.

Namespace: Euresys::Open_eVision

```
[C++]
void Image(
 const EBaseROI& image
 )

void Image(
 const EROIBW8& image,
 EBW8Vector* pColorScale
 )

void Image(
 const EROIBW8& image,
 EC24Vector* pColorScale
 )

void Image(
 const EBaseROI& image,
 float orgX,
 float orgY,
 float width,
 float height
 )
```

```
void Image(
 const EROIBW8& image,
 EC24Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
void Image(
 const EROIBW8& image,
 EBW8Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
```

Parameters

image
Image.

pColorScale
Color scale to draw a grayscale image with.

orgX
X coordinate of the point where to draw the image.

orgY
Y coordinate of the point where to draw the image.

width
Width of the destination rectangle in which to draw the image.

height
Height of the destination rectangle in which to draw the image.

EDrawAdapter::Line

Draws a line between two points.

Namespace: Euresys::Open_eVision

[C++]

```
void Line(
 int x1,
 int y1,
 int x2,
 int y2,
 EPen pen
 )
```

Parameters

x1
X coordinate of line origin point

y1
Y coordinate of line origin point

x2
X coordinate of line end point

y2
Y coordinate of line end point

pen
Optional pen to use

EDrawAdapter::GetPen

EDrawAdapter::SetPen

Default pen.

Namespace: Euresys::Open_eVision

```
[C++]
EPen GetPen() const
void SetPen(const EPen& pen)
```

EDrawAdapter::Polygon

Draws a polygon.

Namespace: Euresys::Open_eVision

```
[C++]
void Polygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen
 )
void Polygon(
 const EPolygon& polygon,
 EPen arg1
 )
```

Parameters

points
Points of the polygon
pen
Optional pen to use
polygon
Polygon
arg1
-

EDrawAdapter::Rectangle

Draws a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void Rectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle
orgY
Y origin of the rectangle
width
Width of the rectangle
height
Height of the rectangle
pen
Optional pen to use

EDrawAdapter::RotatedEllipse

Draws a rotated ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void RotatedEllipse(
 float centerX,
 float centerY,
 float radianAngle,
 float longAxis,
 float shortAxis,
 bool drawDiagonals,
 EPen pen
 )
```

Parameters

centerX
X coordinate of the ellipse center

centerY
Y coordinate of the ellipse center

radianAngle
Angle of the rotated ellipse in radian

longAxis
Long axis length

shortAxis
Short axis length

drawDiagonals
Whether to draw the diagonal

pen
Pen to draw the ellipse with.

Remarks

A default implementation based on EDrawAdapter already exists.

EDrawAdapter::Text

Draws a text.

Namespace: Euresys::Open_eVision

[C++]

```
void Text(
 const std::string& text,
 int x,
 int y,
 EBrush textBrush
 )
```

```
void Text(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush
 )
```

Parameters

text

Text to draw.

x

X position of the text.

y

Y position of the text.

textBrush

Optional brush to use for the color of the text.

orientation

Orientation of the text in radians.

EDrawAdapter::UseCurrentBrush

Use the current pen set in the drawing framework.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentBrush(
 )
```

EDrawAdapter::UseCurrentPen

Use the current pen set in the drawing framework.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentPen(
 )
```

4.88. EEllipseRegion Class

Manages a complete context for an ERegion shaped like an ellipse.

Base Class:ERegion
Namespace: Euresys::Open_eVision

Methods

Drag	Moves the specified handle to a new position and updates all placement parameters of the region.
EEllipseRegion	Constructs an EEllipseRegion context.
GetAngle	Angle of the region
GetCenter	Center of the region
GetHorizontalRadius	Horizontal radius of the region
GetVerticalRadius	Vertical radius of the region
HitTest	Detects if the cursor is placed over one of the dragging handles.
Load	Loads the EEllipseRegion. The given ESerializer must have been created for reading.
operator!=	Checks if this EEllipseRegion instance is not strictly equal to another
operator=	Assignment operator.
operator==	Checks if this EEllipseRegion instance is strictly equal to another
Rotate	Creates a new EEllipseRegion by rotating the EEllipseRegion.
Save	Saves the EEllipseRegion. The given ESerializer must have been created for writing.
Scale	Creates a new EEllipseRegion by scaling the EEllipseRegion.
SetAngle	Angle of the region
SetCenter	Center of the region
SetHorizontalRadius	Horizontal radius of the region
SetVerticalRadius	Vertical radius of the region
Translate	Creates a new EEllipseRegion by translating the EEllipseRegion.

EEllipseRegion::GetAngle

EEllipseRegion::SetAngle

Angle of the region

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
void SetAngle(float angle)
```

EEllipseRegion::GetCenter

EEllipseRegion::SetCenter

Center of the region

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const

void SetCenter(const EPoint& center)
```

EEllipseRegion::Drag

Moves the specified handle to a new position and updates all placement parameters of the region.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
x-coordinate of the mouse cursor.

y
y-coordinate of the mouse cursor.

zoomX
Horizontal zoom factor. By default, true scale is used.

zoomY
Vertical zoom factor. By default, true scale is used.

panX
Horizontal pan offset. By default, no pan is added.

panY
Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with EEllipseRegion::HitTest and EEllipseRegion::Drag.

EEllipseRegion::EEllipseRegion

Constructs an EEllipseRegion context.

Namespace: Euresys::Open_eVision

[C++]

```
void EEllipseRegion(
 )

void EEllipseRegion(
 float centerX,
 float centerY,
 float radius1,
 float radius2,
 float angle
 )

void EEllipseRegion(
 const EPoint& center,
 float radius1,
 float radius2,
 float angle
 )

void EEllipseRegion(
 const EPoint& center,
 const EPoint& axisEnd1,
 const EPoint& axisEnd2
 )

void EEllipseRegion(
 const EPoint& pt1,
 const EPoint& pt2,
 const EPoint& pt3,
 const EPoint& pt4,
 const EPoint& pt5
 )

void EEllipseRegion(
 const EEllipseRegion& other
 )
```

Parameters

centerX
The abscissa of the center of the EEllipseRegion.
centerY
The ordinate of the center of the EEllipseRegion.
radius1
The abscissa radius of the non rotated EEllipseRegion.
radius2
The ordinate radius of the non rotated EEllipseRegion.
angle
The angle of the rotated EEllipseRegion.
center
The center of the EEllipseRegion.
axisEnd1
The point corresponding to the end of the abscissa axis of the non rotated EEllipseRegion.
axisEnd2
The point corresponding to the end of the ordinate axis of the non rotated EEllipseRegion.
pt1
One of the five points defining the EEllipseRegion.
pt2
One of the five points defining the EEllipseRegion.
pt3
One of the five points defining the EEllipseRegion.
pt4
One of the five points defining the EEllipseRegion.
pt5
One of the five points defining the EEllipseRegion.
other
EEllipseRegion context to copy.

Remarks

When defining an EEllipseRegion, the two resulting radius values must not be 0 else an EError_Parameter3OutOfRange or EError is thrown.
When defining an EEllipseRegion, the two resulting radius valued must be small enough so that the region fit in memory.
When defining an EEllipseRegion with two axis ends, the two axis ends and the center must not all lie on the same straight line else an EError is thrown.
When defining an EEllipseRegion with five points, no more than two of the points should lie on the same straight line else an EError is thrown.

EEllipseRegion::HitTest

Detects if the cursor is placed over one of the dragging handles.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EEditionMode HitTest(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal pan offset. By default, no pan is added.

panY

Vertical pan offset. By default, no pan is added.

Remarks

Returns a handle identifier, as defined by EEditionMode.
If zooming and/or panning were used when drawing the region, the same values must be used with EEllipseRegion::HitTest and EEllipseRegion::Drag.

EEllipseRegion::GetHorizontalRadius

EEllipseRegion::SetHorizontalRadius

Horizontal radius of the region

Namespace: Euresys::Open_eVision

[C++]

```
float GetHorizontalRadius() const

void SetHorizontalRadius(float radius)
```

EEllipseRegion::Load

Loads the EEllipseRegion. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EEllipseRegion::operator!=

Checks if this EEllipseRegion instance is not strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const EEllipseRegion& other
 )
```

Parameters

other
Reference to the other EEllipseRegion instance

EEllipseRegion::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EEllipseRegion& operator=(
 const EEllipseRegion& other
 )
```

Parameters

other
Reference to the EEllipseRegion used for the assignment

EEllipseRegion::operator==

Checks if this EEllipseRegion instance is strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EEllipseRegion& other
 )
```

Parameters

other

Reference to the other EEllipseRegion instance

EEllipseRegion::Rotate

Creates a new EEllipseRegion by rotating the EEllipseRegion.

Namespace: Euresys::Open_eVision

```
[C++]
EEllipseRegion Rotate(
 float angle
 )
```

Parameters

angle

Rotation angle

EEllipseRegion::Save

Saves the EEllipseRegion. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

EEllipseRegion::Scale

Creates a new EEllipseRegion by scaling the EEllipseRegion.

Namespace: Euresys::Open_eVision

```
[C++]
EEllipseRegion Scale(
 float scale
 )
EEllipseRegion Scale(
 float scaleX,
 float scaleY
 )
```

Parameters

scale
Isotropic scale

scaleX
Horizontal scale

scaleY
Vertical scale

EEllipseRegion::Translate

Creates a new EEllipseRegion by translating the EEllipseRegion.

Namespace: Euresys::Open_eVision

```
[C++]
EEllipseRegion Translate(
 float dx,
 float dy
 )
```

Parameters

dx
Horizontal translation in pixel value

dy
Vertical translation in pixel value

EEllipseRegion::GetVerticalRadius

EEllipseRegion::SetVerticalRadius

Vertical radius of the region

Namespace: Euresys::Open_eVision

```
[C++]
float GetVerticalRadius() const
void SetVerticalRadius(float radius)
```

4.89. EErrorStatistics Class

This object contains the error statistics (deviation between model and aligned points).

Namespace: Euresys::Open_eVision::Easy3D

Methods

EErrorStatistics	Creates a EErrorStatistics object.
GetMax	Gets/Sets the maximum error (calculated on the valid samples).
GetMean	Gets/Sets the mean error (calculated on the valid samples).
GetMin	Gets/Sets the minimum error (calculated on the valid samples).
GetNumOfErrors	Gets/Sets the number of errors (samples not found) which is the number of samples on which no error could be calculated.
GetNumOfValidSamples	Gets/Sets the number of valid samples which is the number of samples on which the error is calculated.
GetStdDev	Gets/Sets the standard deviation error (calculated on the valid samples).
Load	Loads a EErrorStatistics. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves a EErrorStatistics. The given ESerializer must have been created for writing.
SetMax	Gets/Sets the maximum error (calculated on the valid samples).
SetMean	Gets/Sets the mean error (calculated on the valid samples).
SetMin	Gets/Sets the minimum error (calculated on the valid samples).
SetNumOfErrors	Gets/Sets the number of errors (samples not found) which is the number of samples on which no error could be calculated.

SetNumOfValidSamples	Gets/Sets the number of valid samples which is the number of samples on which the error is calculated.
SetStdDev	Gets/Sets the standard deviation error (calculated on the valid samples).

EErrorStatistics::EErrorStatistics

Creates a EErrorStatistics object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EErrorStatistics(
 )

void EErrorStatistics(
 const EErrorStatistics& other
 )
```

Parameters

other

Reference used for the initialization.

EErrorStatistics::Load

Loads a EErrorStatistics. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EErrorStatistics::GetMax

EErrorStatistics::SetMax

Gets/Sets the maximum error (calculated on the valid samples).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMax() const
void SetMax(float max)
```

EErrorStatistics::GetMean

EErrorStatistics::SetMean

Gets/Sets the mean error (calculated on the valid samples).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMean() const
void SetMean(float mean)
```

EErrorStatistics::GetMin

EErrorStatistics::SetMin

Gets/Sets the minimum error (calculated on the valid samples).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMin() const
void SetMin(float min)
```

EErrorStatistics::GetNumOfErrors

EErrorStatistics::SetNumOfErrors

Gets/Sets the number of errors (samples not found) which is the number of samples on which no error could be calculated.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumOfErrors() const

void SetNumOfErrors(int numOfErrors)
```

EErrorStatistics::GetNumOfValidSamples

EErrorStatistics::SetNumOfValidSamples

Gets/Sets the number of valid samples which is the number of samples on which the error is calculated.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumOfValidSamples() const

void SetNumOfValidSamples(int numOfValidSamples)
```

EErrorStatistics::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EErrorStatistics& operator=(
 const EErrorStatistics& other
 )
```

Parameters

other

-

EErrorStatistics::Save

Saves a EErrorStatistics. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EErrorStatistics::GetStdDev

EErrorStatistics::SetStdDev

Gets/Sets the standard deviation error (calculated on the valid samples).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetStdDev() const
void SetStdDev(float stdDev)
```

4.90. EException Class

Holds the exception information, that is the code and the description of the error that has thrown the exception.

Remarks

Each time an Open eVision error occurs, an exception is thrown. Exceptions feature an error code and a description. To catch a potentially arising exception, the function call is included in a try-catch block.

Namespace: Euresys::Open_eVision

Methods

EException	Constructs a EException object.
GetError	Code of the error that has thrown the exception.
operator=	Assignment operator.
SetError	Code of the error that has thrown the exception.
What	Returns the description of the error that has thrown the exception.

EException::EException

Constructs a EException object.

Namespace: Euresys::Open_eVision

```
[C++]
void EException(
 Euresys::Open_eVision::EError error,
 const std::string& message
 )
void EException(
 const EException& other
 )
void EException(
 const std::string& message
 )
```

Parameters

error
The EError to construct the exception from.

message
A string to construct the exception from.

other
-

EException::GetError

EException::SetError

Code of the error that has thrown the exception.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EError GetError() const
```

```
void SetError(Euresys::Open_eVision::EError error)
```

EException::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EException& operator=(
 const EException& other
 )
```

Parameters

other

-

EException::What

Returns the description of the error that has thrown the exception.

Namespace: Euresys::Open_eVision

```
[C++]
std::string What(
 )
```

4.91. EExplicitGeometricCalibrationModel Class

EExplicitGeometricCalibrationModel is used to calibrate a depth map from a minimal set of explicit geometric values.
All model parameters are given to the EExplicitGeometricCalibrationModel constructor.

Base Class:ECalibrationModel

Namespace: Euresys::Open_eVision::Easy3D

Methods

EExplicitGeometricCalibrationModel	Constructs a EExplicitGeometricCalibrationModel. EExplicitGeometricCalibrationModel is used to calibrate a depth map from a minimal set of explicit geometric values.
GetCameraAngle	Camera view angle.

GetCameraHeight	The height from the camera optical center to the object reference plane (assuming the reference plane is projected in the center line of the image), in mm.
GetFocalLength	Camera focal length.
GetLaserPlaneAngle	Laser plane angle.
GetMotionIncrement	Motion increment (Distance between each line of the depth map).
GetRoiBottomLine	The ROI bottom line used in laser line extraction.
GetRoiLeftColumn	The ROI left column used in laser line extraction.
GetSensorHeight	Camera sensor height.
GetSensorWidth	Camera sensor width.
GetSensorXResolution	Camera X resolution.
GetSensorYResolution	Camera Y resolution.
GetType	Returns the type of calibration model, see ECalibrationType.
IsInitialized	Returns true if the model has been correctly initialized.
Load	Loads the EExplicitGeometricCalibrationModel. The given ESerializer must have been created for reading.
operator=	Assignment operator.
operator==	Comparison operator.
Save	Saves the EExplicitGeometricCalibrationModel. The given ESerializer must have been created for writing.

EExplicitGeometricCalibrationModel::GetCameraAngle

Camera view angle.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCameraAngle() const
```

EExplicitGeometricCalibrationModel::GetCameraHeight

The height from the camera optical center to the object reference plane (assuming the reference plane is projected in the center line of the image), in mm.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCameraHeight() const
```

EExplicitGeometricCalibrationModel::EExplicitGeometricCalibrationModel

Constructs a EExplicitGeometricCalibrationModel.
EExplicitGeometricCalibrationModel is used to calibrate a depth map from a minimal set of explicit geometric values.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EExplicitGeometricCalibrationModel(
 )

void EExplicitGeometricCalibrationModel(
 float sensorWidth,
 float sensorHeight,
 int sensorXResolution,
 int sensorYResolution,
 int roiLeftColumn,
 int roiBottomLine,
 float focalLength,
 float cameraAngle,
 float cameraHeight,
 float laserPlaneAngle,
 float motionIncrement
 )

void EExplicitGeometricCalibrationModel(
 const EExplicitGeometricCalibrationModel& other
 )
```

Parameters

sensorWidth
The camera sensor width, in mm.

sensorHeight
The camera sensor height, in mm.

sensorXResolution
The camera X resolution (pixel count in width).

sensorYResolution
The camera Y resolution (pixel count in height).

roiLeftColumn
The ROI left column used in laser line extraction.
Between the left (0) and the right (width) of the image.

roiBottomLine
The ROI bottom line used in laser line extraction.
Between the top (0) and the bottom (height) of the image. That's the depth map values origin.

focalLength

The camera optics focal length, in mm.

cameraAngle

The camera angle from the vertical axis.
Looking down camera is angle 0 and positive in counter clockwise direction. Valid values are between 0 (vertical orientation) and lower that 90 degrees (horizontal orientation).

cameraHeight

The height from the camera optical center to the object reference plane (assuming the reference plane is projected in the center line of the image), in mm.

laserPlaneAngle

The laser plane angle from the vertical axis.
A perfect vertical laser orientation is angle 0 and negative in clockwise direction. Valid values for laser angle are between -90 degrees (excluded) and +90 degrees (excluded).

motionIncrement

The distance in mm between each line of the depth map.
That's the relative motion of the camera/laser setup to the object position.

other

Another EExplicitGeometricCalibrationModel.

EExplicitGeometricCalibrationModel::GetFocalLength

Camera focal length.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetFocalLength() const
```

EExplicitGeometricCalibrationModel::IsInitialized

Returns true if the model has been correctly initialized.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsInitialized(
 )
```

EExplicitGeometricCalibrationModel::GetLaserPlaneAngle

Laser plane angle.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetLaserPlaneAngle() const
```

EExplicitGeometricCalibrationModel::Load

Loads the EExplicitGeometricCalibrationModel. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EExplicitGeometricCalibrationModel::GetMotionIncrement

Motion increment (Distance between each line of the depth map).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMotionIncrement() const
```

EExplicitGeometricCalibrationModel::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EExplicitGeometricCalibrationModel& operator=(
 const EExplicitGeometricCalibrationModel& other
 )
```

Parameters

other

Another EExplicitGeometricCalibrationModel.

EExplicitGeometricCalibrationModel::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const EExplicitGeometricCalibrationModel& other
 )
```

Parameters

other

The other EExplicitGeometricCalibrationModel.

EExplicitGeometricCalibrationModel::GetRoiBottomLine

The ROI bottom line used in laser line extraction.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRoiBottomLine() const
```

EExplicitGeometricCalibrationModel::GetRoiLeftColumn

The ROI left column used in laser line extraction.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRoiLeftColumn() const
```

EExplicitGeometricCalibrationModel::Save

Saves the EExplicitGeometricCalibrationModel. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EExplicitGeometricCalibrationModel::GetSensorHeight

Camera sensor height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetSensorHeight() const
```

EExplicitGeometricCalibrationModel::GetSensorWidth

Camera sensor width.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetSensorWidth() const
```

EExplicitGeometricCalibrationModel::GetSensorXResolution

Camera X resolution.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetSensorXResolution() const
```

EExplicitGeometricCalibrationModel::GetSensorYResolution

Camera Y resolution.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetSensorYResolution() const
```

EExplicitGeometricCalibrationModel::GetType

Returns the type of calibration model, see ECalibrationType.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::Easy3D::ECalibrationType GetType() const
```

4.92. EExternalDrawAdapter Class

-

Base Class:EDrawAdapter
Namespace: Euresys::Open_eVision

Methods

Arc	Draws an arc defined by a rectangle, angle, and amplitude.
BackedText	Draws a text with a background.
DrawPoint	Draws a point at the given coordinate.
EExternalDrawAdapter	-
Ellipse	Draws an ellipse.
FilledEllipse	Fills an ellipse.
FilledPolygon	Fills a polygon.
FilledRectangle	Fills a rectangle.
GetExtent	Extent of the underlying surface being drawn on.
GetTextSize	Size of the given text using the default font (EExternalDrawAdapter::Font).
Image	Draws an image.
Line	Draws a line between two points.
operator=	-

Polygon	Draws a polygon.
Rectangle	Draws a rectangle.
SetArcFunctionPtr	-
SetBackedTextFunctionPtr	-
SetBrush	Default brush.
SetDrawPointFunctionPtr	-
SetEllipseFunctionPtr	-
SetFillEllipseFunctionPtr	-
SetFillPolygonFunctionPtr	-
SetFillRectangleFunctionPtr	-
SetFont	Default font.
SetGetExtentFunctionPtr	-
SetGetTextSizeFunctionPtr	-
SetImageFunctionPtr	-
SetImageWithColorScaleFunctionPtr	-
SetImageWithGrayscaleScaleFunctionPtr	-
SetLineFunctionPtr	-
SetPen	Default pen.
SetPolygonFunctionPtr	-
SetRectangleFunctionPtr	-
SetSetBrushFunctionPtr	-
SetSetFontFunctionPtr	-
SetSetPenFunctionPtr	-
SetTextFunctionPtr	-
SetUseCurrentBrushFunctionPtr	-
SetUseCurrentPenFunctionPtr	-
Text	Draws a text.

UseCurrentBrush	Use the current pen set in the drawing framework.
UseCurrentPen	Use the current pen set in the drawing framework.

EExternalDrawAdapter::Arc

Draws an arc defined by a rectangle, angle, and amplitude.

Namespace: Euresys::Open_eVision

[C++]

```
void Arc(
 int orgX,
 int orgY,
 int width,
 int height,
 float startAngle,
 float amplitude,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

startAngle
Starting angle of the arc in radians

amplitude
Amplitude of the arc in radians

pen
Optional pen to use

EExternalDrawAdapter::SetArcFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetArcFunctionPtr(void* ptr)
```

EExternalDrawAdapter::BackedText

Draws a text with a background.

Namespace: Euresys::Open_eVision

[C++]

```
void BackedText(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush,
 EBrush backgroundBrush
 )
```

Parameters

text
Text to draw.

x
X position of the text.

y
Y position of the text.

orientation
Orientation of the text.

textBrush
Optional brush to use for the color of the text.

backgroundBrush
Optional brush to use for the background.

EExternalDrawAdapter::SetBackedTextFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetBackedTextFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetBrush

Default brush.

Namespace: Euresys::Open_eVision

[C++]

```
void SetBrush(const EBrush& brush)
```

EExternalDrawAdapter::DrawPoint

Draws a point at the given coordinate.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPoint(
 int x1,
 int y1,
 EPen pen
 )
```

Parameters

x1

X coordinate

y1

Y coordinate

pen

Optional pen to use

EExternalDrawAdapter::SetDrawPointFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetDrawPointFunctionPtr(void* ptr)
```

EExternalDrawAdapter::EExternalDrawAdapter

-

Namespace: Euresys::Open_eVision

[C++]

```
void EExternalDrawAdapter(
 void* adapter
 )
```

```
void EExternalDrawAdapter(
 const EExternalDrawAdapter& other
 )
```

Parameters

adapter

-

other

-

EExternalDrawAdapter::Ellipse

Draws an ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void Ellipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX

X origin of the rectangle containing the ellipse

orgY

Y origin of the rectangle containing the ellipse

width

Width of the rectangle containing the ellipse

height

Height of the rectangle containing the ellipse

pen

Optional pen to use

EExternalDrawAdapter::SetEllipseFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetEllipseFunctionPtr(void* ptr)
```

EExternalDrawAdapter::GetExtent

Extent of the underlying surface being drawn on.

Namespace: Euresys::Open_eVision

[C++]

```
EDrawableExtent GetExtent()
```

EExternalDrawAdapter::FilledEllipse

Fills an ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledEllipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
: X origin of the rectangle containing the ellipse

orgY
: Y origin of the rectangle containing the ellipse

width
: Width of the rectangle containing the ellipse

height
: Height of the rectangle containing the ellipse

pen
: Optional pen to use for drawing the contour of the ellipse

brush
: Optional pen to use for drawing the inside of the ellipse

EExternalDrawAdapter::FilledPolygon

Fills a polygon.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledPolygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen,
 EBrush brush
 )
```

Parameters

points
Points of the polygon

pen
Optional pen to use for drawing the contour of the polygon

brush
Optional pen to use for drawing the inside of the polygon

EExternalDrawAdapter::FilledRectangle

Fills a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledRectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

pen
Optional pen to use for drawing the contour of the rectangle

brush
Optional brush to use for filling the inside of the rectangle

EExternalDrawAdapter::SetFillEllipseFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetFillEllipseFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetFillPolygonFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetFillPolygonFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetFillRectangleFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetFillRectangleFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetFont

Default font.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFont(const EFont& font)
```

EExternalDrawAdapter::SetGetExtentFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetGetExtentFunctionPtr(void* ptr)
```

EExternalDrawAdapter::GetTextSize

Size of the given text using the default font (EExternalDrawAdapter::Font).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetTextSize(
 const std::string& text
 )
```

Parameters

text

Text

EExternalDrawAdapter::SetGetTextSizeFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetGetTextSizeFunctionPtr(void* ptr)
```

EExternalDrawAdapter::Image

Draws an image.

Namespace: Euresys::Open_eVision

[C++]

```
void Image(
 const EBaseROI& image,
 float orgX,
 float orgY,
 float width,
 float height
 )
```

```
void Image(
 const EROIBW8& image,
 EC24Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
void Image(
 const EROIBW8& image,
 EBW8Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
```

Parameters

image
Image.

orgX
X coordinate of the point where to draw the image.

orgY
Y coordinate of the point where to draw the image.

width
Width of the destination rectangle in which to draw the image.

height
Height of the destination rectangle in which to draw the image.

pColorScale
Color scale to draw a grayscale image with.

EExternalDrawAdapter::SetImageFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetImageFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetImageWithColorScaleFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetImageWithColorScaleFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetImageWithGrayscaleScaleFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetImageWithGrayscaleScaleFunctionPtr(void* ptr)
```

EExternalDrawAdapter::Line

Draws a line between two points.

Namespace: Euresys::Open_eVision

[C++]

```
void Line(
 int x1,
 int y1,
 int x2,
 int y2,
 EPen pen
 )
```

Parameters

x1
X coordinate of line origin point

y1
Y coordinate of line origin point

x2
X coordinate of line end point

y2
Y coordinate of line end point

pen
Optional pen to use

EExternalDrawAdapter::SetLineFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetLineFunctionPtr(void* ptr)
```

EExternalDrawAdapter::operator=

-

Namespace: Euresys::Open_eVision

```
[C++]
EExternalDrawAdapter& operator=(
 const EExternalDrawAdapter& other
 )
```

Parameters

other

-

EExternalDrawAdapter::SetPen

Default pen.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPen(const EPen& pen)
```

EExternalDrawAdapter::Polygon

Draws a polygon.

Namespace: Euresys::Open_eVision

```
[C++]
void Polygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen
 )
```

Parameters

points
Points of the polygon

pen
Optional pen to use

EExternalDrawAdapter::SetPolygonFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetPolygonFunctionPtr(void* ptr)
```

EExternalDrawAdapter::Rectangle

Draws a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void Rectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

pen
Optional pen to use

EExternalDrawAdapter::SetRectangleFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetRectangleFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetSetBrushFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetSetBrushFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetSetFontFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetSetFontFunctionPtr(void* ptr)
```

EExternalDrawAdapter::SetSetPenFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetSetPenFunctionPtr(void* ptr)
```

EExternalDrawAdapter::Text

Draws a text.

Namespace: Euresys::Open_eVision

[C++]

```
void Text(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush
 )
```

Parameters

text
Text to draw.

x
X position of the text.

y
Y position of the text.

orientation
Orientation of the text in radians.

textBrush
Optional brush to use for the color of the text.

EExternalDrawAdapter::SetTextFunctionPtr

-

Namespace: Euresys::Open_eVision

[C++]

```
static void SetTextFunctionPtr(void* ptr)
```

EExternalDrawAdapter::UseCurrentBrush

Use the current pen set in the drawing framework.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentBrush(
 )
```

EExternalDrawAdapter::SetUseCurrentBrushFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetUseCurrentBrushFunctionPtr(void* ptr)
```

EExternalDrawAdapter::UseCurrentPen

Use the current pen set in the drawing framework.

Namespace: Euresys::Open_eVision

```
[C++]
void UseCurrentPen(
 )
```

EExternalDrawAdapter::SetUseCurrentPenFunctionPtr

-

Namespace: Euresys::Open_eVision

```
[C++]
static void SetUseCurrentPenFunctionPtr(void* ptr)
```

4.93. EFeaturesAligner Class

Alignment class, used to calculate the best transformation between matching pairs of points.
The object EFeaturesAligner contains a list of points called the 'Model points list'.
The method EFeaturesAligner::Compute takes a second list of points as argument which is called the 'Measured points list' and produces a E3DTransformMatrix as result.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Compute	Computes the best transformation matrix for a given Measured points list. This will compute the best transformation for the alignment between the Measured points and the Model points.

EFeaturesAligner	Creates a EFeaturesAligner object.
GetModelPoints	Sets/Gets the model points list. The model point list is a list of points that is used either as source or as destination of the transformation to calculate. The model points list should at least contain 3 unaligned points.
GetPolarityTransform	Sets/Gets the polarity of the transformation from EAlignmentPolarity.
Load	Loads the EFeaturesAligner object configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the EFeaturesAligner object configuration. The given ESerializer must have been created for writing.
SetModelPoints	Sets/Gets the model points list. The model point list is a list of points that is used either as source or as destination of the transformation to calculate. The model points list should at least contain 3 unaligned points.
SetPolarityTransform	Sets/Gets the polarity of the transformation from EAlignmentPolarity.

EFeaturesAligner::Compute

Computes the best transformation matrix for a given Measured points list.
This will compute the best transformation for the alignment between the Measured points and the Model points.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix Compute(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& measuredPoints
 )

E3DTransformMatrix Compute(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& measuredPoints,
 EErrorStatistics& errorStatistics
 )
```

Parameters

measuredPoints

The measured points list; the measured points list should at least contain 3 unaligned points.

errorStatistics

Optional parameter passed by reference. This object will contains the error statistics (deviation between model and aligned points).

EFeaturesAligner::EFeaturesAligner

Creates a EFeaturesAligner object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EFeaturesAligner(
 )
void EFeaturesAligner(
 const EFeaturesAligner& other
 )
```

Parameters

other

Reference to the object used for the initialization.

EFeaturesAligner::Load

Loads the EFeaturesAligner object configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EFeaturesAligner::GetModelPoints

EFeaturesAligner::SetModelPoints

Sets/Gets the model points list.
The model point list is a list of points that is used either as source or as destination of the transformation to calculate.
The model points list should at least contain 3 unaligned points.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DPoint> GetModelPoints() const
void SetModelPoints(const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& points)
```

EFeaturesAligner::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EFeaturesAligner& operator=(
 const EFeaturesAligner& other
 )
```

Parameters

other

-

EFeaturesAligner::GetPolarityTransform

EFeaturesAligner::SetPolarityTransform

Sets/Gets the polarity of the transformation from EAlignmentPolarity.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EAlignmentPolarity GetPolarityTransform() const
void SetPolarityTransform(Euresys::Open_eVision::Easy3D::EAlignmentPolarity polarity)
```

Remarks

If polarity is EAlignmentPolarity_ModelToMeasured, calculates the best transformation from the Model points list to the Measured points list (default).
If the polarity is EAlignmentPolarity_MeasuredToModel, calculates the best transformation from the Measured points list to the Model points list.

EFeaturesAligner::Save

Saves the EFeaturesAligner object configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.94. EFilePointerSerializer Class

Abstract interface for file-like objects.

Base Class:ESerializer

Namespace: Euresys::Open_eVision

Methods

Close	Closes the file associated with the ESerializer object.
GetWriting	Returns true if the ESerializer object has been created for writing and false otherwise.

EFilePointerSerializer::Close

Closes the file associated with the ESerializer object.

Namespace: Euresys::Open_eVision

[C++]

```
void Close(
 )
```

EFilePointerSerializer::GetWriting

Returns true if the ESerializer object has been created for writing and false otherwise.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetWriting() const
```

4.95. EFileSerializer Class

Abstract interface for file-like objects.

Base Class:ESerializer

Namespace: Euresys::Open_eVision

Methods

Close	Closes the file associated with the ESerializer object.
GetWriting	Returns true if the ESerializer object has been created for writing and false otherwise.

EFileSerializer::Close

Closes the file associated with the ESerializer object.

Namespace: Euresys::Open_eVision

[C++]

```
void Close(
 )
```

EFileSerializer::GetWriting

Returns true if the ESerializer object has been created for writing and false otherwise.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetWriting() const
```

4.96. EFilters Class

Filtering functions used to remove noise on 3D containers.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Median	In a EDepthMap or EZMap, removes noisy pixels by using a median filter. The median filter may ignore undefined values or not. In the first case, defined pixels remain defined and undefined remain undefined. In the second case, defined pixels surrounded by undefined ones may become undefined and vice-versa.
RemoveNoise	In a EDepthMap or EZMap, removes noisy pixels. A square filter window is moved over the depthmap. Within this filter window, the average and/or the standard deviation is/are calculated and a rejection criteria defined by the parameter 'method' determines which pixels will be removed. If the rejection criteria determines that a pixel has to be removed or if there is not enough valid pixels in the window filter, as specified by the parameter 'minValidRatio', the pixel is either simply removed (marked 'invalid') or replaced by the average value (within the filter window), depending on the value of 'replaceByAvg'.

EFilters::Median

In a EDepthMap or EZMap, removes noisy pixels by using a median filter.
The median filter may ignore undefined values or not. In the first case, defined pixels remain defined and undefined remain undefined. In the second case, defined pixels surrounded by undefined ones may become undefined and vice-versa.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Median(
 const EDepthMap8& sourceDepthMap,
 EDepthMap8& destinationDepthMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EDepthMap16& sourceDepthMap,
 EDepthMap16& destinationDepthMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EDepthMap32f& sourceDepthMap,
 EDepthMap32f& destinationDepthMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )
```

```
void Median(
 const EDepthMap8& sourceDepthMap,
 const ERegion& region,
 EDepthMap8& destinationDepthMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EDepthMap16& sourceDepthMap,
 const ERegion& region,
 EDepthMap16& destinationDepthMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EDepthMap32f& sourceDepthMap,
 const ERegion& region,
 EDepthMap32f& destinationDepthMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EZMap8& sourceZMap,
 EZMap8& destinationZMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EZMap16& sourceZMap,
 EZMap16& destinationZMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )

void Median(
 const EZMap32f& sourceZMap,
 EZMap32f& destinationZMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )
```

```
void Median(
 const EZMap8& sourceZMap,
 const ERegion& region,
 EZMap8& destinationZMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )
void Median(
 const EZMap16& sourceZMap,
 const ERegion& region,
 EZMap16& destinationZMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )
void Median(
 const EZMap32f& sourceZMap,
 const ERegion& region,
 EZMap32f& destinationZMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight,
 bool ignoreUndefinedPixels
 )
```

Parameters

sourceDepthMap
The source depthmap.

destinationDepthMap
The destination depthmap. It should have the same dimensions as the source depthmap.

halfOfKernelWidth
Half width of the kernel minus one (by default, halfOfKernelWidth = 1; 0 is allowed).

halfOfKernelHeight
Half height of the kernel minus one (by default, same as halfOfKernelHeight; 0 is allowed).

ignoreUndefinedPixels
Ignores the undefined pixels when computing the medians (by default, true).

region
Region to apply the function on.

sourceZMap
The source ZMap.

destinationZMap
The destination ZMap. It should have the same dimensions as the source ZMap.

EFilters::RemoveNoise

In a EDepthMap or EZMap, removes noisy pixels. A square filter window is moved over the depthmap.
Within this filter window, the average and/or the standard deviation is/are calculated and a rejection criteria defined by the parameter 'method' determines which pixels will be removed.
If the rejection criteria determines that a pixel has to be removed or if there is not enough valid pixels in the window filter, as specified by the parameter 'minValidRatio', the pixel is either simply removed (marked 'invalid') or replaced by the average value (within the filter window), depending on the value of 'replaceByAvg'.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void RemoveNoise(
 const EDepthMap16& sourceDepthMap,
 EDepthMap16& destinationDepthMap,
 Euresys::Open_eVision::Easy3D::ENoiseRemovalMethod method,
 short halfKernelSize,
 float threshold,
 float minValidRatio,
 bool replaceByAvg
 )

void RemoveNoise(
 const EDepthMap8& sourceDepthMap,
 EDepthMap8& destinationDepthMap,
 Euresys::Open_eVision::Easy3D::ENoiseRemovalMethod method,
 short halfKernelSize,
 float threshold,
 float minValidRatio,
 bool replaceByAvg
 )

void RemoveNoise(
 const EZMap16& sourceZMap,
 EZMap16& destinationZMap,
 Euresys::Open_eVision::Easy3D::ENoiseRemovalMethod method,
 short halfKernelSize,
 float threshold,
 float minValidRatio,
 bool replaceByAvg
 )

void RemoveNoise(
 const EZMap8& sourceZMap,
 EZMap8& destinationZMap,
 Euresys::Open_eVision::Easy3D::ENoiseRemovalMethod method,
 short halfKernelSize,
 float threshold,
 float minValidRatio,
 bool replaceByAvg
 )
```

Parameters

sourceDepthMap

The source depthmap.

destinationDepthMap

The destination depthmap. It should have the same dimensions as the source depthmap.

method

Noise removal method of type ENoiseRemovalMethod.

halfKernelSize

The half-size of the window filter. The filter window size (= kernel size) is halfKernelSize * 2 + 1, should be positive, smaller than (or equal to) the image size, and may not exceed 256.

threshold

The threshold used by the rejection criteria, as explained in ENoiseRemovalMethod.

minValidRatio

Required ratio of valid pixels in the filter window to process the calculation. If not enough, marks the pixel for replacement. Setting this ratio to 0.0 means that only one pixel has to be valid. The default value is 0.25 .

replaceByAvg

The marked pixels are removed by default; if this parameter is set to true, replaces the marked pixels by the average value, calculated within the filter window.

sourceZMap

The source ZMap.

destinationZMap

The destination ZMap. It should have the same dimensions as the source ZMap.

4.97. EFindFeaturePoint Class

Represents a feature point obtained from learning a model using EPatternFinder.

Namespace: Euresys::Open_eVision

Methods

EFindFeaturePoint	Constructs a default EFindFeaturePoint object.
GetGradientX	The gradient value in th X direction.
GetGradientY	The gradient value in th Y direction.
GetPosition	The position of the feature point
operator!=	Checks if two EFindFeaturePoint are stricly different.
operator=	Assignment operator for the EFindFeaturePoint.
operator==	Checks if two EFindFeaturePoint are stricly equal.

EFindFeaturePoint::EFindFeaturePoint

Constructs a default EFindFeaturePoint object.

Namespace: Euresys::Open_eVision

[C++]

```
void EFindFeaturePoint(
 )

void EFindFeaturePoint(
 float x,
 float y,
 short gradientX,
 short gradientY
 )

void EFindFeaturePoint(
 const EFindFeaturePoint& other
 )
```

Parameters

x

x coordinate of the point.

y

y coordinate of the point.

gradientX

Gradient value in the x direction.

gradientY

Gradient value in the y direction.

other

Reference to another EFindFeaturePoint used for the initialization.

Remarks

If the default constructor is used, the point is initialized to (0, 0) for both position and gradient values.

EFindFeaturePoint::GetGradientX

The gradient value in th X direction.

Namespace: Euresys::Open_eVision

[C++]

```
short GetGradientX() const
```

EFindFeaturePoint::GetGradientY

The gradient value in th Y direction.

Namespace: Euresys::Open_eVision

[C++]

```
short GetGradientY() const
```

EFindFeaturePoint::operator!=

Checks if two EFindFeaturePoint are stricly different.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EFindFeaturePoint& point
 )
```

Parameters

point

The other point.

EFindFeaturePoint::operator=

Assignment operator for the EFindFeaturePoint.

Namespace: Euresys::Open_eVision

[C++]

```
EFindFeaturePoint& operator=(
 const EFindFeaturePoint& other
 )
```

Parameters

other

-

EFindFeaturePoint::operator==

Checks if two EFindFeaturePoint are stricly equal.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const EFindFeaturePoint& point
 )
```

Parameters

point

The other point.

EFindFeaturePoint::GetPosition

The position of the feature point

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPosition() const
```

4.98. EFloatRange Class

Represents a range of floating point values.

Namespace: Euresys::Open_eVision

Methods

EFloatRange	Creates an EFloatRange object.
GetCenter	Center of the range.
GetLowerBound	Lower bound of the range.
GetSize	Size of the range.
GetUpperBound	Upper bound of the range.
IsInRange	Checks if a value is inside the range.
IsValid	Returns true if the range is defined and valid
operator=	Assignment operator
operator==	Comparison operator
SetBounds	Sets the bounds of the range.
SetFromBaseAndAbsoluteTolerance	Sets the bounds of the range from a base value and an absolute tolerance from this base value.
SetFromBaseAndRelativeTolerance	Sets the bounds of the range from a base value and a relative tolerance from this base value.

Update	Updates the range. If the value passed is outside of the range bounds, they are modified so they include the value.

EFloatRange::GetCenter

Center of the range.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCenter() const
```

EFloatRange::EFloatRange

Creates an EFloatRange object.

Namespace: Euresys::Open_eVision

```
[C++]
void EFloatRange(
 )
void EFloatRange(
 float min,
 float max
 )
void EFloatRange(
 const EFloatRange& range
 )
```

Parameters

min
Lower bound of the range.

max
Upper bound of the range.

range
Range to copy.

EFloatRange::IsInRange

Checks if a value is inside the range.

Namespace: Euresys::Open_eVision

```
[C++]
bool IsInRange(
 float value,
 bool lowerBoundInclusive,
 bool upperBoundInclusive
 )
```

Parameters

value

Value to test.

lowerBoundInclusive

Indicates if the lower bound should be considered inside the range (true) or outside (false).

upperBoundInclusive

Indicates if the upper bound should be considered inside the range (true) or outside (false).

EFloatRange::IsValid

Returns true if the range is defined and valid

Namespace: Euresys::Open_eVision

```
[C++]
bool IsValid(
 )
```

EFloatRange::GetLowerBound

Lower bound of the range.

Namespace: Euresys::Open_eVision

```
[C++]
float GetLowerBound() const
```

EFloatRange::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
EFloatRange& operator=(
 const EFloatRange& other
 )
```

Parameters

other

-

EFloatRange::operator==

Comparison operator

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EFloatRange& other
 )
```

Parameters

other

The other object.

EFloatRange::SetBounds

Sets the bounds of the range.

Namespace: Euresys::Open_eVision

```
[C++]
void SetBounds(
 float min,
 float max
 )
```

Parameters

min

Lower bound of the range.

max

Upper bound of the range.

EFloatRange::SetFromBaseAndAbsoluteTolerance

Sets the bounds of the range from a base value and an absolute tolerance from this base value.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromBaseAndAbsoluteTolerance(
 float baseValue,
 float tolerance
 )
```

Parameters

baseValue

Base value.

tolerance

Absolute tolerance around the base value. Must be positive.

Remarks

The range will be set with a lower bound of (base - tolerance) and an upper bound of (base + tolerance).

EFloatRange::SetFromBaseAndRelativeTolerance

Sets the bounds of the range from a base value and a relative tolerance from this base value.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromBaseAndRelativeTolerance(
 float baseValue,
 float tolerance
 )
```

Parameters

baseValue

Base value.

tolerance

Relative tolerance around the base value. Must be positive.

Remarks

The range will be set with a lower bound of (base - (base * tolerance)) and an upper bound of (base + (base * tolerance)).

EFloatRange::GetSize

Size of the range.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSize() const
```

EFloatRange::Update

Updates the range. If the value passed is outside of the range bounds, they are modified so they include the value.

Namespace: Euresys::Open_eVision

```
[C++]
void Update(
 float value
 )
void Update(
 const EFloatRange& other
 )
```

Parameters

value

Value to be included in the range.

other

-

EFloatRange::GetUpperBound

Upper bound of the range.

Namespace: Euresys::Open_eVision

```
[C++]
float GetUpperBound() const
```

4.99. EFont Class

Font.

Namespace: Euresys::Open_eVision

Methods

EFont	Constructs a EFont.
GetFamily	Font family name.
GetSize	Size of the font.
GetStyle	Font style.
IsValid	Wether the font is valid (Size != 0 and Family defined).

Load	Loads the EFont. The given ESerializer must have been created for reading.
operator!=	Inequality operator.
operator=	-
operator==	Equality operator.
Save	Saves the EFont. The given ESerializer must have been created for writing.
Serialize	Serializes the EFont.
SetFamily	Font family name.
SetSize	Size of the font.
SetStyle	Font style.

EFont::EFont

Constructs a EFont.

Namespace: Euresys::Open_eVision

```
[C++]
void EFont(
 )
void EFont(
 const std::string& fontFamily,
 int pointSize,
 Euresys::Open_eVision::EFontStyle style
 )
void EFont(
 const EFont& other
 )
```

Parameters

fontFamily
Font family

pointSize
Font size

style
Font style

other
-

EFont::GetFamily

EFont::SetFamily

Font family name.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetFamily() const
void SetFamily(const std::string& family)
```

EFont::IsValid

Wether the font is valid (Size != 0 and Family defined).

Namespace: Euresys::Open_eVision

```
[C++]
bool IsValid(
 )
```

EFont::Load

Loads the EFont. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EFont::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const EFont& other
 )
```

Parameters

other

Other font to compare with.

EFont::operator=

-

Namespace: Euresys::Open_eVision

```
[C++]
EFont& operator=(
 const EFont& other
 )
```

Parameters

other

-

EFont::operator==

Equality operator.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EFont& other
 )
```

Parameters

other

Other font to compare with.

EFont::Save

Saves the EFont. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

EFont::Serialize

Serializes the EFont.

Namespace: Euresys::Open_eVision

[C++]

```
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer
Serializer

EFont::GetSize

EFont::SetSize

Size of the font.

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize() const
```

```
void SetSize(int size)
```

EFont::GetStyle

EFont::SetStyle

Font style.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EFontStyle GetStyle() const
void SetStyle(Euresys::Open_eVision::EFontStyle style)
```

4.100. EFoundPattern Class

Represents a single instance of the pattern in the search field, as returned by the EasyFind finding process.

Remarks

EPatternFinder::Find returns a collection of instances of this class. An EFoundPattern object represents one found instance, with all the needed information about it.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the found pattern, in image coordinates.
DrawWithCurrentPen	Draws the found pattern, in image coordinates.
EFoundPattern	Constructs a EFoundPattern object.
GetAngle	Angle of the found instance, in the current angle unit.
GetCenter	Reference point of the instance.
GetDrawBoundingBox	Flag indicating if the EFoundPattern::Draw method must draw the bounding box of the FoundPattern.
GetDrawCenter	Flag indicating if the EFoundPattern::Draw method must draw the center of the FoundPattern.
GetDrawFeaturePoints	Flag indicating if the EFoundPattern::Draw method must draw the feature points of the EFoundPattern object.
GetQuadrangle	Returns the corners of the bounding box of the found pattern.
GetScale	Scaling factor of the found pattern, in units (not percents).
GetScore	Matching score of the found pattern, in units (not percents).

operator!=	Inequality operator.
operator=	Copies all the data from another EFoundPattern object into the current EFoundPattern object
operator==	Equality operator.
SetDrawBoundingBox	Flag indicating if the EFoundPattern::Draw method must draw the bounding box of the FoundPattern.
SetDrawCenter	Flag indicating if the EFoundPattern::Draw method must draw the center of the FoundPattern.
SetDrawFeaturePoints	Flag indicating if the EFoundPattern::Draw method must draw the feature points of the EFoundPattern object.
ToRegion	Creates an ERegion from the EFoundPattern. The ERegion represents all the pixels which are within the bounding box of the Pattern.

EFoundPattern::GetAngle

Angle of the found instance, in the current angle unit.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAngle() const
```

Remarks

Read-only. This returned value is always comprised in the range [- a half turn, + a half turn].

EFoundPattern::GetCenter

Reference point of the instance.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetCenter() const
```

Remarks

By default, this is its center. If the property EPatternFinder::Pivot has been changed in the EPatternFinder, the point returns the pivot_ in the instance.

EFoundPattern::Draw

Draws the found pattern, in image coordinates.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to 0, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

This method draws different features of the EFoundPattern, according to the value of properties EFoundPattern::DrawFeaturePoints, EFoundPattern::DrawCenter, EFoundPattern::DrawBoundingBox. The zoomX, zoomY, panX and panY parameters can be used to scale and/or translate the drawing operations. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EFoundPattern::GetDrawBoundingBox

EFoundPattern::SetDrawBoundingBox

Flag indicating if the EFoundPattern::Draw method must draw the bounding box of the FoundPattern.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetDrawBoundingBox() const
void SetDrawBoundingBox(bool drawBoundingBox)
```

Remarks

The default value is true.

EFoundPattern::GetDrawCenter

EFoundPattern::SetDrawCenter

Flag indicating if the EFoundPattern::Draw method must draw the center of the FoundPattern.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetDrawCenter() const
void SetDrawCenter(bool drawCenter)
```

Remarks

The default value is true.

EFoundPattern::GetDrawFeaturePoints

EFoundPattern::SetDrawFeaturePoints

Flag indicating if the EFoundPattern::Draw method must draw the feature points of the EFoundPattern object.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetDrawFeaturePoints() const
void SetDrawFeaturePoints(bool drawFeaturePoints)
```

Remarks

The default value is false.

EFoundPattern::DrawWithCurrentPen

This method is deprecated.

Draws the found pattern, in image coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to 0, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

This method draws different features of the EFoundPattern, according to the value of properties EFoundPattern::DrawFeaturePoints, EFoundPattern::DrawCenter, EFoundPattern::DrawBoundingBox. The zoomX, zoomY, panX and panY parameters can be used to scale and/or translate the drawing operations. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EFoundPattern::EFoundPattern

Constructs a EFoundPattern object.

Namespace: Euresys::Open_eVision

[C++]

```
void EFoundPattern(
 )
void EFoundPattern(
 const EFoundPattern& other
 )
```

Parameters

other

EFoundPattern object to be copied

EFoundPattern::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EFoundPattern& other
 )
```

Parameters

other

Instance to compare to.

EFoundPattern::operator=

Copies all the data from another EFoundPattern object into the current EFoundPattern object

Namespace: Euresys::Open_eVision

[C++]

```
EFoundPattern& operator=(
 const EFoundPattern& other
 )
```

Parameters

other

EFoundPattern object to be copied

EFoundPattern::operator==

Equality operator.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EFoundPattern& other
 )
```

Parameters

other

Instance to compare to.

EFoundPattern::GetQuadrangle

Returns the corners of the bounding box of the found pattern.

Namespace: Euresys::Open_eVision

```
[C++]
EQuadrangle GetQuadrangle() const
```

EFoundPattern::GetScale

Scaling factor of the found pattern, in units (not percents).

Namespace: Euresys::Open_eVision

```
[C++]
float GetScale() const
```

EFoundPattern::GetScore

Matching score of the found pattern, in units (not percents).

Namespace: Euresys::Open_eVision

```
[C++]
float GetScore() const
```

Remarks

The matching score range is [-1.0..1.0].

EFoundPattern::ToRegion

Creates an ERegion from the EFoundPattern. The ERegion represents all the pixels which are within the bounding box of the Pattern.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion ToRegion(
 )
```

4.101. EFourierTransformer Class

Manages direct and inverse Fast Fourier Transforms.

Namespace: Euresys::Open_eVision

Methods

DirectTransform	Computes the direct transform.
EFourierTransformer	Constructs an EFourierTransformer object initialized to its default values.
GetFrequentialDomainFormat	Gets/Sets the data format used to represent frequential domain images. Default value is EFrequentialDomainFormat_Packed.
InverseTransform	Computes the inverse transform.
Load	Loads the EFourierTransformer. The given ESerializer must have been created for reading.
operator!=	Checks if this EFourierTransformer instance is not strictly equal to another
operator=	Assignment operator.
operator==	Checks if this EFourierTransformer instance is strictly equal to another
Save	Saves the EFourierTransformer. The given ESerializer must have been created for writing.
SetFrequentialDomainFormat	Gets/Sets the data format used to represent frequential domain images. Default value is EFrequentialDomainFormat_Packed.

EFourierTransformer::DirectTransform

Computes the direct transform.

Namespace: Euresys::Open_eVision

[C++]

```
void DirectTransform(
 const EROIBW8& spatialImage,
 EROIBW32f& frequentialImage
 )

void DirectTransform(
 const EROIBW16& spatialImage,
 EROIBW32f& frequentialImage
 )

void DirectTransform(
 const EROIBW32f& spatialImage,
 EROIBW32f& frequentialImage
 )
```

Parameters

spatialImage

input image in spatial domain.

frequentialImage

output image in frequential domain. Must be of the same size as spatialImage if EFourierTransformer::FrequentialDomainFormat is EFrequentialDomainFormat_Packed or twice larger if it is EFrequentialDomainFormat_ComplexExtended.

Remarks

No scaling is applied, scaling applied in EFourierTransformer::InverseTransform.

EFourierTransformer::EFourierTransformer

Constructs an EFourierTransformer object initialized to its default values.

Namespace: Euresys::Open_eVision

[C++]

```
void EFourierTransformer(
 )

void EFourierTransformer(
 const EFourierTransformer& other
 )
```

Parameters

other

Another EFourierTransformer.

EFourierTransformer::GetFrequentialDomainFormat

EFourierTransformer::SetFrequentialDomainFormat

Gets/Sets the data format used to represent frequential domain images. Default value is EFrequentialDomainFormat_Packed.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EFrequentialDomainFormat GetFrequentialDomainFormat() const

void SetFrequentialDomainFormat(Euresys::Open_eVision::EFrequentialDomainFormat format)
```

EFourierTransformer::InverseTransform

Computes the inverse transform.

Namespace: Euresys::Open_eVision

[C++]

```
void InverseTransform(
 const EROIBW32f& frequentialImage,
 EROIBW8& spatialImage
 )

void InverseTransform(
 const EROIBW32f& frequentialImage,
 EROIBW16& spatialImage
 )

void InverseTransform(
 const EROIBW32f& frequentialImage,
 EROIBW32f& spatialImage
 )
```

Parameters

frequentialImage

input image in frequential domain. frequentialImage's width must be a multiple of 2 if EFourierTransformer::FrequentialDomainFormat is EFrequentialDomainFormat_ ComplexExtended.

spatialImage

output image in spatial domain. Must be of the same size as frequentialImage if EFourierTransformer::FrequentialDomainFormat is EFrequentialDomainFormat_Packed or twice thinner if it is EFrequentialDomainFormat_ComplexExtended.

Remarks

Scaling is applied, no scaling applied in EFourierTransformer::DirectTransform.

EFourierTransformer::Load

Loads the EFourierTransformer. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EFourierTransformer::operator!=

Checks if this EFourierTransformer instance is not strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const EFourierTransformer& other
 )
```

Parameters

other
Reference to the other EFourierTransformer instance

EFourierTransformer::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EFourierTransformer& operator=(
 const EFourierTransformer& other
 )
```

Parameters

other

Another EFourierTransformer.

EFourierTransformer::operator==

Checks if this EFourierTransformer instance is strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EFourierTransformer& other
 )
```

Parameters

other

Reference to the other EFourierTransformer instance

EFourierTransformer::Save

Saves the EFourierTransformer. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.102. EFrame Class

Represents a geometrical frame of reference as well as the parameters needed to transform from/to local and global coordinates. It contains a point and an angle and serves as a base class for geometrical elements.

Base Class:EPoint
Derived Class(es):ECircleELineEPolygonERectangleEWedge
Namespace: Euresys::Open_eVision

Methods

CopyTo	Copies all the data from the current EFrame object into another EFrame object, and returns it.
EFrame	Constructs a EFrame object.
GetAngle	Orientation of the frame.
GetCenterX	Abscissa of the origin point of the frame.
GetCenterY	Ordinate of the origin point of the frame.
GetScale	Horizontal sensor resolution, in pixels per unit.
GlobalToLocal	Transforms a geometrical element from global to local coordinates (world to local).
Load	Load the EFrame configuration. The given ESerializer must have been created for reading.
LocalToGlobal	Transforms a geometrical element from local to global coordinates (local to world).
operator=	Copies all the data from another EFrame object into the current EFrame object.
Save	Save the EFrame configuration. The given ESerializer must have been created for writing.
SetAngle	Orientation of the frame.
SetScale	Horizontal sensor resolution, in pixels per unit.

EFrame::GetAngle

EFrame::SetAngle

Orientation of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
void SetAngle(float angle)
```

EFrame::GetCenterX

Abscissa of the origin point of the frame.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCenterX() const
```

EFrame::GetCenterY

Ordinate of the origin point of the frame.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCenterY() const
```

EFrame::CopyTo

Copies all the data from the current EFrame object into another EFrame object, and returns it.

Namespace: Euresys::Open_eVision

```
[C++]
void CopyTo(
 EFrame& other
 )
EFrame* CopyTo(
 EFrame* other
 )
```

Parameters

other

Pointer to the EFrame object in which the current EFrame object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EFrame object will be created and returned.

EFrame::EFrame

Constructs a EFrame object.

Namespace: Euresys::Open_eVision

```
[C++]
void EFrame(
 )
```

```
void EFrame(
 float centerX,
 float centerY,
 float angle,
 float scale
 )

void EFrame(
 const EPoint& center,
 float angle,
 float scale
 )

void EFrame(
 const EFrame& frame
 )
```

Parameters

centerX
Abscissa of the origin point of the frame.

centerY
Ordinate of the origin point of the frame.

angle
Orientation of the frame.

scale
Horizontal sensor resolution.

center
Coordinates of the origin point of the frame.

frame
Pre-existing EFrame object used by the copy constructor.

EFrame::GlobalToLocal

Transforms a geometrical element from global to local coordinates (world to local).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GlobalToLocal(
 const EPoint& global
 )

EFrame GlobalToLocal(
 const EFrame& global
 )
```

Parameters

global
The element, expressed in global coordinates.

EFrame::Load

Load the EFrame configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EFrame::LocalToGlobal

Transforms a geometrical element from local to global coordinates (local to world).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint LocalToGlobal(
 const EPoint& local
 )

EFrame LocalToGlobal(
 const EFrame& local
 )

ELine LocalToGlobal(
 const ELine& local
 )

ECircle LocalToGlobal(
 const ECircle& local
 )
```

Parameters

local
The element, expressed in local coordinates.

EFrame::operator=

Copies all the data from another EFrame object into the current EFrame object.

Namespace: Euresys::Open_eVision

```
[C++]
EFrame& operator=(
 const EFrame& frame
 )
```

Parameters

frame
EFrame object to be copied.

EFrame::Save

Save the EFrame configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EFrame::GetScale

EFrame::SetScale

Horizontal sensor resolution, in pixels per unit.

Namespace: Euresys::Open_eVision

```
[C++]
float GetScale() const
```

```
void SetScale(float scale)
```

4.103. EFrameShape Class

Manages a complete context for measuring frame shapes.

Remarks

This class allows the grouping of several gauges or other frames.

Base Class:EShape

Namespace: Euresys::Open_eVision

Methods

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copy operator.
Drag	Moves a handle to a new position and updates the position parameters of the shape.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
EFrameShape	Construct a EFrameShape object.
GetAngle	The angle of the frame.
GetCenter	Origin point coordinates of the frame.
GetCenterX	Gets the origin point abscissa of the frame.
GetCenterY	Gets the origin point ordinate of the frame.
GetScale	The scale of the frame.
GetSizeX	Frame X-axis length.
GetSizeY	Frame Y-axis length.
GetType	Type of the frame.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
operator=	Assignement operator.
Set	Sets the coordinates of the origin point and the orientation of the frame.
SetAngle	The angle of the frame.
SetCenter	Origin point coordinates of the frame.
SetCenterXY	Sets the origin point coordinates of the frame.

SetScale	The scale of the frame.
SetSize	Sets the frame size.

EFrameShape::GetAngle

EFrameShape::SetAngle

The angle of the frame.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAngle() const
void SetAngle(float angle)
```

EFrameShape::GetCenter

EFrameShape::SetCenter

Origin point coordinates of the frame.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetCenter() const
void SetCenter(const EPoint& point)
```

EFrameShape::GetCenterX

Gets the origin point abscissa of the frame.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCenterX() const
```

EFrameShape::GetCenterY

Gets the origin point ordinate of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

EFrameShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

EFrameShape::CopyTo

Copy operator.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EFrameShape& other,
 bool recursive
 )

EFrameShape* CopyTo(
 EFrameShape* other,
 bool recursive
 )
```

Parameters

other

EFrameShape object to copy to.

recursive

true if the daughter shapes have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EFrameShape object will be created and returned.

EFrameShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int cursorX,
 int cursorY
 )
```

Parameters

cursorX

Current cursor coordinates.

cursorY

Current cursor coordinates.

EFrameShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the shape must be displayed, as defined by EDrawingMode.

daughters

true if the daughter shapes are to be displayed also.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EFrameShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EFrameShape::EFrameShape

Construct a EFrameShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void EFrameShape(
 )
void EFrameShape(
 const EFrameShape& frameShape
 )
```

Parameters

frameShape

Pre-existing EFrameShape object used by the copy constructor.

Remarks

With the default constructor, all parameters are initialized to their respective default value. With the copy constructor, the constructed frame shape measurement context is based on a pre-existing EFrameShape object. By default, the daughter shapes are also copied. Use the EFrameShape::CopyTo method to disable explicitly the daughter shapes copy.

EFrameShape::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 bool daughter
 )
```

Parameters

daughter

true if the daughters shapes handles have to be considered as well.

EFrameShape::operator=

Assignement operator.

Namespace: Euresys::Open_eVision

[C++]

```
EFrameShape& operator=(
 const EFrameShape& other
 )
```

Parameters

other

EFrameShape object to copy.

Remarks

By default, the daughter shapes are also copied. Use the EFrameShape::CopyTo method to disable explicitly the daughter shapes copy.

EFrameShape::GetScale

EFrameShape::SetScale

The scale of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetScale() const

void SetScale(float scale)
```

EFrameShape::Set

Sets the coordinates of the origin point and the orientation of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
void Set(
 const EPoint& center,
 float angle,
 float scale
 )
```

Parameters

center

Coordinates of the origin point of the frame. The default value is (0,0).

angle

Rotation angle of the frame. The default value is 0.

scale

Horizontal sensor resolution, in pixels per unit

EFrameShape::SetCenterXY

Sets the origin point coordinates of the frame.

Namespace: Euresys::Open_eVision

```
[C++]
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Abscissa of the origin point of the frame. Default value is 0.

centerY

Ordinate of the origin point of the frame. Default value is 0.

EFrameShape::SetSize

Sets the frame size.

Namespace: Euresys::Open_eVision

```
[C++]
void SetSize(
 float sizeX,
 float sizeY
 )
```

Parameters

sizeX

Frame X-axis length. The default value is 100.

sizeY

Frame Y-axis length. By default, both axes have the same length.

Remarks

By default, both frame axis value are set to 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EFrameShape::GetSizeX

Frame X-axis length.

Namespace: Euresys::Open_eVision

```
[C++]
float GetSizeX()
```

Remarks

By default, both frame axis values are set to 100, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

EFrameShape::GetSizeY

Frame Y-axis length.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSizeY()
```

Remarks

By default, both frame axis values are set to 100, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

EFrameShape::GetType

Type of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

4.104. EGDIPlusDrawAdapter Class

This class is deprecated.

Deprecated: This class was renamed EWindowsDrawAdapter.

Base Class:EWindowsDrawAdapter

Namespace: Euresys::Open_eVision

Methods

EGDIPlusDrawAdapter	-

EGDIPlusDrawAdapter::EGDIPlusDrawAdapter

This method is deprecated.

-

Namespace: Euresys::Open_eVision

[C++]

```
void EGDIPlusDrawAdapter(
 HDC dc
 )
```

Parameters

dc

-

4.105. EGenericDrawAdapter Class

A type of draw adapter able to make rendering in images and yielding similar results on all platforms. The EGenericDrawAdapter draws directly in an EROIC24A and uses CPU calculation.

Remarks

EGenericDrawAdapter only support drawing in EC24A images and ROIs.

Base Class:EDrawAdapter

Namespace: Euresys::Open_eVision

Methods

Arc	Draws an arc defined by a rectangle, angle, and amplitude.
BackedText	Draws a text with a background.
DrawPoint	Draws a point at the given coordinate.
EGenericDrawAdapter	Constructs an instance of EGenericDrawAdapter that will render to an EROIC24A.
Ellipse	Draws an ellipse.
FilledEllipse	Fills an ellipse.
FilledPolygon	Fills a polygon.
FilledRectangle	Fills a rectangle.
GetFont	Default font. Defaults to "Roboto" in EGenericDrawAdapter. That font is distributed with Open eVision and will be available on all platforms.
GetTextSize	Size of the given text using the default font (EGenericDrawAdapter::Font).
GetUseAntialiasing	Indicates if anti-aliasing should be used for rendering.
Image	Draws an image.
Line	Draws a line between two points.
Polygon	Draws a polygon.
Rectangle	Draws a rectangle.

SetFont	Default font. Defaults to "Roboto" in EGenericDrawAdapter. That font is distributed with Open eVision and will be available on all platforms.
SetUseAntialiasing	Indicates if anti-aliasing should be used for rendering.
Text	Draws a text.
UseCurrentBrush	Resets the brush to a white opaque brush.
UseCurrentPen	Resets the pen to a black opaque pen.

EGenericDrawAdapter::Arc

Draws an arc defined by a rectangle, angle, and amplitude.

Namespace: Euresys::Open_eVision

[C++]

```
void Arc(
 int orgX,
 int orgY,
 int width,
 int height,
 float startAngle,
 float amplitude,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

startAngle
Starting angle of the arc in radians

amplitude
Amplitude of the arc in radians

pen
Optional pen to use

EGenericDrawAdapter::BackedText

Draws a text with a background.

Namespace: Euresys::Open_eVision

[C++]

```
void BackedText(
 const std::string& text,
 int x,
 int y,
 EBrush textBrush,
 EBrush backgroundBrush
 )
void BackedText(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush,
 EBrush backgroundBrush
 )
```

Parameters

text
Text to draw.

x
X position of the text.

y
Y position of the text.

textBrush
Optional brush to use for the color of the text.

backgroundBrush
Optional brush to use for the background.

orientation
Orientation of the text.

EGenericDrawAdapter::DrawPoint

Draws a point at the given coordinate.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPoint(
 int x1,
 int y1,
 EPen pen
 )
```

Parameters

x1
X coordinate

y1
Y coordinate

pen
Optional pen to use

EGenericDrawAdapter::EGenericDrawAdapter

Constructs an instance of EGenericDrawAdapter that will render to an EROIC24A.

Namespace: Euresys::Open_eVision

[C++]

```
void EGenericDrawAdapter(
 EROIC24A* pImage,
 bool useAntiAliasing
 )
```

Parameters

pImage
The image to render to

useAntiAliasing
Indicates if anti-aliasing should be used for rendering.

EGenericDrawAdapter::Ellipse

Draws an ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void Ellipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle containing the ellipse

orgY
Y origin of the rectangle containing the ellipse

width
Width of the rectangle containing the ellipse

height
Height of the rectangle containing the ellipse

pen
Optional pen to use

EGenericDrawAdapter::FilledEllipse

Fills an ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledEllipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle containing the ellipse

orgY
Y origin of the rectangle containing the ellipse

width
Width of the rectangle containing the ellipse

height
Height of the rectangle containing the ellipse

pen
Optional pen to use for drawing the contour of the ellipse

brush
Optional pen to use for drawing the inside of the ellipse

EGenericDrawAdapter::FilledPolygon

Fills a polygon.

Namespace: Euresys::Open_eVision

```
[C++]
void FilledPolygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen,
 EBrush brush
 )
```

Parameters

points
Points of the polygon

pen
Optional pen to use for drawing the contour of the polygon

brush
Optional pen to use for drawing the inside of the polygon

EGenericDrawAdapter::FilledRectangle

Fills a rectangle.

Namespace: Euresys::Open_eVision

```
[C++]
void FilledRectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

pen
Optional pen to use for drawing the contour of the rectangle

brush
Optional brush to use for filling the inside of the rectangle

EGenericDrawAdapter::GetFont

EGenericDrawAdapter::SetFont

Default font. Defaults to "Roboto" in EGenericDrawAdapter. That font is distributed with Open eVision and will be available on all platforms.

Namespace: Euresys::Open_eVision

```
[C++]
EFont GetFont() const
void SetFont(const EFont& font)
```

EGenericDrawAdapter::GetTextSize

Size of the given text using the default font (EGenericDrawAdapter::Font).

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetTextSize(
 const std::string& text
 )
```

Parameters

text

Text

EGenericDrawAdapter::Image

Draws an image.

Namespace: Euresys::Open_eVision

```
[C++]
void Image(
 const EBaseROI& image
 )
void Image(
 const EROIBW8& image,
 EBW8Vector* pColorScale
 )
```

```
void Image(
 const EROIBW8& image,
 EC24Vector* pColorScale
 )
void Image(
 const EBaseROI& image,
 float orgX,
 float orgY,
 float width,
 float height
 )
void Image(
 const EROIBW8& image,
 EC24Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
void Image(
 const EROIBW8& image,
 EBW8Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
```

Parameters

image
Image.

pColorScale
Color scale to draw a grayscale image with.

orgX
X coordinate of the point where to draw the image.

orgY
Y coordinate of the point where to draw the image.

width
Width of the destination rectangle in which to draw the image.

height
Height of the destination rectangle in which to draw the image.

EGenericDrawAdapter::Line

Draws a line between two points.

Namespace: Euresys::Open_eVision

```
[C++]
void Line(
 int x1,
 int y1,
 int x2,
 int y2,
 EPen pen
 )
```

Parameters

x1
X coordinate of line origin point

y1
Y coordinate of line origin point

x2
X coordinate of line end point

y2
Y coordinate of line end point

pen
Optional pen to use

EGenericDrawAdapter::Polygon

Draws a polygon.

Namespace: Euresys::Open_eVision

```
[C++]
void Polygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen
 )
```

Parameters

points
Points of the polygon

pen
Optional pen to use

EGenericDrawAdapter::Rectangle

Draws a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void Rectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

pen
Optional pen to use

EGenericDrawAdapter::Text

Draws a text.

Namespace: Euresys::Open_eVision

[C++]

```
void Text(
 const std::string& text,
 int x,
 int y,
 EBrush textBrush
 )

void Text(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush
 )
```

Parameters

text
Text to draw.

x
X position of the text.

y
Y position of the text.

textBrush
Optional brush to use for the color of the text.

orientation
Orientation of the text in radians.

EGenericDrawAdapter::GetUseAntialiasing

EGenericDrawAdapter::SetUseAntialiasing

Indicates if anti-aliasing should be used for rendering.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetUseAntialiasing()
void SetUseAntialiasing(bool value)
```

EGenericDrawAdapter::UseCurrentBrush

Resets the brush to a white opaque brush.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentBrush(
 )
```

EGenericDrawAdapter::UseCurrentPen

Resets the pen to a black opaque pen.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentPen(
 )
```

4.106. EGrabberDepthMap16 Class

The EGrabberDepthMap16 class is used to wrap an EGrabber buffer whose pixel type is "Coord3D_C16". It can be used in Open eVision processing as an EDepthMap16 object.

Base Class:EDepthMap16

Namespace: Euresys::Open_eVision::EGrabberBridge

Methods

EGrabberDepthMap16	Constructs an EGrabberDepthMap16.

EGrabberDepthMap16::EGrabberDepthMap16

Constructs an EGrabberDepthMap16.

Namespace: Euresys::Open_eVision::EGrabberBridge

[C++]

```
void EGrabberDepthMap16(
 const BufferInfo& infos
 )

void EGrabberDepthMap16(
 FormatConvert& converter,
 const BufferInfo& infos
 )
```

Parameters

infos

Information pertaining to an EGrabber buffer

converter

EGrabber format converter

Remarks

The FormatConverter class allows automatic pixel format conversions. If no format converter is provided, the pixel format of the buffer must be "Coord3D_C16"

4.107. EGrabberDepthMap8 Class

The EGrabberDepthMap8 class is used to wrap an EGrabber buffer whose pixel type is "Coord3D_C8". It can be used in Open eVision processing as an EDepthMap8 object.

Base Class:EDepthMap8

Namespace: Euresys::Open_eVision::EGrabberBridge

Methods

EGrabberDepthMap8	Constructs an EGrabberDepthMap8.

EGrabberDepthMap8::EGrabberDepthMap8

Constructs an EGrabberDepthMap8.

Namespace: Euresys::Open_eVision::EGrabberBridge

```
[C++]
void EGrabberDepthMap8(
 const BufferInfo& infos
 )
void EGrabberDepthMap8(
 FormatConvert& converter,
 const BufferInfo& infos
 )
```

Parameters

infos

Information pertaining to an EGrabber buffer

converter

EGrabber format converter

Remarks

The FormatConverter class allows automatic pixel format conversions. If no format converter is provided, the pixel format of the buffer must be "Coord3D_C8"

4.108. EGrabberImageBW16 Class

The EGrabberImageBW16 class is used to wrap an EGrabber buffer whose pixel type is "Mono16". It can be used in Open eVision processing as an EImageBW16 object.

Base Class:EImageBW16

Namespace: Euresys::Open_eVision::EGrabberBridge

Methods

EGrabberImageBW16	Constructs an EGrabberImageBW16.

EGrabberImageBW16::EGrabberImageBW16

Constructs an EGrabberImageBW16.

Namespace: Euresys::Open_eVision::EGrabberBridge

[C++]

```
void EGrabberImageBW16(
 const BufferInfo& infos
 )

void EGrabberImageBW16(
 FormatConvert& converter,
 const BufferInfo& infos
 )
```

Parameters

infos

Information pertaining to an EGrabber buffer

converter

EGrabber format converter

Remarks

The FormatConverter class allows automatic pixel format conversions. If no format converter is provided, the pixel format of the buffer must be "Mono16"

4.109. EGrabberImageBW8 Class

The EGrabberImageBW8 class is used to wrap an EGrabber buffer whose pixel type is "Mono8". It can be used in Open eVision processing as an EImageBW8 object.

Base Class:EImageBW8

Namespace: Euresys::Open_eVision::EGrabberBridge

Methods

EGrabberImageBW8	Constructs an EGrabberImageBW8.

EGrabberImageBW8::EGrabberImageBW8

Constructs an EGrabberImageBW8.

Namespace: Euresys::Open_eVision::EGrabberBridge

[C++]

```
void EGrabberImageBW8(
 const BufferInfo& infos
 )

void EGrabberImageBW8(
 FormatConvert& converter,
 const BufferInfo& infos
 )
```

Parameters

infos

Information pertaining to an EGrabber buffer

converter

EGrabber format converter

Remarks

The FormatConverter class allows automatic pixel format conversions. If no format converter is provided, the pixel format of the buffer must be "Mono8"

4.110. EGrabberImageC24 Class

The EGrabberImageC24 class is used to wrap an EGrabber buffer whose pixel type is "BGR8". It can be used in Open eVision processing as an EImageC24 object.

Base Class:EImageC24

Namespace: Euresys::Open_eVision::EGrabberBridge

Methods

EGrabberImageC24	Constructs an EGrabberImageC24.

EGrabberImageC24::EGrabberImageC24

Constructs an EGrabberImageC24.

Namespace: Euresys::Open_eVision::EGrabberBridge

[C++]

```
void EGrabberImageC24(
 const BufferInfo& infos
 )

void EGrabberImageC24(
 FormatConvert& converter,
 const BufferInfo& infos
 )
```

Parameters

infos

Information pertaining to an EGrabber buffer

converter

EGrabber format converter

Remarks

The FormatConverter class allows automatic pixel format conversions. If no format converter is provided, the pixel format of the buffer must be "BGR8"

4.111. EGrayscaleDoubleThresholdSegmenter Class

Segments an image using a double threshold on a grayscale image.

Remarks

This segmenter is applicable to EROIBW8 and EROIBW16 grayscale images. It produces coded images with three layers: The Black layer (usually, with index 0) contains the unmasked pixels having a gray value strictly below the low threshold value; the White layer (usually, with index 2) contains the unmasked pixels having a gray value above or equal to the high threshold value; and the Neutral layer (usually, with index 1) contains the remaining unmasked pixels.

Base Class:EThreeLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetHighThreshold	Value of the high threshold.
GetLowThreshold	Value of the low threshold.
Load	Load the EGrayscaleDoubleThresholdSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EGrayscaleDoubleThresholdSegmenter configuration. The given ESerializer must have been created for writing.
SetHighThreshold	Value of the high threshold.
SetLowThreshold	Value of the low threshold.

EGrayscaleDoubleThresholdSegmenter::GetHighThreshold

EGrayscaleDoubleThresholdSegmenter::SetHighThreshold

Value of the high threshold.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetHighThreshold() const
void SetHighThreshold(OEV_UINT32 threshold)
```

EGrayscaleDoubleThresholdSegmenter::Load

Load the EGrayscaleDoubleThresholdSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EGrayscaleDoubleThresholdSegmenter::GetLowThreshold

EGrayscaleDoubleThresholdSegmenter::SetLowThreshold

Value of the low threshold.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
OEV_UINT32 GetLowThreshold() const
void SetLowThreshold(OEV_UINT32 threshold)
```

EGrayscaleDoubleThresholdSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool operator==(
 const EGrayscaleDoubleThresholdSegmenter& other
 )
```

Parameters

other
Other segmenter to compare to.

EGrayscaleDoubleThresholdSegmenter::Save

Save the EGrayscaleDoubleThresholdSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.112. EGrayscaleSingleThresholdSegmenter Class

Segments an image using a single threshold on a grayscale image.

Remarks

This segmenter is applicable to EROIBW8 and EROIBW16 grayscale images. It produces coded images with two layers: the black layer (usually, with index 0) contains the unmasked pixels having a gray value strictly below the threshold value; and the white layer (usually, with index 1) contains the remaining unmasked pixels, i.e. unmasked pixels having a gray value greater or equal to the threshold value. The default thresholding method is minimum residue. If another method is required, the EGrayscaleSingleThresholdSegmenter::Mode property must be set prior to encoding.

Base Class:ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetAbsoluteThreshold	Value of the threshold to be used in the case of absolute thresholding.
GetLastThreshold	Retrieves the actual threshold value that was used for the last image that got encoded by the segmenter.
GetMode	Threshold selection mode.
GetRelativeThreshold	Fraction of the image pixels that belongs to the black layer in the case of relative thresholding.

IsFirstApplication	Checks whether the segmenter has already been used to segment an image.
Load	Load the EGrayscaleSingleThresholdSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EGrayscaleSingleThresholdSegmenter configuration. The given ESerializer must have been created for writing.
SetAbsoluteThreshold	Value of the threshold to be used in the case of absolute thresholding.
SetMode	Threshold selection mode.
SetRelativeThreshold	Fraction of the image pixels that belongs to the black layer in the case of relative thresholding.

EGrayscaleSingleThresholdSegmenter::GetAbsoluteThreshold

EGrayscaleSingleThresholdSegmenter::SetAbsoluteThreshold

Value of the threshold to be used in the case of absolute thresholding.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetAbsoluteThreshold() const

void SetAbsoluteThreshold(OEV_UINT32 threshold)
```

EGrayscaleSingleThresholdSegmenter::IsFirstApplication

Checks whether the segmenter has already been used to segment an image.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool IsFirstApplication(
 )
```

EGrayscaleSingleThresholdSegmenter::GetLastThreshold

Retrieves the actual threshold value that was used for the last image that got encoded by the segmenter.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetLastThreshold() const
```

Remarks

A call to this method will result in an exception if it is the first time the segmenter is applied. To check whether the segmenter has already been applied, call the EGrayscaleSingleThresholdSegmenter::IsFirstApplication method.

EGrayscaleSingleThresholdSegmenter::Load

Load the EGrayscaleSingleThresholdSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EGrayscaleSingleThresholdSegmenter::GetMode

EGrayscaleSingleThresholdSegmenter::SetMode

Threshold selection mode.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
Euresys::Open_eVision::EGrayscaleSingleThreshold GetMode() const

void SetMode(Euresys::Open_eVision::EGrayscaleSingleThreshold mode)
```

EGrayscaleSingleThresholdSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool operator==(
 const EGrayscaleSingleThresholdSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to.

EGrayscaleSingleThresholdSegmenter::GetRelativeThreshold

EGrayscaleSingleThresholdSegmenter::SetRelativeThreshold

Fraction of the image pixels that belongs to the black layer in the case of relative thresholding.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
float GetRelativeThreshold() const

void SetRelativeThreshold(float threshold)
```

EGrayscaleSingleThresholdSegmenter::Save

Save the EGrayscaleSingleThresholdSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.113. EGridDecimator Class

Decimation of an EPointCloud.
The grid decimator decimates a point cloud by creating a new point cloud containing a point for each grid cell where at least one point exists in the initial point cloud.
For each grid cell, the point in the new pointCloud is the centroid of the points in the initial point cloud. If the cloud has attributes, the attributes of each new point are the ones of the closest point of the initial point cloud in the same grid.

Base Class:EDecimator

Namespace: Euresys::Open_eVision::Easy3D

Methods

Decimate	Decimates a given EPointCloud and writes the result into a new point cloud.
EGridDecimator	Creates an EGridDecimator object. If the required cell size is not specified, the default value is 10, 10, 10.
GetCellSize	Gets/sets the size of the cells in the grid.
operator!=	test inequality between two EGridDecimator.
operator=	Assignment operator.
operator==	test equality between two EGridDecimator.
SetCellSize	Gets/sets the size of the cells in the grid.

EGridDecimator::GetCellSize

EGridDecimator::SetCellSize

Gets/sets the size of the cells in the grid.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetCellSize() const

void SetCellSize(const E3DPoint& cellSize)
```

EGridDecimator::Decimate

Decimates a given EPointCloud and writes the result into a new point cloud.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Decimate(
 const EPointCloud& cloudIn,
 EPointCloud& cloudOut
 )
```

Parameters

cloudIn

The input point cloud.

cloudOut

The output point cloud.

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

EGridDecimator::EGridDecimator

Creates an EGridDecimator object. If the required cell size is not specified, the default value is 10, 10, 10.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EGridDecimator(
 )

void EGridDecimator(
 const E3DPoint& cellSize
 )

void EGridDecimator(
 float isotropicCellSize
 )

void EGridDecimator(
 const EGridDecimator& other
 )
```

Parameters

cellSize

E3DPoint whose dimensions represent the x,y,z size of the cell

isotropicCellSize

float representing the x,y and z size of the cell

other

Reference to the EGridDecimator object used for the initialization.

EGridDecimator::operator!=

test inequality between two EGridDecimator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator!=(
 const EDecimator& other
 )
```

Parameters

other

the EGridDecimator to be compared with

EGridDecimator::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EGridDecimator& operator=(
 const EGridDecimator& other
 )
```

Parameters

other

The EGridDecimator object that should be copied.

EGridDecimator::operator==

test equality between two EGridDecimator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator==(
 const EDecimator& other
 )
```

Parameters

other

the EGridDecimator to be compared with

4.114. EGs1Translator Class

Allows To retrieve human-readable GS1 code from machine-readable GS1 code.

Namespace: Euresys::Open_eVision

Methods

GetHumanReadableCode	Converts a machine-readable GS1 code to its human-readable counterpart.

EGs1Translator::GetHumanReadableCode

Converts a machine-readable GS1 code to its human-readable counterpart.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetHumanReadableCode(
 const std::string& machineReadableCode
 )
```

Parameters

machineReadableCode

The machine readable code returned by one of our code readers.

4.115. EHarrisCornerDetector Class

Manages a complete context for the Harris corner detector.

Remarks

This implementation of the Harris corner detector operates exclusively on a grayscale BW8 images.

Namespace: Euresys::Open_eVision

Methods

Apply	Apply the Harris corner detector on an image/ROI.
EHarrisCornerDetector	Constructs a EHarrisCornerDetector object initialized to its default values.
GetDerivationScale	Sets the derivation scale.
GetGradientNormalizationEnabled	Sets whether the gradient is normalized before the computation of the cornerness measure.
GetIntegrationScale	The integration scale.
GetSubpixelPrecisionEnabled	Sets whether the sub-pixel interpolation is enabled.
GetThreshold	Threshold on the cornerness measure for a pixel to be considered as a corner.
GetThresholdingMode	Thresholding mode for the cornerness measure.
SetDerivationScale	Sets the derivation scale.
SetGradientNormalizationEnabled	Sets whether the gradient is normalized before the computation of the cornerness measure.
SetIntegrationScale	The integration scale.
SetSubpixelPrecisionEnabled	Sets whether the sub-pixel interpolation is enabled.
SetThreshold	Threshold on the cornerness measure for a pixel to be considered as a corner.
SetThresholdingMode	Thresholding mode for the cornerness measure.

EHarrisCornerDetector::Apply

Apply the Harris corner detector on an image/ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void Apply(
 const EROIBW8& source,
 EHarrisInterestPoints& interestPoints
 )
```

Parameters

source

The source image/ROI.

interestPoints

The container in which to store the interest points.

EHarrisCornerDetector::GetDerivationScale

EHarrisCornerDetector::SetDerivationScale

Sets the derivation scale.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDerivationScale() const
void SetDerivationScale(float derivationScale)
```

Remarks

The derivation scale is the standard deviation of the Gaussian Filter used for the noise reduction during the computation of the gradient. Whenever the integration scale is set through EHarrisCornerDetector::IntegrationScale, the derivation scale is reset to its default value, 0.7 * integrationScale. This is a recommended value, as suggested by the literature.

EHarrisCornerDetector::EHarrisCornerDetector

Constructs a EHarrisCornerDetector object initialized to its default values.

Namespace: Euresys::Open_eVision

```
[C++]
void EHarrisCornerDetector(
 )
```

EHarrisCornerDetector::GetGradientNormalizationEnabled

EHarrisCornerDetector::SetGradientNormalizationEnabled

Sets whether the gradient is normalized before the computation of the cornerness measure.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetGradientNormalizationEnabled() const
void SetGradientNormalizationEnabled(bool isEnabled)
```

Remarks

If this flag is enabled, the values of the X-gradient and of the Y-gradient are first divided by their maximum absolute value (in the internal computations). This results in a cornerness measure that is roughly distributed around the value 1. If this flag is disabled, the cornerness measure will be much greater.

EHarrisCornerDetector::GetIntegrationScale

EHarrisCornerDetector::SetIntegrationScale

The integration scale.

Namespace: Euresys::Open_eVision

```
[C++]
float GetIntegrationScale() const
void SetIntegrationScale(float integrationScale)
```

Remarks

The *integration scale* is the standard deviation of the Gaussian filter that is used for scale analysis.

EHarrisCornerDetector::GetSubpixelPrecisionEnabled

EHarrisCornerDetector::SetSubpixelPrecisionEnabled

Sets whether the sub-pixel interpolation is enabled.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSubpixelPrecisionEnabled() const
void SetSubpixelPrecisionEnabled(bool subpixelEnabled)
```

Remarks

When this flag is enabled, a sub-pixel interpolation is carried on so as to improve the accuracy of the location of the corners, to the expense of a loss of speed.

EHarrisCornerDetector::GetThreshold

EHarrisCornerDetector::SetThreshold

Threshold on the cornerness measure for a pixel to be considered as a corner.

Namespace: Euresys::Open_eVision

```
[C++]
float GetThreshold() const
void SetThreshold(float threshold)
```

Remarks

If the threshold mode is set to EHarrisThresholdingMode_Absolute, the threshold value is interpreted as an absolute threshold on the cornerness. In this case, the threshold must be a strictly positive real value.

If the threshold mode is set to EHarrisThresholdingMode_Relative, the threshold is expressed as a fraction ranging from 0 to 1 of the maximum value of the cornerness of the source image.

EHarrisCornerDetector::GetThresholdingMode

EHarrisCornerDetector::SetThresholdingMode

Thresholding mode for the cornerness measure.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EHarrisThresholdingMode GetThresholdingMode() const
void SetThresholdingMode(Euresys::Open_eVision::EHarrisThresholdingMode mode)
```

4.116. EHarrisInterestPoints Class

Container class for the results of the Harris corner detector.

Remarks

The EHarrisCornerDetector class stores its results in this container.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the location of the corner points.
DrawCorner	Draws the location of a specific corner point.
DrawCornerWithCurrentPen	Draws the location of a specific corner point.
DrawWithCurrentPen	Draws the location of the corner points.
EHarrisInterestPoints	Constructs a container for the results of a Harris corner detector.
GetCornerness	Returns the cornerness measure of a corner point.
GetGradientMagnitude	Returns the magnitude of the gradient at a corner point.
GetGradientOrientation	Returns the orientation of the gradient at a corner point.
GetGradientX	Returns the gradient along the X-axis of a corner point.
GetGradientY	Returns the gradient along the Y-axis of a corner point.

GetPoint	Returns the coordinates of a corner point.
GetPointCount	The number of corner points in the container.
GetX	Returns the abscissa of a corner point.
GetY	Returns the ordinate of a corner point.

EHarrisInterestPoints::Draw

Draws the location of the corner points.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX

Horizontal panning value expressed in pixels. By default, no panning occurs.

originY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EHarrisInterestPoints::DrawCorner

Draws the location of a specific corner point.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawCorner(
 EDrawAdapter* graphicContext,
 int index,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void DrawCorner(
 HDC graphicContext,
 int index,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void DrawCorner(
 HDC graphicContext,
 const ERGBColor& color,
 int index,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Graphic context on which to draw.

index
Corner index

zoomX
Horizontal zooming factor. By default, true scale is used.

zoomY
Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX
Horizontal panning value expressed in pixels. By default, no panning occurs.

originY
Vertical panning value expressed in pixels. By default, no panning occurs.

color
The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EHarrisInterestPoints::DrawCornerWithCurrentPen

This method is deprecated.

Draws the location of a specific corner point.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawCornerWithCurrentPen(
 HDC graphicContext,
 int index,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Graphic context on which to draw.

index
Corner index

zoomX
Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX

Horizontal panning value expressed in pixels. By default, no panning occurs.

originY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EHarrisInterestPoints::DrawWithCurrentPen

This method is deprecated.

Draws the location of the corner points.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

originX

Horizontal panning value expressed in pixels. By default, no panning occurs.

originY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EHarrisInterestPoints::EHarrisInterestPoints

Constructs a container for the results of a Harris corner detector.

Namespace: Euresys::Open_eVision

```
[C++]
void EHarrisInterestPoints(
 )
```

EHarrisInterestPoints::GetCornerness

Returns the cornerness measure of a corner point.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCornerness(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetGradientMagnitude

Returns the magnitude of the gradient at a corner point.

Namespace: Euresys::Open_eVision

```
[C++]
float GetGradientMagnitude(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetGradientOrientation

Returns the orientation of the gradient at a corner point.

Namespace: Euresys::Open_eVision

```
[C++]
float GetGradientOrientation(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetGradientX

Returns the gradient along the X-axis of a corner point.

Namespace: Euresys::Open_eVision

```
[C++]
float GetGradientX(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetGradientY

Returns the gradient along the Y-axis of a corner point.

Namespace: Euresys::Open_eVision

```
[C++]
float GetGradientY(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetPoint

Returns the coordinates of a corner point.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetX

Returns the abscissa of a corner point.

Namespace: Euresys::Open_eVision

[C++]

```
float GetX(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetY

Returns the ordinate of a corner point.

Namespace: Euresys::Open_eVision

[C++]

```
float GetY(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the corner point.

EHarrisInterestPoints::GetPointCount

The number of corner points in the container.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetPointCount() const
```

4.117. EHDRColorFuser Class

A EHDRColorFuser instance is a tool that flexibly fuses color images using HDR principles.

Namespace: Euresys::Open_eVision

Methods

EHDRColorFuser	Constructors for EHDRColorFuser objects. The image used must be the lightest (slowest shutter speed).
Fuse	Fuses a dark image with the light image passed during object construction.
GetFusedImage	Retrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.
operator=	Assignment operator

EHDRColorFuser::EHDRColorFuser

Constructors for EHDRColorFuser objects. The image used must be the lightest (slowest shutter speed).

Namespace: Euresys::Open_eVision

[C++]

```
void EHDRColorFuser(
 const EROIC24* lightSrc
 )

void EHDRColorFuser(
 const EROIC48* lightSrc
 )

void EHDRColorFuser(
 const EHDRColorFuser& other
 )
```

Parameters

lightSrc

-

other

-

EHDRColorFuser::Fuse

Fuses a dark image with the light image passed during object construction.

Namespace: Euresys::Open_eVision

```
[C++]
void Fuse(
 const EROIC24* darkSrc,
 int shutterSpeedFactor
 )
void Fuse(
 const EROIC48* darkSrc,
 int shutterSpeedFactor
 )
```

Parameters

darkSrc

Dark input image (higher shutter speed).

shutterSpeedFactor

Shutter speed factor between light and dark image.

EHDRColorFuser::GetFusedImage

Retrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetFusedImage(
 EROIC24* dst
 )
bool GetFusedImage(
 EROIC48* dst
 )
```

Parameters

dst

Output image.

EHDRColorFuser::operator=

Assignment operator

Namespace: Euresys::Open_eVision

[C++]

```
EHDRColorFuser& operator=(
 const EHDRColorFuser& other
 )
```

Parameters

other

-

4.118. EHDRFuser Class

A EHDRFuser instance is a tool that flexibly fuses grayscale images using HDR principles.

Namespace: Euresys::Open_eVision

Methods

EHDRFuser	Constructors for EHDRFuser objects. The image used must be the lightest (slowest shutter speed).
Fuse	Fuses a dark image with the light image passed during object construction.
GetFusedImage	Retrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.
operator=	Assignment operator

EHDRFuser::EHDRFuser

Constructors for EHDRFuser objects. The image used must be the lightest (slowest shutter speed).

Namespace: Euresys::Open_eVision

[C++]

```
void EHDRFuser(
 const EROIBW8* lightSrc
 )

void EHDRFuser(
 const EROIBW16* lightSrc
 )

void EHDRFuser(
 const EHDRFuser& other
 )
```

Parameters

lightSrc

-

other

-

EHDRFuser::Fuse

Fuses a dark image with the light image passed during object construction.

Namespace: Euresys::Open_eVision

```
[C++]

void Fuse(
 const EROIBW8* darkSrc,
 int shutterSpeedFactor
 )

void Fuse(
 const EROIBW16* darkSrc,
 int shutterSpeedFactor
 )
```

Parameters

darkSrc

Dark input image (higher shutter speed).

shutterSpeedFactor

Shutter speed factor between light and dark image.

EHDRFuser::GetFusedImage

Retrieves the resulting image. Returns false if part of the potential image dynamic is lost due to the chosen output image type.

Namespace: Euresys::Open_eVision

```
[C++]

bool GetFusedImage(
 EROIBW8* dst
 )

bool GetFusedImage(
 EROIBW16* dst
 )

bool GetFusedImage(
 EROIBW32* dst
 )
```

Parameters

dst

Output image.

EHDRFuser::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
EHDRFuser& operator=(
 const EHDRFuser& other
 )
```

Parameters

other

-

4.119. EHitAndMissKernel Class

Class that defines a kernel for the morphological hit-and-miss operations.

Namespace: Euresys::Open_eVision

Methods

EHitAndMissKernel	Constructs a EHitAndMissKernel object.
GetEndX	Returns the abscissa of the rightmost element of the kernel.
GetEndY	Returns the abscissa of the bottommost element of the kernel.
GetStartX	Returns the abscissa of the leftmost element of the kernel.
GetStartY	Returns the ordinate of the toptmost element of the kernel.
GetValue	Returns the value of an element of the kernel at a given coordinate.
SetSize	Modify the size of the kernel.
SetValue	Sets the value of an element of the kernel at a given coordinate.

EHitAndMissKernel::EHitAndMissKernel

Constructs a EHitAndMissKernel object.

Namespace: Euresys::Open_eVision

[C++]

```
void EHitAndMissKernel(
 int startX,
 int startY,
 int endX,
 int endY
 )

void EHitAndMissKernel(
 OEV_UINT32 halfSizeX,
 OEV_UINT32 halfSizeY
 )

void EHitAndMissKernel(
 )
```

Parameters

startX

The abscissa of the top leftmost element of the kernel. This value must be less than or equal to zero.

startY

The ordinate of the top leftmost element of the kernel. This value must be less than or equal to zero.

endX

The abscissa of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

endY

The ordinate of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

halfSizeX

-

halfSizeY

-

Remarks

The constructor without argument creates a centered kernel of size 3x3.

All the elements of the kernel are initialized with EHitAndMissValue_DontCare values.

If the object is constructed by specifying the halves of the kernel dimensions, the width (resp. the height) of the kernel is given by "2 * halfSizeX + 1" (resp. "2 * halfSizeY + 1"). Otherwise, the width (resp. the height) of the kernel is given by "endX - startX + 1" (resp. "endY - startY + 1").

EHitAndMissKernel::GetEndX

Returns the abscissa of the rightmost element of the kernel.

Namespace: Euresys::Open_eVision

[C++]

```
int GetEndX() const
```

EHitAndMissKernel::GetEndY

Returns the abscissa of the bottommost element of the kernel.

Namespace: Euresys::Open_eVision

[C++]

```
int GetEndY() const
```

EHitAndMissKernel::GetValue

Returns the value of an element of the kernel at a given coordinate.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EHitAndMissValue GetValue(
 int x,
 int y
 )
```

Parameters

x

The abscissa of the element.

y

The ordinate of the element.

EHitAndMissKernel::SetSize

Modify the size of the kernel.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSize(
 int startX,
 int startY,
 int endX,
 int endY
 )
```

```
void SetSize(
 OEV_UINT32 halfSizeX,
 OEV_UINT32 halfSizeY
 )
```

Parameters

startX

The abscissa of the top leftmost element of the kernel. This value must be less than or equal to zero.

startY

The ordinate of the top leftmost element of the kernel. This value must be less than or equal to zero.

endX

The abscissa of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

endY

The ordinate of the bottom rightmost element of the kernel. This value must be greater than or equal to zero.

halfSizeX

The half of the kernel width minus 1. This value must be greater than zero.

halfSizeY

The half of the kernel height minus 1. This value must be greater than zero.

Remarks

All the elements of the kernel are initialized with EHitAndMissValue_DontCare values.

If the object is constructed by specifying the halves of the kernel dimensions, the width (resp. the height) of the kernel is given by "2 * halfSizeX + 1" (resp. "2 * halfSizeY + 1"). Otherwise, the width (resp. the height) of the kernel is given by "endX - startX + 1" (resp. "endY - startY + 1").

EHitAndMissKernel::SetValue

Sets the value of an element of the kernel at a given coordinate.

Namespace: Euresys::Open_eVision

[C++]

```
void SetValue(
 int x,
 int y,
 Euresys::Open_eVision::EHitAndMissValue value
 )
```

Parameters

x
The abscissa of the element.

y
The ordinate of the element.

value
The value of the element.

EHitAndMissKernel::GetStartX

Returns the abscissa of the leftmost element of the kernel.

Namespace: Euresys::Open_eVision

```
[C++]
int GetStartX() const
```

EHitAndMissKernel::GetStartY

Returns the ordinate of the toptmost element of the kernel.

Namespace: Euresys::Open_eVision

```
[C++]
int GetStartY() const
```

4.120. EHole Class

This class represents a hole inside an object (blob) of an encoded image.

Remarks

This class inherits from the ECodedElement class and provides an additional method to retrieve the parent object of a particular hole.

Base Class:ECodedElement

Namespace: Euresys::Open_eVision

Methods

EHole	-
GetParentObjectIndex	Returns the index of the parent object of the hole.
operator=	-

EHole::EHole

-

Namespace: Euresys::Open_eVision

```
[C++]
void EHole(
 const EHole& other
 )
```

Parameters

other

-

EHole::operator=

-

Namespace: Euresys::Open_eVision

```
[C++]
EHole& operator=(
 const EHole& other
 )
```

Parameters

other

-

EHole::GetParentObjectIndex

Returns the index of the parent object of the hole.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetParentObjectIndex() const
```

4.121. EImageBW1 Class

This class is deprecated.

The EImageBW1 class is used to represent rectangular regions of interest inside EBW1 black and white images. See ROIs.

Base Class:EROIBW1

Namespace: Euresys::Open_eVision

Methods

EImageBW1	Constructs a EImageBW1 image.
GetBitIndex	Gets the index of the bit at the given position in the image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageBW1 image.

EImageBW1::EImageBW1

This method is deprecated.

Constructs a EImageBW1 image.

Namespace: Euresys::Open_eVision

[C++]

```
void EImageBW1(
 )

void EImageBW1(
 int width,
 int height
 )

void EImageBW1(
 int width,
 int height,
 EBW1 constant
 )

void EImageBW1(
 const EImageBW1& other
 )
```

Parameters

width
The width, in pixels.

height
The height, in pixels.

constant
Binary constant.

other
Another EImageBW1 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageBW1 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageBW1::GetBitIndex

This method is deprecated.

Gets the index of the bit at the given position in the image.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT64 GetBitIndex(
 int x,
 int y
 )
```

Parameters

x

-

y

-

EImageBW1::InitializeFromUnalignedBuffer

This method is deprecated.

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

```
[C++]
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageBW1::operator=

This method is deprecated.

Copies a EImageBW1 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW1& operator=(
 const EImageBW1& other
 )
```

Parameters

other

Another EImageBW1 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.122. EImageBW16 Class

The EImageBW16 class is used to represent rectangular regions of interest inside EBW16 gray-level images. See ROIs.

Base Class:EROIBW16

Derived Class(es):EGrabberImageBW16

Namespace: Euresys::Open_eVision

Methods

EImageBW16	Constructs a EImageBW16 image.

InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageBW16 image.

EImageBW16::EImageBW16

Constructs a EImageBW16 image.

Namespace: Euresys::Open_eVision

```
[C++]
void EImageBW16(
 )
void EImageBW16(
 int width,
 int height
 )
void EImageBW16(
 int width,
 int height,
 EBW16 constant
 )
void EImageBW16(
 const EImageBW16& other
 )
```

Parameters

width
The width, in pixels.

height
The height, in pixels.

constant
Gray-level constant.

other
Another EImageBW16 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageBW16 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageBW16::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

```
[C++]
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageBW16::operator=

Copies a EImageBW16 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW16& operator=(
 const EImageBW16& other
 )
```

Parameters

other

Another EImageBW16 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.123. EImageBW32 Class

The EImageBW32 class is used to represent rectangular regions of interest inside EBW32 gray-level images. See ROIs.

Base Class:EROIBW32

Namespace: Euresys::Open_eVision

Methods

EImageBW32	Constructs a EImageBW32 image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageBW32 image.

EImageBW32::EImageBW32

Constructs a EImageBW32 image.

Namespace: Euresys::Open_eVision

[C++]

```
void EImageBW32(
 )

void EImageBW32(
 int width,
 int height
 )

void EImageBW32(
 int width,
 int height,
 EBW32 constant
 )

void EImageBW32(
 const EImageBW32& other
 )
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

constant

Gray-level constant.

other

Another EImageBW32 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageBW32 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageBW32::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

[C++]

```
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageBW32::operator=

Copies a EImageBW32 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW32& operator=(
 const EImageBW32& other
 )
```

Parameters

other

Another EImageBW32 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.124. EImageBW32f Class

The EImageBW32f class is used to represent rectangular regions of interest inside EBW32f gray-level images. See ROIs.

Base Class:EROIBW32f

Namespace: Euresys::Open_eVision

Methods

EImageBW32f	Constructs a EImageBW32f image.
InitializeFromUnalignedBuffer	Initializes image from an unaligned buffer.
operator=	Copies a EImageBW32f image.

EImageBW32f::EImageBW32f

Constructs a EImageBW32f image.

Namespace: Euresys::Open_eVision

```
[C++]
void EImageBW32f(
 )
```

```
void EImageBW32f(
 int width,
 int height
 )

void EImageBW32f(
 int width,
 int height,
 EBW32f constant
 )

void EImageBW32f(
 const EImageBW32f& other
 )
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

constant

Gray-level constant.

other

Another EImageBW32f object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageBW32f to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageBW32f::InitializeFromUnalignedBuffer

Initializes image from an unaligned buffer.

Namespace: Euresys::Open_eVision

[C++]

```
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageBW32f::operator=

Copies a EImageBW32f image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW32f& operator=(
 const EImageBW32f& other
 )
```

Parameters

other

Another EImageBW32f object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.125. EImageBW8 Class

The EImageBW8 class is used to represent rectangular regions of interest inside EBW8 gray-level images. See ROIs.

Base Class:EROIBW8

Derived Class(es):EGrabberImageBW8

Namespace: Euresys::Open_eVision

Methods

EImageBW8	Constructs a EImageBW8 image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageBW8 image.

EImageBW8::EImageBW8

Constructs a EImageBW8 image.

Namespace: Euresys::Open_eVision

[C++]

```
void EImageBW8(
 )

void EImageBW8(
 int width,
 int height
 )

void EImageBW8(
 int width,
 int height,
 EBW8 constant
 )

void EImageBW8(
 const EImageBW8& other
 )
```

Parameters

width
The width, in pixels.

height
The height, in pixels.

constant
Gray-level constant.

other
Another EImageBW8 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageBW8 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageBW8::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

```
[C++]
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageBW8::operator=

Copies a EImageBW8 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW8& operator=(
 const EImageBW8& other
 )
```

Parameters

other

Another EImageBW8 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.126. EImageC15 Class

The EImageC15 class is used to represent rectangular regions of interest inside EC15 color images. See ROIs.

Base Class:EROIC15

Namespace: Euresys::Open_eVision

Methods

EImageC15	Constructs a EImageC15 image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageC15 image.

EImageC15::EImageC15

Constructs a EImageC15 image.

Namespace: Euresys::Open_eVision

[C++]

```
void EImageC15(
 )

void EImageC15(
 int width,
 int height
 )

void EImageC15(
 int width,
 int height,
 EC15 constant
 )

void EImageC15(
 const EImageC15& other
 )
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

constant

Color constant.

other

Another EImageC15 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

EImageC15::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

```
[C++]
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageC15::operator=

Copies a EImageC15 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageC15& operator=(
 const EImageC15& other
 )
```

Parameters

other

Another EImageC15 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.127. EImageC16 Class

The EImageC16 class is used to represent rectangular regions of interest inside EC16 color images. See ROIs.

Base Class:EROIC16

Namespace: Euresys::Open_eVision

Methods

EImageC16	Constructs a EImageC16 image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageC16 image.

EImageC16::EImageC16

Constructs a EImageC16 image.

Namespace: Euresys::Open_eVision

```
[C++]
void EImageC16(
 )
```

```
void EImageC16(
 int width,
 int height
 )

void EImageC16(
 int width,
 int height,
 EC16 constant
 )

void EImageC16(
 const EImageC16& other
 )
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

constant

Color constant.

other

Another EImageC16 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageC16 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageC16::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

[C++]

```
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageC16::operator=

Copies a EImageC16 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageC16& operator=(
 const EImageC16& other
 )
```

Parameters

other

Another EImageC16 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.128. EImageC24 Class

The EImageC24 class is used to represent rectangular regions of interest inside EC24 color images. See ROIs.

Base Class:EROIC24

Derived Class(es):EGrabberImageC24

Namespace: Euresys::Open_eVision

Methods

EImageC24	Constructs a EImageC24 image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageC24 image.

EImageC24::EImageC24

Constructs a EImageC24 image.

Namespace: Euresys::Open_eVision

```
[C++]
void EImageC24(
 )
void EImageC24(
 int width,
 int height
 )
void EImageC24(
 int width,
 int height,
 EC24 constant
 )
void EImageC24(
 const EImageC24& other
 )
```

Parameters

width
The width, in pixels.

height
The height, in pixels.

constant
Color constant.

other
Another EImageC24 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageC24 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageC24::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

```
[C++]
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageC24::operator=

Copies a EImageC24 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageC24& operator=(
 const EImageC24& other
 )
```

Parameters

other

Another EImageC24 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.129. EImageC24A Class

The EImageC24A class is used to represent rectangular regions of interest inside EC24A color images. See ROIs.

Base Class:EROIC24A

Namespace: Euresys::Open_eVision

Methods

EImageC24A	Constructs a EImageC24A image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageC24A image.

EImageC24A::EImageC24A

Constructs a EImageC24A image.

Namespace: Euresys::Open_eVision

[C++]

```
void EImageC24A(
 )

void EImageC24A(
 int width,
 int height
 )

void EImageC24A(
 int width,
 int height,
 EC24A constant
 )

void EImageC24A(
 const EImageC24A& other
 )
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

constant

Color constant.

other

Another EImageC24A object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageC24A to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageC24A::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

[C++]

```
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageC24A::operator=

Copies a EImageC24A image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageC24A& operator=(
 const EImageC24A& other
 )
```

Parameters

other

Another EImageC24A object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.130. EImageC48 Class

The EImageC48 class is used to represent rectangular regions of interest inside EC48 color images. See ROIs.

Base Class:EROIC48

Namespace: Euresys::Open_eVision

Methods

EImageC48	Constructs a EImageC48 image.
InitializeFromUnalignedBuffer	Initialize image from an unaligned buffer.
operator=	Copies a EImageC48 image.

EImageC48::EImageC48

Constructs a EImageC48 image.

Namespace: Euresys::Open_eVision

```
[C++]
void EImageC48(
 )
```

```
void EImageC48(
 int width,
 int height
 )

void EImageC48(
 int width,
 int height,
 EC48 constant
 )

void EImageC48(
 const EImageC48& other
 )
```

Parameters

width

The width, in pixels.

height

The height, in pixels.

constant

Color constant.

other

Another EImageC48 object.

Remarks

The constructor with no arguments creates a zero-sized image. You can modify the image size by calling EBaseROI::SetSize method. The sizing constructor constructs an image of the given size. See EBaseROI::SetSize for informations about the sizing restrictions. The copy constructor copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object. Note that if you used EImageC48 to assign an external buffer to the image, that buffer will not be copied, the resulting image will just point to the same buffer as the original.

EImageC48::InitializeFromUnalignedBuffer

Initialize image from an unaligned buffer.

Namespace: Euresys::Open_eVision

[C++]

```
void InitializeFromUnalignedBuffer(
 int width,
 int height,
 void* buffer,
 int pitch
 )
```

Parameters

width

Width of the image in the buffer.

height

Height of the image in the buffer.

buffer

Address of the buffer.

pitch

Pitch of the buffer (in bytes). If omitted, the method assumes there is no padding (the pitch is assumed be equal (width * sizeOfAPixel))

Remarks

The image will be initialized to fit the width and height passed as arguments. The contents of the buffer will then be copied into the image. The buffer contents must be compatible with the pixel type of the image.

EImageC48::operator=

Copies a EImageC48 image.

Namespace: Euresys::Open_eVision

```
[C++]
EImageC48& operator=(
 const EImageC48& other
 )
```

Parameters

other

Another EImageC48 object.

Remarks

This method copies all the supplied image properties (content, attributes, sub-ROIs...) into the current object.

4.131. EImageEncoder Class

This class is responsible for the encoding of an image into an ECodedImage2 object.

Remarks

This class is responsible for the extraction of the runs in a source image, the aggregation of the runs into objects, as well as the proper handling of the continuous mode. It also provides methods to configure the image segmentation process.

The segmentation process classifies the pixels of the source image according to their value to create a set of layers. In each layer taken separately, the encoding process then assembles the connected pixels to build the coded elements (blobs).

By default, the segmentation method consists of grayscale single thresholding, with automatic threshold selection (as determined by the minimum residue rule).

Namespace: Euresys::Open_eVision

Methods

CopyTo	-
EImageEncoder	Constructs an image encoder.
Encode	Encodes an image or an ROI as a coded image.
FlushContinuousMode	Resets the continuous mode, emptying the internal memory, after writing the objects that are not yet completed in a coded image.
GetBinaryImageSegmenter	Returns a reference to the internal instance of the the segmenter for binary images, in order to set its parameters.
GetColorRangeThresholdSegmenter	Returns a reference to the internal instance of the the segmenter for color-range thresholding, in order to set its parameters.
GetColorSingleThresholdSegmenter	Returns a reference to the internal instance of the the segmenter for single color thresholding, in order to set its parameters.
GetContinuousModeEnabled	Continuous mode enabling status.
GetContinuousModeMaxHeight	Maximum number of rows that are kept in memory in the continuous mode.
GetEncodingConnexity	Connexity mode.
GetGrayscaleDoubleThresholdSegmenter	Returns a reference to the internal instance of the the segmenter for double color thresholding, in order to set its parameters.
GetGrayscaleSingleThresholdSegmenter	Returns a reference to the internal instance of the the segmenter for single grayscale thresholding, in order to set its parameters.
GetImageRangeSegmenter	Returns a reference to the internal instance of the the segmenter for pixel-by-pixel, range-based thresholding, in order to set its parameters.
GetLabeledImageSegmenter	Returns a reference to the internal instance of the the segmenter thats maps the value of the pixels directly to a layer index, in order to set its parameters.
GetReferenceImageSegmenter	Returns a reference to the internal instance of the the segmenter for single pixel-by-pixel thresholding, in order to set its parameters.
GetSegmentationMethod	Segmentation method used during the encoding.

Load	Loads the EImageEncoder. The given ESerializer must have been created for reading.
operator!=	Inequality operator.
operator=	-
operator==	Equality operator.
ResetContinuousMode	Resets the continuous mode, emptying the internal memory.
Save	Saves the EImageEncoder. The given ESerializer must have been created for writing.
Serialize	Serializes the EImageEncoder.
SetContinuousModeEnabled	Continuous mode enabling status.
SetContinuousModeMaxHeight	Maximum number of rows that are kept in memory in the continuous mode.
SetEncodingConnexity	Connexity mode.
SetSegmentationMethod	Segmentation method used during the encoding.

EImageEncoder::GetBinaryImageSegmenter

Returns a reference to the internal instance of the the segmenter for binary images, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EBinaryImageSegmenter& GetBinaryImageSegmenter()
```

EImageEncoder::GetColorRangeThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for color-range thresholding, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EColorRangeThresholdSegmenter& GetColorRangeThresholdSegmenter()
```

EImageEncoder::GetColorSingleThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for single color thresholding, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EColorSingleThresholdSegmenter& GetColorSingleThresholdSegmenter()
```

EImageEncoder::GetContinuousModeEnabled

EImageEncoder::SetContinuousModeEnabled

Continuous mode enabling status.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetContinuousModeEnabled() const
void SetContinuousModeEnabled(bool enabled)
```

Remarks

In the continuous mode, objects are constructed over a sequence of images: The image encoder encodes only the objects that contain no run touching the last row of the source image. The objects touching the inferior border of the image are not written in the coded image: These objects are indeed expected to continue in the subsequent image chunks. Such objects are kept in memory, and are consumed when analyzing the subsequent images.

EImageEncoder::GetContinuousModeMaxHeight

EImageEncoder::SetContinuousModeMaxHeight

Maximum number of rows that are kept in memory in the continuous mode.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetContinuousModeMaxHeight() const
void SetContinuousModeMaxHeight(OEV_UINT32 maxHeight)
```

Remarks

This property can be used to put a bound on the size of the internal memory of the image encoder in the continuous mode. If this property is set to zero, then memory can grow arbitrarily (there is no maximum number of rows).

EImageEncoder::CopyTo

-

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EImageEncoder& other
 )
```

Parameters

other

-

EImageEncoder::EImageEncoder

Constructs an image encoder.

Namespace: Euresys::Open_eVision

[C++]

```
void EImageEncoder(
 )
void EImageEncoder(
 const EImageEncoder& other
 )
```

Parameters

other

-

EImageEncoder::Encode

This method is deprecated.

Encodes an image or an ROI as a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void Encode(
 const EROIBW1& sourceImage,
 ECodedImage2& codedImage
 )
void Encode(
 const EROIBW1& sourceImage,
 const EROIBW8& inputMask,
 ECodedImage2& codedImage
 )
```

```
void Encode(
 const EROIBW1& sourceImage,
 const ERegion& region,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIBW8& sourceImage,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIBW16& sourceImage,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIC24& sourceImage,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIBW8& sourceImage,
 const EROIBW8& inputMask,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIBW16& sourceImage,
 const EROIBW8& inputMask,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIC24& sourceImage,
 const EROIBW8& inputMask,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIBW8& sourceImage,
 const ERegion& region,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIBW16& sourceImage,
 const ERegion& region,
 ECodedImage2& codedImage
 )

void Encode(
 const EROIC24& sourceImage,
 const ERegion& region,
 ECodedImage2& codedImage
 )
```

Parameters

sourceImage

The input image that is to be encoded.

codedImage

The coded image that will hold the result of the encoding process.

inputMask

The possible input Flexible Mask that restricts the encoding. The input mask is a grayscale image having the same height and the same width as the source image. Any pixel in the source image that is covered by a value of 0 in the input mask will not get encoded in any layer. Any other pixel value in the input mask causes the pixel to be a candidate for the encoding.

region

Region of the input image that is to be encoded.

Remarks

The previous content of the result coded image is discarded.

EImageEncoder::GetEncodingConnexity

EImageEncoder::SetEncodingConnexity

Connexity mode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EEncodingConnexity GetEncodingConnexity() const
void SetEncodingConnexity(Euresys::Open_eVision::EEncodingConnexity connexity)
```

Remarks

The connexity mode specifies the conditions that must hold for neighboring pixels to belong to the same object.

EImageEncoder::FlushContinuousMode

Resets the continuous mode, emptying the internal memory, after writing the objects that are not yet completed in a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void FlushContinuousMode(
 ECodedImage2& codedImage
 )
```

Parameters

codedImage

The coded image in which the not-yet-completed objects are written.

EImageEncoder::GetGrayscaleDoubleThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for double color thresholding, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EGrayscaleDoubleThresholdSegmenter& GetGrayscaleDoubleThresholdSegmenter()
```

EImageEncoder::GetGrayscaleSingleThresholdSegmenter

Returns a reference to the internal instance of the the segmenter for single grayscale thresholding, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EGrayscaleSingleThresholdSegmenter& GetGrayscaleSingleThresholdSegmenter()
```

EImageEncoder::GetImageRangeSegmenter

Returns a reference to the internal instance of the the segmenter for pixel-by-pixel, range-based thresholding, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EImageRangeSegmenter& GetImageRangeSegmenter()
```

EImageEncoder::GetLabeledImageSegmenter

Returns a reference to the internal instance of the the segmenter thats maps the value of the pixels directly to a layer index, in order to set its parameters.

Namespace: Euresys::Open_eVision

[C++]

```
ELabeledImageSegmenter& GetLabeledImageSegmenter()
```

EImageEncoder::Load

Loads the EImageEncoder. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EImageEncoder::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EImageEncoder& other
 )
```

Parameters

other
Other image encoder to compare with

EImageEncoder::operator=

-

Namespace: Euresys::Open_eVision

[C++]

```
EImageEncoder& operator=(
 const EImageEncoder& other
 )
```

Parameters

other

-

EImageEncoder::operator==

Equality operator.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EImageEncoder& other
 )
```

Parameters

other

Other image encoder to compare with

EImageEncoder::GetReferenceImageSegmenter

Returns a reference to the internal instance of the the segmenter for single pixel-by-pixel thresholding, in order to set its parameters.

Namespace: Euresys::Open_eVision

```
[C++]
EReferenceImageSegmenter& GetReferenceImageSegmenter()
```

EImageEncoder::ResetContinuousMode

Resets the continuous mode, emptying the internal memory.

Namespace: Euresys::Open_eVision

```
[C++]
void ResetContinuousMode(
 )
```

EImageEncoder::Save

Saves the EImageEncoder. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EImageEncoder::GetSegmentationMethod

EImageEncoder::SetSegmentationMethod

Segmentation method used during the encoding.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ESegmentationMethod GetSegmentationMethod() const
void SetSegmentationMethod(Euresys::Open_eVision::ESegmentationMethod method)
```

EImageEncoder::Serialize

Serializes the EImageEncoder.

Namespace: Euresys::Open_eVision

[C++]

```
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer

Serializer

4.132. EImageRangeSegmenter Class

Segments an image using a pixel-by-pixel double threshold given as two images.

Remarks

This segmenter is applicable to EROIBW8, EROIBW16 and EROIC24 images. It produces coded images with two layers. The low threshold and the high threshold are defined for each pixel individually by means of two reference images of the same type as the source image: the Low Image and the High Image.

For grayscales images, the White layer (usually, with index 1) contains unmasked pixels having a gray value in a range defined by the gray value of the corresponding unmasked pixels in the Low Image and the High Image.

For RGB color images, the White layer (usually, with index 1) contains unmasked pixels having a color inside the cube of the color space defined by the colors of the corresponding unmasked pixels in the Low Image and the High Image.

The Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class:ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetHighImageBW16	High image for the segmentation of EROIBW16 images.
GetHighImageBW8	High image for the segmentation of EROIBW8 images.
GetHighImageC24	High image for the segmentation of EROIC24 images.
GetLowImageBW16	Low image for the segmentation of EROIBW16 images.
GetLowImageBW8	Low image for the segmentation of EROIBW8 images.
GetLowImageC24	Low image for the segmentation of EROIC24 images.
Load	Load the EImageRangeSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EImageRangeSegmenter configuration. The given ESerializer must have been created for writing.
SetBlackLayerEncoded	Black layer encoding status.
SetBlackLayerIndex	Index of the black layer in the destination coded image.
SetHighImageBW16	High image for the segmentation of EROIBW16 images.
SetHighImageBW8	High image for the segmentation of EROIBW8 images.
SetHighImageC24	High image for the segmentation of EROIC24 images.
SetLowImageBW16	Low image for the segmentation of EROIBW16 images.
SetLowImageBW8	Low image for the segmentation of EROIBW8 images.

SetLowImageC24	Low image for the segmentation of EROIC24 images.
SetWhiteLayerEncoded	White layer encoding status.
SetWhiteLayerIndex	Index of the white layer in the destination coded image.

EImageRangeSegmenter::SetBlackLayerEncoded

Black layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void SetBlackLayerEncoded(bool encode)
```

EImageRangeSegmenter::SetBlackLayerIndex

Index of the black layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void SetBlackLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the black layer.

EImageRangeSegmenter::GetHighImageBW16

EImageRangeSegmenter::SetHighImageBW16

High image for the segmentation of EROIBW16 images.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
const EROIBW16* GetHighImageBW16() const
void SetHighImageBW16(const EROIBW16* image)
```

EImageRangeSegmenter::GetHighImageBW8

EImageRangeSegmenter::SetHighImageBW8

High image for the segmentation of EROIBW8 images.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
const EROIBW8* GetHighImageBW8() const
void SetHighImageBW8(const EROIBW8* image)
```

EImageRangeSegmenter::GetHighImageC24

EImageRangeSegmenter::SetHighImageC24

High image for the segmentation of EROIC24 images.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
const EROIC24* GetHighImageC24() const
void SetHighImageC24(const EROIC24* image)
```

EImageRangeSegmenter::Load

Load the EImageRangeSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EImageRangeSegmenter::GetLowImageBW16

EImageRangeSegmenter::SetLowImageBW16

Low image for the segmentation of EROIBW16 images.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
const EROIBW16* GetLowImageBW16() const
void SetLowImageBW16(const EROIBW16* image)
```

EImageRangeSegmenter::GetLowImageBW8

EImageRangeSegmenter::SetLowImageBW8

Low image for the segmentation of EROIBW8 images.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
const EROIBW8* GetLowImageBW8() const
void SetLowImageBW8(const EROIBW8* image)
```

EImageRangeSegmenter::GetLowImageC24

EImageRangeSegmenter::SetLowImageC24

Low image for the segmentation of EROIC24 images.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
const EROIC24* GetLowImageC24() const
void SetLowImageC24(const EROIC24* image)
```

EImageRangeSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool operator==(
 const EImageRangeSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to

EImageRangeSegmenter::Save

Save the EImageRangeSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EImageRangeSegmenter::SetWhiteLayerEncoded

White layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void SetWhiteLayerEncoded(bool encode)
```

EImageRangeSegmenter::SetWhiteLayerIndex

Index of the white layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void SetWhiteLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the white layer.

4.133. EImageSegmenter Class

Base class from which all the segmenters derive.

Derived Class (es): ELabeledImageSegmenter EThreeLayersImageSegmenter ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

4.134. EIntegerRange Class

Represents a range of integer values.

Namespace: Euresys::Open_eVision

Methods

EIntegerRange	Creates an EIntegerRange object.
GetCenter	Center of the range.
GetLowerBound	Lower bound of the range.
GetSize	Size of the range.
GetUpperBound	Upper bound of the range.
IsInRange	Checks if a value is inside the range.
operator=	Assignment operator
SetBounds	Sets the bounds of the range.
SetFromBaseAndAbsoluteTolerance	Sets the bounds of the range from a base value and an absolute tolerance from this base value.
SetFromBaseAndRelativeTolerance	Sets the bounds of the range from a base value and a relative tolerance from this base value.

Update	Updates the range. If the value passed is outside of the range bounds, they are modified so they include the value.

EIntegerRange::GetCenter

Center of the range.

Namespace: Euresys::Open_eVision

```
[C++]
int GetCenter() const
```

EIntegerRange::EIntegerRange

Creates an EIntegerRange object.

Namespace: Euresys::Open_eVision

```
[C++]
void EIntegerRange(
 )
void EIntegerRange(
 int min,
 int max
 )
void EIntegerRange(
 const EIntegerRange& range
 )
```

Parameters

min
Lower bound of the range.

max
Upper bound of the range.

range
Range to copy.

EIntegerRange::IsInRange

Checks if a value is inside the range.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsInRange(
 int value,
 bool lowerBoundInclusive,
 bool upperBoundInclusive
 )

bool IsInRange(
 float value,
 bool lowerBoundInclusive,
 bool upperBoundInclusive
 )
```

Parameters

value

Value to test.

lowerBoundInclusive

Indicates if the lower bound should be considered inside the range (true) or outside (false).

upperBoundInclusive

Indicates if the upper bound should be considered inside the range (true) or outside (false).

EIntegerRange::GetLowerBound

Lower bound of the range.

Namespace: Euresys::Open_eVision

[C++]

```
int GetLowerBound() const
```

EIntegerRange::operator=

Assignment operator

Namespace: Euresys::Open_eVision

[C++]

```
EIntegerRange& operator=(
 const EIntegerRange& other
 )
```

Parameters

other

-

EIntegerRange::SetBounds

Sets the bounds of the range.

Namespace: Euresys::Open_eVision

```
[C++]
void SetBounds(
 int min,
 int max
 )
```

Parameters

min

Lower bound of the range.

max

Upper bound of the range.

EIntegerRange::SetFromBaseAndAbsoluteTolerance

Sets the bounds of the range from a base value and an absolute tolerance from this base value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromBaseAndAbsoluteTolerance(
 int baseValue,
 int tolerance
 )
```

Parameters

baseValue

Base value.

tolerance

Absolute tolerance around the base value. Must be positive.

Remarks

The range will be set with a lower bound of (base - tolerance) and an upper bound of (base + tolerance).

EIntegerRange::SetFromBaseAndRelativeTolerance

Sets the bounds of the range from a base value and a relative tolerance from this base value.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromBaseAndRelativeTolerance(
 int baseValue,
 float tolerance
 )
```

Parameters

baseValue

Base value.

tolerance

Relative tolerance around the base value. Must be positive.

Remarks

The range will be set with a lower bound of (base - (base * tolerance)) and an upper bound of (base + (base * tolerance)).

EIntegerRange::GetSize

Size of the range.

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize() const
```

EIntegerRange::Update

Updates the range. If the value passed is outside of the range bounds, they are modified so they include the value.

Namespace: Euresys::Open_eVision

[C++]

```
void Update(
 int value
 )
```

Parameters

value

Value to be included in the range.

EIntegerRange::GetUpperBound

Upper bound of the range.

Namespace: Euresys::Open_eVision

[C++]

```
int GetUpperBound() const
```

4.135. EInterestPointLocator Class

EasyLocate tool - interest point version.

The EInterestPointLocator locates and classifies objects (or defects). With this class, an object is defined by its position it and a label (see ELocatorObject class). The tool must be trained using a dataset with annotated objects.
The minimum size resolution supported by the tool is 128. The maximum size of the image supported is 500 000 pixels.
The tool has 5 main parameters:
- The object size (EInterestPointLocator::ObjectSize) to indicate an approximate size for the interest point that you want to detect.
- A detection threshold (ELocatorBase::DetectionThreshold) to accept or reject predicted objects based on their score.
- The maximum number of objects in an image (ELocatorBase::MaxNumberOfObjects)
- The maximum proximity or the minimum distance between predicted objects with the same label (EInterestPointLocator::SameLabelMinDistance, ELocatorBase::SameLabelMaxObjectProximity).
- The maximum proximity or the minimum distance between predicted objects regardless of their label (EInterestPointLocator::AbsoluteMinDistance, ELocatorBase::AbsoluteMaxObjectProximity).
The proximity is defined as max(0, EInterestPointLocator::ObjectSize - d) / (EInterestPointLocator::ObjectSize + d) where d is the manhattan distance.
Except for the EInterestPointLocator::ObjectSize, all the parameters can be changed before and after training the tool.

Base Class:ELocatorBase

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

EInterestPointLocator	Constructs a EInterestPointLocator object.
GetAbsoluteMinDistance	Minimum distance between two predicted objects regardless of their label. This value must be between 0 and EInterestPointLocator::ObjectSize. It is internally converted to ELocatorBase::AbsoluteMaxObjectProximity. Default value: 0.
GetObjectSize	Specify the size of predicted objects when only the position is predicted. This parameter is only used to prepare the predicted objects for drawing.
GetSameLabelMinDistance	Minimum distance between two predicted objects with the same label. This value must be between 0 and EInterestPointLocator::ObjectSize. It is internally converted to ELocatorBase::SameLabelMaxObjectProximity. Default value: EInterestPointLocator::ObjectSize / 3.

GetToolType	Type of the deep learning tool.
operator=	Assignment operator
SerializeSettings	Serializes the settings of the locator.
SetAbsoluteMinDistance	Minimum distance between two predicted objects regardless of their label. This value must be between 0 and EInterestPointLocator::ObjectSize. It is internally converted to ELocatorBase::AbsoluteMaxObjectProximity. Default value: 0.
SetObjectSize	Specify the size of predicted objects when only the position is predicted. This parameter is only used to prepare the predicted objects for drawing.
SetSameLabelMinDistance	Minimum distance between two predicted objects with the same label. This value must be between 0 and EInterestPointLocator::ObjectSize. It is internally converted to ELocatorBase::SameLabelMaxObjectProximity. Default value: EInterestPointLocator::ObjectSize / 3.

EInterestPointLocator::GetAbsoluteMinDistance

EInterestPointLocator::SetAbsoluteMinDistance

Minimum distance between two predicted objects regardless of their label.
This value must be between 0 and EInterestPointLocator::ObjectSize. It is internally converted to ELocatorBase::AbsoluteMaxObjectProximity. Default value: 0.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAbsoluteMinDistance() const

void SetAbsoluteMinDistance(float val)
```

EInterestPointLocator::EInterestPointLocator

Constructs a EInterestPointLocator object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EInterestPointLocator(
 )

void EInterestPointLocator(
 int objectSize
 )
```

```
void EInterestPointLocator(
 const EInterestPointLocator& other
 )
```

Parameters

objectSize

Size of the interest point

other

Reference to the EInterestPointLocator object that should be copied

EInterestPointLocator::GetObjectSize

EInterestPointLocator::SetObjectSize

Specify the size of predicted objects when only the position is predicted. This parameter is only used to prepare the predicted objects for drawing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetObjectSize() const
void SetObjectSize(int objectSize)
```

EInterestPointLocator::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EInterestPointLocator& operator=(
 const EInterestPointLocator& other
 )
```

Parameters

other

Reference to the EInterestPointLocator object that should be copied

EInterestPointLocator::GetSameLabelMinDistance

EInterestPointLocator::SetSameLabelMinDistance

Minimum distance between two predicted objects with the same label.
This value must be between 0 and EInterestPointLocator::ObjectSize. It is internally converted to ELocatorBase::SameLabelMaxObjectProximity. Default value: EInterestPointLocator::ObjectSize / 3.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSameLabelMinDistance() const

void SetSameLabelMinDistance(float val)
```

EInterestPointLocator::SerializeSettings

Serializes the settings of the locator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer
Pointer to ESerializer

EInterestPointLocator::GetToolType

Type of the deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetToolType() const
```

4.136. EKernel Class

Kernel for use in convolution operations.

Namespace: Euresys::Open_eVision

Methods

EKernel	Constructs an EKernel object.
GetGain	Global gain.
GetKernelData	Returns the convolution coefficient of given indices.
GetOffset	Global offset (a constant added to the convolution result).
GetOutsideValue	Out-of-limits image value (only influences the result along image edges).
GetRawDataPtr	Pointer to the upper left convolution coefficient.
GetRectifier	Rectification mode. This property allows specifying how negative convolution result values are handled.
GetSizeX	Number of coefficients along a row.
GetSizeY	Number of coefficients along a column.
SetGain	Global gain.
SetKernelData	Sets the convolution coefficient values at the given indices.
SetOffset	Global offset (a constant added to the convolution result).
SetOutsideValue	Out-of-limits image value (only influences the result along image edges).
SetRectifier	Rectification mode. This property allows specifying how negative convolution result values are handled.
SetSize	Sets the number of coefficients along a row and along a column.

EKernel::EKernel

Constructs an EKernel object.

Namespace: Euresys::Open_eVision

[C++]

```
void EKernel(
 )

void EKernel(
 short sizeX,
 short sizeY,
 float gain,
 OEV_UINT32 offset,
 Euresys::Open_eVision::EKernelRectifier rectifier,
 OEV_UINT32 outsideValue
 )
```

```
void EKernel(
 Euresys::Open_eVision::EKernelType KernelType
 )
```

Parameters

sizeX
Number of coefficients along a row.

sizeY
Number of coefficients along a column.

gain
Global gain.

offset
Global offset.

rectifier
Rectification mode, as defined by EKernelRectifier.

outsideValue
Out-of-limits image value.

KernelType
Kernel type, as defined by EKernelType.

Remarks

The default constructor constructs a void kernel. A void kernel has no associated convolution coefficients. The sizing constructor constructs a kernel of given size and global parameters. The third constructor constructs a kernel of a predefined type.

EKernel::GetGain

EKernel::SetGain

Global gain.

Namespace: Euresys::Open_eVision

[C++]

```
float GetGain()

void SetGain(float f32Gain)
```

Remarks

Before the global gain is applied, the coefficients are normalized so that their sum equals one, unless their sum equals zero (as is the case for a derivation operator). The rectification enables to handle the negative values that may appear after convolution.

EKernel::GetKernelData

Returns the convolution coefficient of given indices.

Namespace: Euresys::Open_eVision

```
[C++]
void GetKernelData(
 int columnIndex,
 int rowIndex,
 float& coefficientValue
 )
```

Parameters

columnIndex

Column index, from 0 on, increasing rightwards.

rowIndex

Row index, from 0 on, increasing downwards.

coefficientValue

Reference to the coefficient value.

EKernel::GetOffset

EKernel::SetOffset

Global offset (a constant added to the convolution result).

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetOffset()
void SetOffset(OEV_UINT32 un32Offset)
```

EKernel::GetOutsideValue

EKernel::SetOutsideValue

Out-of-limits image value (only influences the result along image edges).

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetOutsideValue()
```

```
void SetOutsideValue(OEV_UINT32 un32OutsideValue)
```

EKernel::GetRawDataPtr

Pointer to the upper left convolution coefficient.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr()
```

Remarks

This pointer is actually the base address of a float array containing all coefficients.

EKernel::GetRectifier

EKernel::SetRectifier

Rectification mode. This property allows specifying how negative convolution result values are handled.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EKernelRectifier GetRectifier()
void SetRectifier(Euresys::Open_eVision::EKernelRectifier rectifier)
```

EKernel::SetKernelData

Sets the convolution coefficient values at the given indices.

Namespace: Euresys::Open_eVision

[C++]

```
void SetKernelData(
 int columnIndex,
 int rowIndex,
 float coefficientValue
 )
```

```
void SetKernelData(
 float coefficientValue00,
 float coefficientValue10,
 float coefficientValue20,
 float coefficientValue01,
 float coefficientValue11,
 float coefficientValue21,
 float coefficientValue02,
 float coefficientValue12,
 float coefficientValue22
 )
void SetKernelData(
 float coefficientValue00,
 float coefficientValue10,
 float coefficientValue20,
 float coefficientValue30,
 float coefficientValue40,
 float coefficientValue01,
 float coefficientValue11,
 float coefficientValue21,
 float coefficientValue31,
 float coefficientValue41,
 float coefficientValue02,
 float coefficientValue12,
 float coefficientValue22,
 float coefficientValue32,
 float coefficientValue42,
 float coefficientValue03,
 float coefficientValue13,
 float coefficientValue23,
 float coefficientValue33,
 float coefficientValue43,
 float coefficientValue04,
 float coefficientValue14,
 float coefficientValue24,
 float coefficientValue34,
 float coefficientValue44
 )
```

```
void SetKernelData(
 float coefficientValue00,
 float coefficientValue10,
 float coefficientValue20,
 float coefficientValue30,
 float coefficientValue40,
 float coefficientValue50,
 float coefficientValue60,
 float coefficientValue01,
 float coefficientValue11,
 float coefficientValue21,
 float coefficientValue31,
 float coefficientValue41,
 float coefficientValue51,
 float coefficientValue61,
 float coefficientValue02,
 float coefficientValue12,
 float coefficientValue22,
 float coefficientValue32,
 float coefficientValue42,
 float coefficientValue52,
 float coefficientValue62,
 float coefficientValue03,
 float coefficientValue13,
 float coefficientValue23,
 float coefficientValue33,
 float coefficientValue43,
 float coefficientValue53,
 float coefficientValue63,
 float coefficientValue04,
 float coefficientValue14,
 float coefficientValue24,
 float coefficientValue34,
 float coefficientValue44,
 float coefficientValue54,
 float coefficientValue64,
 float coefficientValue05,
 float coefficientValue15,
 float coefficientValue25,
 float coefficientValue35,
 float coefficientValue45,
 float coefficientValue55,
 float coefficientValue65,
 float coefficientValue06,
 float coefficientValue16,
 float coefficientValue26,
 float coefficientValue36,
 float coefficientValue46,
 float coefficientValue56,
 float coefficientValue66
 )
```

Parameters

columnIndex
Column index, from 0 on, increasing rightwards.

rowIndex
Row index, from 0 on, increasing downwards.

coefficientValue
New coefficientValue.

coefficientValue00
Coefficient value at corresponding column and row indices.

coefficientValue10
Coefficient value at corresponding column and row indices.

coefficientValue20
Coefficient value at corresponding column and row indices.

coefficientValue01
Coefficient value at corresponding column and row indices.

coefficientValue11
Coefficient value at corresponding column and row indices.

coefficientValue21
Coefficient value at corresponding column and row indices.

coefficientValue02
Coefficient value at corresponding column and row indices.

coefficientValue12
Coefficient value at corresponding column and row indices.

coefficientValue22
Coefficient value at corresponding column and row indices.

coefficientValue30
Coefficient value at corresponding column and row indices.

coefficientValue40
Coefficient value at corresponding column and row indices.

coefficientValue31
Coefficient value at corresponding column and row indices.

coefficientValue41
Coefficient value at corresponding column and row indices.

coefficientValue32
Coefficient value at corresponding column and row indices.

coefficientValue42
Coefficient value at corresponding column and row indices.

coefficientValue03
Coefficient value at corresponding column and row indices.

coefficientValue13
Coefficient value at corresponding column and row indices.

coefficientValue23
Coefficient value at corresponding column and row indices.

coefficientValue33
Coefficient value at corresponding column and row indices.
coefficientValue43
Coefficient value at corresponding column and row indices.
coefficientValue04
Coefficient value at corresponding column and row indices.
coefficientValue14
Coefficient value at corresponding column and row indices.
coefficientValue24
Coefficient value at corresponding column and row indices.
coefficientValue34
Coefficient value at corresponding column and row indices.
coefficientValue44
Coefficient value at corresponding column and row indices.
coefficientValue50
Coefficient value at corresponding column and row indices.
coefficientValue60
Coefficient value at corresponding column and row indices.
coefficientValue51
Coefficient value at corresponding column and row indices.
coefficientValue61
Coefficient value at corresponding column and row indices.
coefficientValue52
Coefficient value at corresponding column and row indices.
coefficientValue62
Coefficient value at corresponding column and row indices.
coefficientValue53
Coefficient value at corresponding column and row indices.
coefficientValue63
Coefficient value at corresponding column and row indices.
coefficientValue54
Coefficient value at corresponding column and row indices.
coefficientValue64
Coefficient value at corresponding column and row indices.
coefficientValue05
Coefficient value at corresponding column and row indices.
coefficientValue15
Coefficient value at corresponding column and row indices.
coefficientValue25
Coefficient value at corresponding column and row indices.
coefficientValue35
Coefficient value at corresponding column and row indices.
coefficientValue45
Coefficient value at corresponding column and row indices.

coefficientValue55
Coefficient value at corresponding column and row indices.
coefficientValue65
Coefficient value at corresponding column and row indices.
coefficientValue06
Coefficient value at corresponding column and row indices.
coefficientValue16
Coefficient value at corresponding column and row indices.
coefficientValue26
Coefficient value at corresponding column and row indices.
coefficientValue36
Coefficient value at corresponding column and row indices.
coefficientValue46
Coefficient value at corresponding column and row indices.
coefficientValue56
Coefficient value at corresponding column and row indices.
coefficientValue66
Coefficient value at corresponding column and row indices.

Remarks

The function can also set the coefficient values for 3x3, 5x5 and 7x7 kernels.

EKernel::SetSize

Sets the number of coefficients along a row and along a column.

Namespace: Euresys::Open_eVision

```
[C++]
void SetSize(
 short n16SizeX,
 short n16SizeY
 )
```

Parameters

n16SizeX
The number of coefficients along a row.
n16SizeY
The number of coefficients along a column.

EKernel::GetSizeX

Number of coefficients along a row.

Namespace: Euresys::Open_eVision

[C++]

```
short GetSizeX()
```

EKernel::GetSizeY

Number of coefficients along a column.

Namespace: Euresys::Open_eVision

[C++]

```
short GetSizeY()
```

4.137. ELabeledImageSegmenter Class

Segments an image by mapping the value of the pixels directly to a layer index.

Remarks

This segmenter is applicable to EROIBW8 and EROIBW16 grayscale images. It produces coded images with a varying number of layers. The layer with index N contains all the unmasked pixels having a gray value equal to N.

By default, the segmentation is restricted to the range of layers whose index is between 0 and 255 (inclusive). This default range can be changed through ELabeledImageSegmenter::MinLayer and ELabeledImageSegmenter::MaxLayer.

Base Class:EImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetMaxLayer	High index of the range of layers to be encoded.
GetMinLayer	Low index of the range of layers to be encoded.
Load	Load the ELabeledImageSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the ELabeledImageSegmenter configuration. The given ESerializer must have been created for writing.
SetMaxLayer	High index of the range of layers to be encoded.
SetMinLayer	Low index of the range of layers to be encoded.

ELabeledImageSegmenter::Load

Load the ELabeledImageSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ELabeledImageSegmenter::GetMaxLayer

ELabeledImageSegmenter::SetMaxLayer

High index of the range of layers to be encoded.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
EBW16 GetMaxLayer() const
void SetMaxLayer(EBW16 maxLayer)
```

ELabeledImageSegmenter::GetMinLayer

ELabeledImageSegmenter::SetMinLayer

Low index of the range of layers to be encoded.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
EBW16 GetMinLayer() const
```

```
void SetMinLayer(EBW16 minLayer)
```

ELabeledImageSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool operator==(
 const ELabeledImageSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to.

ELabeledImageSegmenter::Save

Save the ELabeledImageSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.138. ELandmark Class

The landmark descriptor class.

Namespace: Euresys::Open_eVision

Methods

ELandmark	Constructs a ELandmark object.
GetSensorX	The X value of the sensor coordinate pair of the landmark.
GetSensorY	Gets the Y value of the sensor coordinate pair of the landmark.
GetWorldX	Gets the X value of the world coordinate pair of the landmark.
GetWorldY	Gets the Y value of the world coordinate pair of the landmark.
operator=	Assignement operator.
SetSensorX	The X value of the sensor coordinate pair of the landmark.
SetSensorY	Gets the Y value of the sensor coordinate pair of the landmark.
SetWorldX	Gets the X value of the world coordinate pair of the landmark.
SetWorldY	Gets the Y value of the world coordinate pair of the landmark.

ELandmark::ELandmark

Constructs a ELandmark object.

Namespace: Euresys::Open_eVision

[C++]

```
void ELandmark(
 )
```

ELandmark::operator=

Assignement operator.

Namespace: Euresys::Open_eVision

[C++]

```
ELandmark& operator=(
 const ELandmark& other
 )
```

Parameters

other

ELandmark object to copy.

ELandmark::GetSensorX

ELandmark::SetSensorX

The X value of the sensor coordinate pair of the landmark.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSensorX() const
void SetSensorX(float val)
```

ELandmark::GetSensorY

ELandmark::SetSensorY

Gets the Y value of the sensor coordinate pair of the landmark.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSensorY() const
void SetSensorY(float val)
```

ELandmark::GetWorldX

ELandmark::SetWorldX

Gets the X value of the world coordinate pair of the landmark.

Namespace: Euresys::Open_eVision

[C++]

```
float GetWorldX() const
void SetWorldX(float val)
```

ELandmark::GetWorldY

ELandmark::SetWorldY

Gets the Y value of the world coordinate pair of the landmark.

Namespace: Euresys::Open_eVision

[C++]

```
float GetWorldY() const
void SetWorldY(float val)
```

4.139. ELaserLineExtractor Class

Manages a laser line extraction context.

Namespace: Euresys::Open_eVision::Easy3D

Methods

ELaserLineExtractor	Creates an ELaserLineExtractor object.
ExtractProfileFromFrame	Extracts a profile from a frame and adds it to the current depth map. Returns true if the depth map is complete and ready for further processing.
GetAnalysisMode	Analysis mode.
GetAnalysisThreshold	Analysis threshold. Set this value to eliminate noise.
GetDepthMap	Returns the current depth map.
GetEnableSmoothing	Enables or disables grayscale profile smoothing before extraction.
GetProfile	Returns the last extracted profile.
operator=	Assignment operator
SetAnalysisMode	Analysis mode.
SetAnalysisThreshold	Analysis threshold. Set this value to eliminate noise.
SetEnableSmoothing	Enables or disables grayscale profile smoothing before extraction.
SetSmoothingParameters	Sets the parameters of the smoothing kernel.

ELaserLineExtractor::GetAnalysisMode

ELaserLineExtractor::SetAnalysisMode

Analysis mode.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EMaximumAnalysisMode GetAnalysisMode()
void SetAnalysisMode(Euresys::Open_eVision::Easy3D::EMaximumAnalysisMode mode)
```

ELaserLineExtractor::GetAnalysisThreshold

ELaserLineExtractor::SetAnalysisThreshold

Analysis threshold. Set this value to eliminate noise.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetAnalysisThreshold()
void SetAnalysisThreshold(int threshold)
```

Remarks

In the center of gravity (COG) analysis mode, this threshold is used to discriminate peaks and should be set accordingly.

ELaserLineExtractor::GetDepthMap

Returns the current depth map.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EDepthMap16& GetDepthMap()
```

Remarks

Should be called only when the previous call to ExtractProfileFromFrame() returned true. Otherwise, the depth map returned will be incomplete.

ELaserLineExtractor::ELaserLineExtractor

Creates an ELaserLineExtractor object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ELaserLineExtractor(
 int frameWidth,
 int frameHeight,
 int numFramesPerMap,
 float zResolution
 )

void ELaserLineExtractor(
 const ELaserLineExtractor& other
 )
```

Parameters

frameWidth

Width of the frames from which the profiles will be extracted.

frameHeight

Height of the frames from which the profiles will be extracted.

numFramesPerMap

Number of frames (and thus profiles) to be used per depth map. Each extracted profile create a line in the depth map.

zResolution

Optional parameter for the Z resolution of the extracted profile.
With a value of 0, the resolution will be automatically calculated to maximize the sub-pixel accuracy.

other

The object that should be copied

ELaserLineExtractor::GetEnableSmoothing

ELaserLineExtractor::SetEnableSmoothing

Enables or disables grayscale profile smoothing before extraction.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableSmoothing()

void SetEnableSmoothing(bool enable)
```

ELaserLineExtractor::ExtractProfileFromFrame

Extracts a profile from a frame and adds it to the current depth map.
Returns true if the depth map is complete and ready for further processing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ExtractProfileFromFrame(
 const EROIBW8& frame
 )
```

Parameters

frame

Frame from which the profile will be extracted.

ELaserLineExtractor::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ELaserLineExtractor& operator=(
 const ELaserLineExtractor& other
 )
```

Parameters

other

The object that should be copied

ELaserLineExtractor::GetProfile

Returns the last extracted profile.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<float> GetProfile()
```

Remarks

If a point could not be extracted, its value will be set to FLOAT_MAX.

ELaserLineExtractor::SetSmoothingParameters

Sets the parameters of the smoothing kernel.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetSmoothingParameters(
 int param0,
 int param1,
 int param2
 )
```

Parameters

param0

First kernel parameter.

param1

Second kernel parameter.

param2

Third kernel parameter.

Remarks

If enabled, the smoothing will be performed using the following formula: f[i] = (f[i-1] * param0) + (f[i] * param1) + (f[i+1] * param2).

4.140. ELine Class

Represents a model of a line segment.

Base Class:EFrame

Namespace: Euresys::Open_eVision

Methods

CopyTo	Copies all the data of the current ELine object into another ELine object and returns it.
ELine	Constructs a ELine object.
GetAngleBetweenLines	Computes the angle between two lines.
GetDistanceBetweenPointAndLine	Computes the distance between a point and a line.
GetEnd	End point coordinates of the ELine object.
GetIntersectionOfLines	Computes the intersection between two lines.
GetLength	Length of the ELine object.
GetOrg	Origin point coordinates of the ELine object.

GetPoint	Returns the coordinates of a point along the line.
GetProjectionOfPointOnLine	Computes the projection of a point on a line.
Load	Load the ELine configuration. The given ESerializer must have been created for reading.
operator=	Copies all the data from another ELine object into the current ELine object
Save	Save the ELine configuration. The given ESerializer must have been created for writing.
SetFromOriginAndEnd	Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELine object.
SetFromTwoPoints	DEPRECATED (you should use ELine::SetFromOriginAndEnd) : Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELine object.
SetLength	Length of the ELine object.

ELine::CopyTo

Copies all the data of the current ELine object into another ELine object and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ELine& other
 )

ELine* CopyTo(
 ELine* other
 )
```

Parameters

other

Pointer to the ELine object in which the current ELine object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ELine object will be created and returned.

ELine::ELine

Constructs a ELine object.

Namespace: Euresys::Open_eVision

[C++]

```
void ELine(
 )

void ELine(
 const EPoint& center,
 float length,
 float angle
 )

void ELine(
 const EPoint& origin,
 const EPoint& end
 )

void ELine(
 const ELine& other
 )
```

Parameters

center
Center coordinates of the line at its nominal position. The default value is (0,0).

length
Nominal length of the line. The default value is 100.

angle
Nominal rotation angle of the line. The default value is 0.

origin
Origin point coordinates of the line.

end
End point coordinates of the line.

other
Another ELine object to be copied in the new ELine object.

ELine::GetEnd

End point coordinates of the ELine object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetEnd() const
```

ELine::GetAngleBetweenLines

Computes the angle between two lines.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAngleBetweenLines(
 const ELine& line1,
 const ELine& line2
 )
```

Parameters

line1

First line

line2

Second line

Remarks

The angle returned by this function is signed in the trigonometric sense, meaning that angle (1,2) = -angle(2,1).

ELine::GetDistanceBetweenPointAndLine

Computes the distance between a point and a line.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDistanceBetweenPointAndLine(
 const EPoint& pt,
 const ELine& line,
 bool limited
 )
```

Parameters

pt

The point.

line

The line.

limited

Indicates if the line parameter should be considered as an infinite line or as a segment.

ELine::GetIntersectionOfLines

Computes the intersection between two lines.

Namespace: Euresys::Open_eVision

[C++]

```
int GetIntersectionOfLines(
 const ELine& line1,
 const ELine& line2,
 EPoint& intersection,
 bool limited
 )
```

Parameters

line1

First line.

line2

Second line.

intersection

Found intersection.

limited

Indicates if the line parameters should be considered as infinite lines or as a segments.

Remarks

The function returns the number of intersections found. It will return -1 if the two lines are overlapping.

ELine::GetPoint

Returns the coordinates of a point along the line.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the line length (range [-1, +1]).

ELine::GetProjectionOfPointOnLine

Computes the projection of a point on a line.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetProjectionOfPointOnLine(
 const EPoint& pt,
 const ELine& line
 )
```

Parameters

pt

The point.

line

The line.

ELine::GetLength

ELine::SetLength

Length of the ELine object.

Namespace: Euresys::Open_eVision

[C++]

```
float GetLength()

void SetLength(float length)
```

Remarks

By default, the length of the line is 100, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

ELine::Load

Load the ELine configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ELine::operator=

Copies all the data from another ELine object into the current ELine object

Namespace: Euresys::Open_eVision

```
[C++]
ELine& operator=(
 const ELine& other
 )
```

Parameters

other
ELine object to be copied

ELine::GetOrg

Origin point coordinates of the ELine object.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetOrg() const
```

ELine::Save

Save the ELine configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ELine::SetFromOriginAndEnd

Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELine object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOriginAndEnd(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin
Origin point coordinates of the line.

end
End point coordinates of the line.

ELine::SetFromTwoPoints

This method is deprecated.

DEPRECATED (you should use ELine::SetFromOriginAndEnd) : Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELine object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromTwoPoints(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin
Origin point coordinates of the line.

end
End point coordinates of the line.

4.141. ELineGauge Class

Manages a line fitting gauge.

Base Class:ELineShape

Namespace: Euresys::Open_eVision

Methods

AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.
CopyTo	Copies all the data of the current ELineGauge object into another ELineGauge object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
ELineGauge	Constructs a line measurement context.
GetAverageDistance	Average distance between the sampled points and the fitted model.
GetClippingMode	Clipping mode, that allows to choose how the fitted segment length and center are computed.
GetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
GetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
GetKnownAngle	Flag indicating whether the slope of the line to be fitted is known or not.
GetMeasuredLine	Information pertaining to the fitted line.
GetMeasuredPeak	Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.
GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.
GetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
GetMinArea	Minimum area value.
GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetNumFilteringPasses	Number of filtering passes for a model fitting operation.

GetNumMeasuredPoints	Number of edge-crossing points along the sample path of the gauge that was inspected with the last call to ELineGauge::MeasureSample.
GetNumSamples	Number of sampled points during the model fitting operation.
GetNumSkipRanges	Number of skip ranges in the gauge after a call to ELineGauge::AddSkipRange.
GetNumValidSamples	Number of valid sample points remaining after a model fitting operation.
GetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
GetSample	Allows to retrieve the sample points found along the line.
GetSamplingStep	Approximate distance between sampled points during a model fitting operation.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ELineGauge::AddSkipRange method.
GetSmoothing	Number of pixels used for the low-pass filtering operation.
GetThickness	Number of parallel segments used to extract the data profile.
GetThreshold	Threshold level used to delimit significant peaks in the data profile.
GetTolerance	Searching area half thickness of the line fitting gauge.
GetTransitionChoice	Transition choice.
GetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
GetTransitionType	Transition type.
GetType	Shape type.
GetValid	Flag indicating if at least one valid transition has been found.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without line fitting operation.
operator=	Compares the instance with another ELineGauge object and returns true if they are identical.
Plot	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
PlotWithCurrentPen	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

RemoveAllSkipRanges	Removes all the skip ranges previously created by a call to ELineGauge::AddSkipRange.
RemoveSkipRange	After a call to ELineGauge::AddSkipRange, removes the skip range with the given index.
SetActive	Sets the flag indicating whether the gauge is active or not.
SetClippingMode	Clipping mode, that allows to choose how the fitted segment length and center are computed.
SetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
SetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
SetKnownAngle	Flag indicating whether the slope of the line to be fitted is known or not.
SetLine	Sets the nominal position, length and rotation angle of the line fitting gauge, according to a known ELine object.
SetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
SetMinArea	Minimum area value.
SetMinNumFitSamples	Sets the minimum number of samples required for fitting on each side of the shape.
SetNumFilteringPasses	Number of filtering passes for a model fitting operation.
SetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
SetSamplingStep	Approximate distance between sampled points during a model fitting operation.
SetSmoothing	Number of pixels used for the low-pass filtering operation.
SetThickness	Number of parallel segments used to extract the data profile.
SetThreshold	Threshold level used to delimit significant peaks in the data profile.
SetTolerance	Searching area half thickness of the line fitting gauge.
SetTransitionChoice	Transition choice.
SetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
SetTransitionType	Transition type.

ELineGauge::SetActive

Sets the flag indicating whether the gauge is active or not.

Namespace: Euresys::Open_eVision

```
[C++]
void SetActive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying ELineGauge::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (true).

ELineGauge::AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 AddSkipRange(
 const OEV_UINT32 start,
 const OEV_UINT32 end
 )
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process. The ELineGauge::AddSkipRange method allows to define skip ranges in an ELineGauge. This means that, at measure time, samples belonging to these ranges will not be taken into account. A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another. The range is allowed to be reversed (i.e. end is not required to be greater than start). Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for ELineGauge::NumSamples).

ELineGauge::GetAverageDistance

Average distance between the sampled points and the fitted model.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAverageDistance()
```

Remarks

Irrelevant in case of a point location operation.

ELineGauge::GetClippingMode

ELineGauge::SetClippingMode

Clipping mode, that allows to choose how the fitted segment length and center are computed.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EClippingMode GetClippingMode() const

void SetClippingMode(Euresys::Open_eVision::EClippingMode clippingMode)
```

Remarks

By default, the clipping mode is EClippingMode_CenteredNominal, which corresponds to the behavior appearing in Open eVision version 6.4 and before.

ELineGauge::CopyTo

Copies all the data of the current ELineGauge object into another ELineGauge object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ELineGauge& other,
 bool recursive
 )

ELineGauge* CopyTo(
 ELineGauge* other,
 bool recursive
 )
```

Parameters

other

Pointer to the ELineGauge object in which the current ELineGauge object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ELineGauge object will be created and returned.

ELineGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int x,
 int y
 )
```

Parameters

x

Cursor current X coordinate.

y

Cursor current Y coordinate.

ELineGauge::Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELineGauge::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELineGauge::ELineGauge

Constructs a line measurement context.

Namespace: Euresys::Open_eVision

```
[C++]
void ELineGauge(
 )
void ELineGauge(
 const ELineGauge& other
 )
```

Parameters

other

Another ELineGauge object to be copied in the new ELineGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed line measurement context is based on a pre-existing ELineGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the ELineGauge::CopyTo method.

ELineGauge::GetFilteringThreshold

ELineGauge::SetFilteringThreshold

Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

Namespace: Euresys::Open_eVision

```
[C++]
float GetFilteringThreshold()
void SetFilteringThreshold(float f32FilteringThreshold)
```

Remarks

Irrelevant in case of a point location operation. During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).

ELineGauge::GetMeasuredPeak

Returns information pertaining to the default derivative peak, along one of the sample paths of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
EPeak GetMeasuredPeak(
 OEV_UINT32 index
 )
```

Parameters

index

This argument must be left unchanged from its default value, i.e. ~0 (= 0xFFFFFFFF).

Remarks

ELineGauge::GetMeasuredPeak returns the information about the derivative peak that meets the following two requirements: 1. It lies on the sample path inspected with the last call to ELineGauge::MeasureSample, and 1. It is associated with the edge-crossing point along the latter sample path that is selected by the transition choice parameter (cf. ELineGauge::TransitionChoice).

Note. For this method to succeed, it is necessary to previously call ELineGauge::MeasureSample.

ELineGauge::GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetMeasuredPoint(
 OEV_UINT32 index
 )
```

Parameters

index

This argument must be left unchanged from its default value, i.e. ~0 (= 0xFFFFFFFF).

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current ELineGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry. ELineGauge::GetMeasuredPoint returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to ELineGauge::MeasureSample, and 1. Among all the sample points along the latter sample path, it is the one selected by the ELineGauge::TransitionChoice property.

Note. For this method to succeed, it is necessary to previously call ELineGauge::MeasureSample.

ELineGauge::GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void GetMinNumFitSamples(
 int& side0,
 int& side1,
 int& side2,
 int& side3
 )
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

ELineGauge::GetSample

Allows to retrieve the sample points found along the line.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSample(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSample(
 ESamplePoint& pt,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will contain the sample position.

index

The sample index

Remarks

The method provides the sample point corresponding to the supplied index. The returned value corresponds to the sample validity.

ELineGauge::GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ELineGauge::AddSkipRange method.

Namespace: Euresys::Open_eVision

```
[C++]
void GetSkipRange(
 const OEV_UINT32 index,
 OEV_UINT32& start,
 OEV_UINT32& end
 )
```

Parameters

index

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

ELineGauge::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool daughters
 )
```

Parameters

daughters

true if the daughters gauges handles have to be considered as well.

ELineGauge::GetHVConstraint

ELineGauge::SetHVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetHVConstraint()
void SetHVConstraint(bool HVConstraint)
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

ELineGauge::GetKnownAngle

ELineGauge::SetKnownAngle

Flag indicating whether the slope of the line to be fitted is known or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetKnownAngle()
void SetKnownAngle(bool bKnownAngle)
```

Remarks

A line model to be fitted may have a well-known slope. It is possible to impose the value of this slope, thus removing one degree of freedom. The line fitting gauge slope is set by means of ELineGauge. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ELineGauge::SetLine

Sets the nominal position, length and rotation angle of the line fitting gauge, according to a known ELine object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetLine(const ELine& line)
```

ELineGauge::Measure

Triggers the point location or the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
void Measure(
 EROIBW8* sourceImage
 )
void Measure(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Pointer to the source image.

region

Region to use with the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ELineGauge::GetMeasuredLine

Information pertaining to the fitted line.

Namespace: Euresys::Open_eVision

```
[C++]
ELine GetMeasuredLine()
```

Remarks

Use method EShape::GetFound to get the status of the measurement. ELineGauge::MeasuredLine returns a successful fitted line if EShape::GetFound is true, otherwise it returns the orginal (nominal) line.

ELineGauge::MeasureSample

Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureSample(
 EROIBW8* sourceImage,
 OEV_UINT32 pathIndex
 )
void MeasureSample(
 EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 pathIndex
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

region

Region on which to measure.

Remarks

This method stores its results into a temporary variable inside the ELineGauge object.

ELineGauge::MeasureWithoutFitting

Triggers the point location without line fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureWithoutFitting(
 EROIBW8* sourceImage
 )
```

```
void MeasureWithoutFitting(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Source image.

region

The region on which to measure.

Remarks

This method performs the actual measurement for each transition, but does not perform the line fitting. This means that individual samples will be available through the ELineGauge::GetSample method, but the gauge position will not be changed. Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

ELineGauge::GetMinAmplitude

ELineGauge::SetMinAmplitude

Offset added to the Threshold when a peak is to be detected.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinAmplitude()
```

```
void SetMinAmplitude(OEV_UINT32 un32MinAmplitude)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value. To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected. When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

ELineGauge::GetMinArea

ELineGauge::SetMinArea

Minimum area value.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinArea()
void SetMinArea(OEV_UINT32 un32MinArea)
```

Remarks

A transition is detected if its derivative peak reaches Threshold + MinAmplitude value, and then declared valid if the area between the peak curve and the horizontal at level Threshold reaches the MinArea value.

ELineGauge::GetNumFilteringPasses

ELineGauge::SetNumFilteringPasses

Number of filtering passes for a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumFilteringPasses()
void SetNumFilteringPasses(OEV_UINT32 un32NumFilteringPasses)
```

Remarks

Irrelevant in case of a point location operation. During a filtering pass, the points that are too distant from the model are discarded. During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious). By default (the number of filtering passes is 0), the outliers rejection process is disabled.

ELineGauge::GetNumMeasuredPoints

Number of edge-crossing points along the sample path of the gauge that was inspected with the last call to ELineGauge::MeasureSample.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumMeasuredPoints()
```

Remarks

Note. For this method to succeed, it is necessary to previously call ELineGauge::MeasureSample.

ELineGauge::GetNumSamples

Number of sampled points during the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamples() const
```

Remarks

Irrelevant in case of a point location operation. After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

ELineGauge::GetNumSkipRanges

Number of skip ranges in the gauge after a call to ELineGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSkipRanges() const
```

ELineGauge::GetNumValidSamples

Number of valid sample points remaining after a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumValidSamples()
```

Remarks

Irrelevant in case of a point location operation. After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

ELineGauge::operator=

Compares the instance with another ELineGauge object and returns true if they are identical.

Namespace: Euresys::Open_eVision

[C++]

```
ELineGauge& operator=(
 const ELineGauge& other
 )
```

Parameters

other

ELineGauge object to be compared

ELineGauge::Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void Plot(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.

height
Height of the plot.

originX
Origin point coordinates of the plot along the X axis.

originY
Origin point coordinates of the plot along the Y axis.

color
The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to ELineGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call ELineGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELineGauge::PlotWithCurrentPen

This method is deprecated.

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void PlotWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.
height
Height of the plot.
originX
Origin point coordinates of the plot along the X axis.
originY
Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to ELineGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call ELineGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELineGauge::Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

Namespace: Euresys::Open_eVision

[C++]

```
void Process(
 EROIBW8* sourceImage,
 bool daughters
 )

void Process(
 EROIBW8* sourceImage,
 const ERegion& region,
 bool daughters
 )
```

Parameters

sourceImage
Pointer to the BW8 roi source image.
daughters
Flag indicating whether the daughters shapes inherit of the same behavior.
region
-

Remarks

When complex gauging is required, several gauges can be grouped together. Applying Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

ELineGauge::GetRectangularSamplingArea

ELineGauge::SetRectangularSamplingArea

Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRectangularSamplingArea()

void SetRectangularSamplingArea(bool bRectangularSamplingArea)
```

Remarks

By default, this flag is set to true: the sampling area always remains a rectangle. Setting this property is only useful when the thickness transition parameter is greater than 1. In fact, when thickness transition parameter is equal to 1, rectangle and parallelogram reduce to the same segment.

ELineGauge::RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to ELineGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveAllSkipRanges(
 )
```

ELineGauge::RemoveSkipRange

After a call to ELineGauge::AddSkipRange, removes the skip range with the given index.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveSkipRange(
 const OEV_UINT32 index
 )
```

Parameters

index

Index of the skip range to remove, as returned by ELineGauge::AddSkipRange.

ELineGauge::GetSamplingStep

ELineGauge::SetSamplingStep

Approximate distance between sampled points during a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
float GetSamplingStep()
void SetSamplingStep(float f32SamplingStep)
```

Remarks

Irrelevant in case of a point location operation. To fit a model, a series of point location operations are performed along the model. The point location gauges spacing is given by the sampling step. By default, the sampling step is set to 5, which means 5 pixels when the field of view is not calibrated, and 5 physical units in case of a calibrated field of view. Be aware that if the sampling step is too large, the number of sampled points along the model will not be sufficient enough to accurately locate it.

ELineGauge::SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void SetMinNumFitSamples(
 int side0,
 int side1,
 int side2,
 int side3
 )
```

Parameters

side0

Required number of samples to correctly fit the line. The default value is 2. It is the only parameter taken into account.

side1

Not used.

side2

Not used.

side3

Not used.

Remarks

Irrelevant in case of a point location operation. When the ELineGauge::Measure method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ELineGauge::GetSmoothing

ELineGauge::SetSmoothing

Number of pixels used for the low-pass filtering operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetSmoothing()
void SetSmoothing(OEV_UINT32 smoothing)
```

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

ELineGauge::GetThickness

ELineGauge::SetThickness

Number of parallel segments used to extract the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThickness()
void SetThickness(OEV_UINT32 thickness)
```

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

ELineGauge::GetThreshold

ELineGauge::SetThreshold

Threshold level used to delimit significant peaks in the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 threshold)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value. To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected. When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

ELineGauge::GetTolerance

ELineGauge::SetTolerance

Searching area half thickness of the line fitting gauge.

Namespace: Euresys::Open_eVision

```
[C++]
float GetTolerance()
void SetTolerance(float tolerance)
```

Remarks

By default, the searching area thickness of the line fitting gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

ELineGauge::GetTransitionChoice

ELineGauge::SetTransitionChoice

Transition choice.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionChoice GetTransitionChoice()
void SetTransitionChoice(Euresys::Open_eVision::ETransitionChoice transitionChoice)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. In case of ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd transition choice, set ELineGauge::TransitionIndex to specify the desired transition. By default, the selected transition corresponds to the one with the largest amplitude (ETransitionChoice_LargestAmplitude).

ELineGauge::GetTransitionIndex

ELineGauge::SetTransitionIndex

Index (from 0 on) of the transition to be retained when the transition choice parameter is set to NthFromBegin or NthFromEnd.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTransitionIndex()
void SetTransitionIndex(OEV_UINT32 transitionIndex)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. By default, the first transition is retained (the index value is 0).

ELineGauge::GetTransitionType

ELineGauge::SetTransitionType

Transition type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::ETransitionType GetTransitionType()
void SetTransitionType(Euresys::Open_eVision::ETransitionType transitionType)
```

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object. By default, the searched transition type is indifferently a black to white or a white to black transition (ETransitionType_BwOrWb).

ELineGauge::GetType

Shape type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EShapeType GetType()
```

ELineGauge::GetValid

Flag indicating if at least one valid transition has been found.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetValid()
```

Remarks

A false value means that no measurement has been performed. A true value means that a transition was found along the sample path inspected with the last call to ELineGauge::MeasureSample, and thus a point has been measured.

4.142. ELineShape Class

Manages a line shape.

Base Class:EShape
Derived Class(es):ELineGauge
Namespace: Euresys::Open_eVision

Methods

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copies all the data of the current ELineShape object into another ELineShape object and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the shape.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
ELineShape	Constructs a ELineShape object.
GetAngle	Orientation of the shape.
GetCenter	Center point of the frame.
GetCenterX	Abscissa of the origin point of the frame.
GetCenterY	Ordinate of the origin point of the frame.
GetEnd	End point coordinates of the ELineShape object.
GetLength	Length of the ELineShape object.
GetOrg	Origin point coordinates of the ELineShape object.
GetPoint	Returns the coordinates of a point along the line.
GetScale	Horizontal sensor resolution, in pixels per unit.
GetType	Shape type.
HitTest	Checks if there is a handle under the cursor.
operator=	Assignment operator
SetAngle	Orientation of the shape.
SetCenter	Center point of the frame.
SetCenterXY	Sets the center coordinates of a ELineShape object.
SetFromOriginAndEnd	Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELineShape object.
SetFromTwoPoints	DEPRECATED (you should use ELineShape::SetFromOriginAndEnd) : Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELineShape object.
SetLength	Length of the ELineShape object.
SetLine	Sets the nominal position, length and rotation angle of the line, according to a known ELine object.
SetScale	Horizontal sensor resolution, in pixels per unit.

ELineShape::GetAngle

ELineShape::SetAngle

Orientation of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
void SetAngle(float f32Angle)
```

ELineShape::GetCenter

ELineShape::SetCenter

Center point of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

ELineShape::GetCenterX

Abscissa of the origin point of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

ELineShape::GetCenterY

Ordinate of the origin point of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

ELineShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

```
[C++]
void Closest(
 )
```

ELineShape::CopyTo

Copies all the data of the current ELineShape object into another ELineShape object and returns it.

Namespace: Euresys::Open_eVision

```
[C++]
void CopyTo(
 ELineShape& other,
 bool bRecursive
 )

ELineShape* CopyTo(
 ELineShape* dest,
 bool bRecursive
 )
```

Parameters

other

-

bRecursive

true if the children shapes have to be copied as well, false otherwise.

dest

Pointer to the ELineShape object in which the current ELineShape object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ELineShape object will be created and returned.

ELineShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

ELineShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELineShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELineShape::ELineShape

Constructs a ELineShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void ELineShape(
 )
void ELineShape(
 const EPoint& origin,
 const EPoint& end
 )
void ELineShape(
 const EPoint& center,
 float length,
 float angle
 )
void ELineShape(
 const ELineShape& other
 )
```

Parameters

origin
Origin point coordinates of the line.

end
End point coordinates of the line.

center
Center coordinates of the line at its nominal position. The default value is (0,0).

length
Nominal length of the line. The default value is 100.

angle
Nominal rotation angle of the line. The default value is 0.

other
Another ELineShape object to be copied in the new ELineShape object.

ELineShape::GetEnd

End point coordinates of the ELineShape object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetEnd()
```

ELineShape::GetPoint

Returns the coordinates of a point along the line.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the line length (range [-1, +1]).

ELineShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

ELineShape::GetLength

ELineShape::SetLength

Length of the ELineShape object.

Namespace: Euresys::Open_eVision

```
[C++]
float GetLength()
void SetLength(float length)
```

Remarks

By default, the length of the line is 100, which means 100 pixels when the field of view is not calibrated, and 100 physical units in case of a calibrated field of view.

ELineShape::SetLine

Sets the nominal position, length and rotation angle of the line, according to a known ELine object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetLine(const ELine& line)
```

ELineShape::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
ELineShape& operator=(
 const ELineShape& other
 )
```

Parameters

other

Reference to the ELineShape object used for the assignment

ELineShape::GetOrg

Origin point coordinates of the ELineShape object.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetOrg()
```

ELineShape::GetScale

ELineShape::SetScale

Horizontal sensor resolution, in pixels per unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetScale() const
void SetScale(float f32Scale)
```

ELineShape::SetCenterXY

Sets the center coordinates of a ELineShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the ELineShape object.

centerY

Center coordinates of the ELineShape object.

ELineShape::SetFromOriginAndEnd

Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELineShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromOriginAndEnd(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin

Origin point coordinates of the line.

end

End point coordinates of the line.

ELineShape::SetFromTwoPoints

This method is deprecated.

DEPRECATED (you should use ELineShape::SetFromOriginAndEnd) : Sets the geometric parameters (center coordinates, length, and rotation angle) of a ELineShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromTwoPoints(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin

Origin point coordinates of the line.

end

End point coordinates of the line.

ELineShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

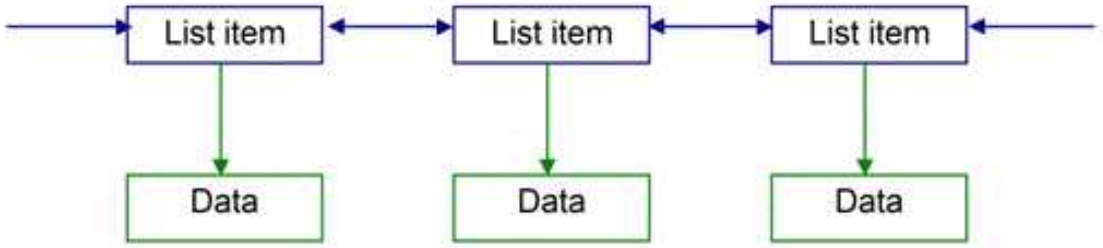
4.143. EListItem Class

This class is deprecated.

Describes list items. This class pertains to the EasyObject legacy API. Please use ECodedImage2 for all new developments instead.

Remarks

A list is a sequence of orderly list items. Each list item contains a pointer to a memory zone containing its data, a pointer to the previous list item, and a pointer to the next list item.



List itemsA few ECodedImage methods handle EListItem objects, or EListItem pointers. Runs listsECodedImage::GetFirstRunData, ECodedImage::GetFirstRunPtr, ECodedImage::GetLastRunData, ECodedImage::GetLastRunPtr, ECodedImage::GetPreviousRunData, ECodedImage::GetPreviousRunPtr, ECodedImage::GetNextRunData, ECodedImage::GetNextRunPtr These properties and methods allow to traverse the runs lists from the first run to the last, or from one run to its previous or next neighbor. A run can also be directly reached by its index within the list. The first run has index 0. The last run has index NumRuns-1. The ECodedImage::GetRunData and ECodedImage::GetRunDataPtr methods return the run data, or a pointer to the run data. Objects listsECodedImage::GetFirstObjData, ECodedImage::GetLastObjData, ECodedImage::GetPreviousObjData, ECodedImage::GetPreviousObjPtr, ECodedImage::GetNextObjData, ECodedImage::GetNextObjPtr These properties and methods allow to traverse the objects lists from the first object to the last, or from one object to its previous or next neighbor. An object can also be directly reached by its index within the list. The first object has index 0. The last object has index NumObjects-1. The ECodedImage::GetObjectData and ECodedImage::GetObjDataPtr methods return the object data, or a pointer to the object data.

Namespace: Euresys::Open_eVision

4.144. ELocator Class

EasyLocate tool - axis aligned bounding box version.

The ELocator locates and classifies objects (or defects). With this class, an object is defined by the axis-aligned bounding box that surrounds it and a label (see ELocatorObject class). The tool must be trained using a dataset with annotated objects.
The minimum size resolution supported by the tool is 128. The maximum size of the image supported is 500 000 pixels.
The tool has 5 main parameters:
- A set of anchors, i.e. pre-defined object size that it must be able to detect. The anchors can be automatically determined from the training dataset, specified manually using ELocator, or generated according to size information about the objects (see ELocator::GenerateAnchors).
- A detection threshold (ELocatorBase::DetectionThreshold) to accept or reject predicted objects based on their score.
- The maximum number of objects in an image (ELocatorBase::MaxNumberOfObjects)
- The maximum overlap between predicted objects with the same label (ELocator::SameLabelMaxOverlap, ELocatorBase::SameLabelMaxObjectProximity)
- The maximum overlap between predicted objects regardless of their label (ELocator::AbsoluteMaxOverlap, ELocatorBase::AbsoluteMaxObjectProximity).
Except for the anchors, all the parameters can be changed before and after training the tool.

Base Class:ELocatorBase

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

ELocator	Constructs a ELocator object.
GenerateAnchors	Generates anchors based on the objects in the dataset or a formal specification. For the version that takes a dataset, the properties ELocator and ELocator must be set.
GetAbsoluteMaxOverlap	Maximum overlap (intersection over union) between two predicted objects regardless of their label. The value of the parameter must be between 0 (no overlap allowed between two predicted objects with different labels) and 1 (full overlap allowed between two predicted objects with different labels). Default value: 1.
GetSameLabelMaxOverlap	Maximum overlap (intersection over union) between two predicted objects with the same label. The value of the parameter must be between 0 (no overlap allowed between two predicted objects with the same label) and 1 (full overlap allowed between two predicted objects with the same label). A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects. Default value: 0.5
GetToolType	Type of the deep learning tool.
operator=	Assignment operator
SerializeSettings	Serializes the settings of the locator.
SetAbsoluteMaxOverlap	Maximum overlap (intersection over union) between two predicted objects regardless of their label. The value of the parameter must be between 0 (no overlap allowed between two predicted objects with different labels) and 1 (full overlap allowed between two predicted objects with different labels). Default value: 1.
SetSameLabelMaxOverlap	Maximum overlap (intersection over union) between two predicted objects with the same label. The value of the parameter must be between 0 (no overlap allowed between two predicted objects with the same label) and 1 (full overlap allowed between two predicted objects with the same label). A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects. Default value: 0.5

ELocator::GetAbsoluteMaxOverlap

ELocator::SetAbsoluteMaxOverlap

Maximum overlap (intersection over union) between two predicted objects regardless of their label.
The value of the parameter must be between 0 (no overlap allowed between two predicted objects with different labels) and 1 (full overlap allowed between two predicted objects with different labels).
Default value: 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAbsoluteMaxOverlap() const

void SetAbsoluteMaxOverlap(float val)
```

Remarks

This parameter can be changed before and after training.

ELocator::ELocator

Constructs a ELocator object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ELocator(
 )

void ELocator(
 const ELocator& other
 )
```

Parameters

other

Reference to the ELocator object that should be copied

ELocator::GenerateAnchors

Generates anchors based on the objects in the dataset or a formal specification.
For the version that takes a dataset, the properties ELocator and ELocator must be set.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::ESize> GenerateAnchors(
 const EClassificationDataset& dataset
 )
std::vector<Euresys::Open_eVision::ESize> GenerateAnchors(
 float minDimension,
 float maxDimension,
 int numSubScales,
 std::vector<float> aspectRatios
 )
```

Parameters

dataset

Dataset

minDimension

Minimum dimension of objects (minimum value: 16)

maxDimension

Maximum dimension of objects.

numSubScales

Number of sub-scales to produce for each scale covered by the given dimensions.

aspectRatios

Aspect ratios to generate at each scale. The aspect ratios are the ratios between the width and height of the anchor. Recommended value: {1.0f, 0.5f, 2.0f}.

Remarks

The dimension of an object corresponds to the square root of its area.

ELocator::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ELocator& operator=(
 const ELocator& other
 )
```

Parameters

other

Reference to the ELocator object that should be copied

ELocator::GetSameLabelMaxOverlap

ELocator::SetSameLabelMaxOverlap

Maximum overlap (intersection over union) between two predicted objects with the same label.
The value of the parameter must be between 0 (no overlap allowed between two predicted objects with the same label) and 1 (full overlap allowed between two predicted objects with the same label). A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects.
Default value: 0.5

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSameLabelMaxOverlap() const

void SetSameLabelMaxOverlap(float val)
```

Remarks

This parameter is also used to compute the matching between ground truth and predicted objects during evaluation.
This parameter can be changed before and after training.

ELocator::SerializeSettings

Serializes the settings of the locator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer

Pointer to ESerializer

ELocator::GetToolType

Type of the deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetToolType() const
```

4.145. ELocatorBase Class

Base class for EasyLocate tool.
The children classes of this class differs by their prediction features (ELocatorFeature). The ground truth objects used to train must have the same set of features as the prediction.

Base Class:EDeepLearningTool

Derived Class(es):EInterestPointLocatorELocator

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Apply	Applies the tool on the given image and its mask region.
Evaluate	Evaluates the dataset.
GetAbsoluteMaxObjectProximity	Maximum proximity between two predicted objects regardless of their label. The value of the parameter can be between - 0 when the two objects are completely outside their respective zone of influence; and - 1 when the two predicted objects are the same. The actual implementation of the proximity depends on the type of locator tool. A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects. See also ELocator::AbsoluteMaxOverlap and EInterestPointLocator::AbsoluteMinDistance. Default value: 1.
GetCapacity	Capacity of the ELocatorBase. A higher capacity makes the supervised segmenter capable of learning more information at the cost of a slower processing speed.
GetChannels	Number of channels of input images. It can either be 1 for grayscale images or 3 for color images.
GetDetectionThreshold	Detection threshold. The detection threshold is set automatically during training to maximize accuracy. It can also be changed after training to obtain a different true positive/false positive tradeoff.
GetHeight	Height of input images.
GetLocatorFeatures	The features supported by the locator.
GetMaxNumberOfObjects	Maximum number of objects in an image (default value: 100).

GetPredictionAnchors	Prediction anchors. The prediction anchors are a set of object bounding box sizes. Each anchor is used to detect objects with a size similar to that anchor. As such, the prediction anchors must reflect the variety of sizes of objects that must be detected.
GetSameLabelMaxObjectProximity	Maximum proximity between two predicted objects with the same label. The value of the parameter can be between - 0 when the two objects are completely outside their respective zone of influence; and - 1 when the two predicted objects are the same. The actual implementation of the proximity depends on the type of locator tool. A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects. See also ELocator::SameLabelMaxOverlap and EInterestPointLocator::SameLabelMinDistance. Default value: 0.5
GetTrainingMetrics	Training metrics at the given iteration
GetValidationMetrics	Validation metrics at the given iteration
GetWidth	Width of input images.
HasFeature	Whether the given feature is enabled.
SerializeSettings	Serializes the settings of the locator.
SetAbsoluteMaxObjectProximity	Maximum proximity between two predicted objects regardless of their label. The value of the parameter can be between - 0 when the two objects are completely outside their respective zone of influence; and - 1 when the two predicted objects are the same. The actual implementation of the proximity depends on the type of locator tool. A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects. See also ELocator::AbsoluteMaxOverlap and EInterestPointLocator::AbsoluteMinDistance. Default value: 1.
SetCapacity	Capacity of the ELocatorBase. A higher capacity makes the supervised segmenter capable of learning more information at the cost of a slower processing speed.
SetChannels	Number of channels of input images. It can either be 1 for grayscale images or 3 for color images.
SetDetectionThreshold	Detection threshold. The detection threshold is set automatically during training to maximize accuracy. It can also be changed after training to obtain a different true positive/false positive tradeoff.
SetHeight	Height of input images.

SetMaxNumberOfObjects	Maximum number of objects in an image (default value: 100).
SetPredictionAnchors	Prediction anchors. The prediction anchors are a set of object bounding box sizes. Each anchor is used to detect objects with a size similar to that anchor. As such, the prediction anchors must reflect the variety of sizes of objects that must be detected.
SetSameLabelMaxObjectProximity	Maximum proximity between two predicted objects with the same label. The value of the parameter can be between - 0 when the two objects are completely outside their respective zone of influence; and - 1 when the two predicted objects are the same. The actual implementation of the proximity depends on the type of locator tool. A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects. See also ELocator::SameLabelMaxOverlap and EInterestPointLocator::SameLabelMinDistance. Default value: 0.5
SetWidth	Width of input images.

ELocatorBase::GetAbsoluteMaxObjectProximity

ELocatorBase::SetAbsoluteMaxObjectProximity

Maximum proximity between two predicted objects regardless of their label.
The value of the parameter can be between
- 0 when the two objects are completely outside their respective zone of influence; and
- 1 when the two predicted objects are the same.
The actual implementation of the proximity depends on the type of locator tool.
A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects.
See also ELocator::AbsoluteMaxOverlap and EInterestPointLocator::AbsoluteMinDistance. Default value: 1.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAbsoluteMaxObjectProximity() const

void SetAbsoluteMaxObjectProximity(float val)
```

Remarks

This parameter can be changed before and after training.

ELocatorBase::Apply

Applies the tool on the given image and its mask region.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ELocatorResult Apply(
 const EBaseROI& img
 )

ELocatorResult Apply(
 const EBaseROI& img,
 const ERegion& mask
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW8>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW8>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW16>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW16>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageC24>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageC24>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<const Euresys::Open_eVision::EBaseROI*>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorResult> Apply(
 const std::vector<const Euresys::Open_eVision::EBaseROI*>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )
```

Parameters

img
Image.

mask
Mask region of the image.

imgs
Vector of images.

masks
Vector of mask regions for the images.

ELocatorBase::GetCapacity

ELocatorBase::SetCapacity

Capacity of the ELocatorBase.
A higher capacity makes the supervised segmenter capable of learning more information at the cost of a slower processing speed.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
Euresys::Open_eVision::EasyDeepLearning::ELocatorCapacity GetCapacity() const
void SetCapacity(Euresys::Open_eVision::EasyDeepLearning::ELocatorCapacity capacity)
```

ELocatorBase::GetChannels

ELocatorBase::SetChannels

Number of channels of input images. It can either be 1 for grayscale images or 3 for color images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetChannels() const
void SetChannels(OEV_UINT32 channels)
```

Remarks

If the locator tool is not trained or the value was not explicitely set, its value will be 0.

ELocatorBase::GetDetectionThreshold

ELocatorBase::SetDetectionThreshold

Detection threshold. The detection threshold is set automatically during training to maximize accuracy. It can also be changed after training to obtain a different true positive/false positive tradeoff.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetDetectionThreshold() const

void SetDetectionThreshold(float threshold)
```

ELocatorBase::Evaluate

Evaluates the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ELocatorMetrics Evaluate(
 const EClassificationDataset& dataset
 )
```

Parameters

dataset

Dataset to evaluate

ELocatorBase::GetTrainingMetrics

Training metrics at the given iteration

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ELocatorMetrics GetTrainingMetrics(
 int iteration
 )
```

Parameters

iteration

Iteration at which to get the metrics

ELocatorBase::GetValidationMetrics

Validation metrics at the given iteration

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ELocatorMetrics GetValidationMetrics(
 int iteration
 )
```

Parameters

iteration

Iteration at which to get the metrics

ELocatorBase::HasFeature

Whether the given feature is enabled.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasFeature(
 Euresys::Open_eVision::EasyDeepLearning::ELocatorFeature feature
 )
```

Parameters

feature

The feature to check

ELocatorBase::GetHeight

ELocatorBase::SetHeight

Height of input images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetHeight() const
void SetHeight(OEV_UINT32 height)
```

Remarks

If the locator tool is not trained or the value was not explicitely set, its value will be 0.

ELocatorBase::GetLocatorFeatures

The features supported by the locator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetLocatorFeatures() const
```

ELocatorBase::GetMaxNumberOfObjects

ELocatorBase::SetMaxNumberOfObjects

Maximum number of objects in an image (default value: 100).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetMaxNumberOfObjects() const
void SetMaxNumberOfObjects(int val)
```

Remarks

The value of this parameter will be automatically changed when training if it is lower than the maximum number of objects in an image from the training dataset.
This parameter can be changed before and after training.

ELocatorBase::GetPredictionAnchors

ELocatorBase::SetPredictionAnchors

Prediction anchors.
The prediction anchors are a set of object bounding box sizes. Each anchor is used to detect objects with a size similar to that anchor. As such, the prediction anchors must reflect the variety of sizes of objects that must be detected.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::ESize> GetPredictionAnchors() const
void SetPredictionAnchors(const std::vector<Euresys::Open_eVision::ESize>& anchors)
```

ELocatorBase::GetSameLabelMaxObjectProximity

ELocatorBase::SetSameLabelMaxObjectProximity

Maximum proximity between two predicted objects with the same label.
The value of the parameter can be between
- 0 when the two objects are completely outside their respective zone of influence; and
- 1 when the two predicted objects are the same.
The actual implementation of the proximity depends on the type of locator tool.
A low value will reduce the amount of falsely detected objects but may increase the number of missed objects. A high value will increase the number of falsely detected objects and reduce the number of missed objects.
See also ELocator::SameLabelMaxOverlap and EInterestPointLocator::SameLabelMinDistance.
Default value: 0.5

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSameLabelMaxObjectProximity() const

void SetSameLabelMaxObjectProximity(float val)
```

Remarks

This parameter is also used to compute the matching between ground truth and predicted objects during evaluation.
This parameter can be changed before and after training.

ELocatorBase::SerializeSettings

Serializes the settings of the locator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer
Pointer to ESerializer

ELocatorBase::GetWidth

ELocatorBase::SetWidth

Width of input images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetWidth() const
void SetWidth(OEV_UINT32 width)
```

Remarks

If the locator tool is not trained or the value was not explicitely set, its value will be 0.

4.146. ELocatorMetrics Class

Collection of metrics used to evaluate the results of a ELocator tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

ELocatorMetrics	Constructs an ELocatorMetrics object.
GetAveragePrecision	Average Precision when matching prediction and ground truth with a minimum proximity of ELocatorBase::SameLabelMaxObjectProximity.
GetAveragePrecision50	Average precision when matching prediction and ground truth with a minimum "proximity" of 0.5.
GetAverageProximity	Average proximity of the predicted objects to the ground truth objects. This value is the average of ELocatorMetrics::GetLabelAverageProximity over all the labels.
GetBestWeightedFScore	Best weighted F-Score achievable by changing the detection threshold.
GetBestWeightedFScoreAndThreshold	Best weighted F-Score achievable with the corresponding threshold.
GetBestWeightedFScoreThreshold	Detection threshold that gives the best weighted F-Score (ELocatorMetrics).
GetBestWeightedPrecision	Best weighted precision achievable by changing the detection threshold.
GetBestWeightedPrecisionAndThreshold	Best weighted precision achievable with the corresponding threshold.
GetBestWeightedPrecisionThreshold	Detection threshold that gives the best weighted precision (ELocatorMetrics).

GetBestWeightedRecall	Best weighted recall achievable by changing the detection threshold.
GetBestWeightedRecallAndThreshold	Best weighted recall achievable with the corresponding threshold.
GetBestWeightedRecallThreshold	Detection threshold that gives the best weighted recall (ELocatorMetrics).
GetDetectionThreshold	Threshold for detected objects.
GetError	Error of the EasyLocate training algorithm (only available for the training metrics, see EDeepLearningTool).
GetFScore	F-Score (harmonic mean of ELocatorMetrics::Recall and ELocatorMetrics::Precision).
GetImageAccuracy	Image accuracy: the proportion of images that are correctly detected to contain or not objects, regardless of their labels.
GetIntersectionOverUnion	Deprecated. Same as ELocatorMetrics::AverageProximity.
GetLabel	Label.
GetLabelAveragePrecision	Average precision for detection of objects from the given label.
GetLabelAverageProximity	Average proximity of the predicted objects of the given label to the ground truth objects.
GetLabelFScore	F-Score for the given label.
GetLabelIntersectionOverUnion	Deprecated. Same as ELocatorMetrics::GetLabelAverageProximity.
GetLabelPrecision	Precision for the given label.
GetLabelRecall	Recall for the given label.
GetNumBadlyPredictedImagesWithObjects	Number of images with objects that are badly predicted as containing no object. This is the number of false negatives at the image level.
GetNumBadlyPredictedImagesWithoutObjects	Number of images without objects that are badly predicted as containing objects. This is the number of false positives a the image level.
GetNumCorrectlyDetectedObjects	Number of correctly detected objects. A correctly detected object is a predicted object that has a "proximity" higher than ELocatorBase::SameLabelMaxObjectProximity with a ground truth object of the same label. A label can be specified to obtain the number of correctly detected objects for that label. Otherwise, the number of correctly detected objects is for all the labels.
GetNumCorrectlyPredictedImagesWithObjects	Number of images containing objects that are correctly predicted as containing objects, regardless of the labels of the objects. This is the number of true positives at the image level.
GetNumCorrectlyPredictedImagesWithoutObjects	Number of images without objects that are correctly predicted as containing no object. This is the number of true negatives at the image level.

GetNumDetectedObjects	Number of detected objects. A label can be specified to obtain the number of detected objects for that label. Otherwise, the number of detected objects is for all the labels.
GetNumLabels	Number of labels recognized by the ELocator tool that produced these metrics.
GetNumUndetectedObjects	Number of undetected objects. An undetected object is a ground truth object that is not matched to any predicted object of the same label with a proximity higher than ELocatorBase::SameLabelMaxObjectProximity. A label can be specified to obtain the number of undetected objects for that label. Otherwise, the number of undetected objects is for all the labels.
GetPrecision	Precision (proportion of detected objects that are correct).
GetRecall	Recall (true positive rate, proportion of ground truth object that are correctly detected).
GetWeightedFScore	Weighted average of the F-Score for each label.
GetWeightedPrecision	Weighted average of the precision (proportion of detected objects that are correct) for each label.
GetWeightedRecall	Weighted average of the recall (true positive rate) for each label.
IsValid	Indicates whether the object contains at least one result.
Load	Loads a locator metric. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves a locator metric. The given ESerializer must have been created for writing.
SetDetectionThreshold	Threshold for detected objects.

ELocatorMetrics::GetAveragePrecision

Average Precision when matching prediction and ground truth with a minimum proximity of ELocatorBase::SameLabelMaxObjectProximity.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAveragePrecision() const
```

ELocatorMetrics::GetAveragePrecision50

Average precision when matching prediction and ground truth with a minimum "proximity" of 0.5.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAveragePrecision50() const
```

ELocatorMetrics::GetAverageProximity

Average proximity of the predicted objects to the ground truth objects.
This value is the average of ELocatorMetrics::GetLabelAverageProximity over all the labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAverageProximity() const
```

Remarks

Look at the documentation of ELocator and EInterestPointLocator for their definition of proximity.

ELocatorMetrics::GetDetectionThreshold

ELocatorMetrics::SetDetectionThreshold

Threshold for detected objects.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetDetectionThreshold() const

void SetDetectionThreshold(float threshold)
```

ELocatorMetrics::ELocatorMetrics

Constructs an ELocatorMetrics object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ELocatorMetrics(
 )

void ELocatorMetrics(
 const ELocatorMetrics& other
 )
```

Parameters

other

Reference to the ELocatorMetrics object that should be copied

ELocatorMetrics::GetError

Error of the EasyLocate training algorithm (only available for the training metrics, see EDeepLearningTool).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetError() const
```

ELocatorMetrics::GetFScore

F-Score (harmonic mean of ELocatorMetrics::Recall and ELocatorMetrics::Precision).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetFScore() const
```

ELocatorMetrics::GetBestWeightedFScore

Best weighted F-Score achievable by changing the detection threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedFScore(
 )

float GetBestWeightedFScore(
 const std::vector<float>& weights
 )
```

Parameters

weights

Label weights

ELocatorMetrics::GetBestWeightedFScoreAndThreshold

Best weighted F-Score achievable with the corresponding threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void GetBestWeightedFScoreAndThreshold(
 float& fScore,
 float& threshold
 )

void GetBestWeightedFScoreAndThreshold(
 const std::vector<float>& weights,
 float& fScore,
 float& threshold
 )
```

Parameters

fScore
Best weighted F-Score achievable

threshold
Threshold that achieves the best weighted F-Score

weights
Label weights

ELocatorMetrics::GetBestWeightedFScoreThreshold

Detection threshold that gives the best weighted F-Score (ELocatorMetrics).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedFScoreThreshold(
 )

float GetBestWeightedFScoreThreshold(
 const std::vector<float>& weights
 )
```

Parameters

weights
Label weights

ELocatorMetrics::GetBestWeightedPrecision

Best weighted precision achievable by changing the detection threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedPrecision(
 )
```

```
float GetBestWeightedPrecision(
 const std::vector<float>& weights
 )
```

Parameters

weights
Label weights

ELocatorMetrics::GetBestWeightedPrecisionAndThreshold

Best weighted precision achievable with the corresponding threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void GetBestWeightedPrecisionAndThreshold(
 float& precision,
 float& threshold
 )

void GetBestWeightedPrecisionAndThreshold(
 const std::vector<float>& weights,
 float& precision,
 float& threshold
 )
```

Parameters

precision
Best weighted precision achievable
threshold
Threshold that achieves the best weighted precision
weights
Label weights

ELocatorMetrics::GetBestWeightedPrecisionThreshold

Detection threshold that gives the best weighted precision (ELocatorMetrics).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedPrecisionThreshold(
 )

float GetBestWeightedPrecisionThreshold(
 const std::vector<float>& weights
 )
```

Parameters

weights
Label weights

ELocatorMetrics::GetBestWeightedRecall

Best weighted recall achievable by changing the detection threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetBestWeightedRecall(
 )
float GetBestWeightedRecall(
 const std::vector<float>& weights
 )
```

Parameters

weights
Label weights

ELocatorMetrics::GetBestWeightedRecallAndThreshold

Best weighted recall achievable with the corresponding threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void GetBestWeightedRecallAndThreshold(
 float& recall,
 float& threshold
 )
void GetBestWeightedRecallAndThreshold(
 const std::vector<float>& weights,
 float& recall,
 float& threshold
 )
```

Parameters

recall
Best weighted recall achievable
threshold
Threshold that achieves the best weighted recall
weights
Label weights

ELocatorMetrics::GetBestWeightedRecallThreshold

Detection threshold that gives the best weighted recall (ELocatorMetrics).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedRecallThreshold(
 )
float GetBestWeightedRecallThreshold(
 const std::vector<float>& weights
 )
```

Parameters

weights

Label weights

ELocatorMetrics::GetLabel

Label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetLabel(
 int idx
 )
```

Parameters

idx

Index of the label.

ELocatorMetrics::GetLabelAveragePrecision

Average precision for detection of objects from the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelAveragePrecision(
 int labelIdx
 )
```

Parameters

labelIdx

Label index

ELocatorMetrics::GetLabelAverageProximity

Average proximity of the predicted objects of the given label to the ground truth objects.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelAverageProximity(
 int labelIdx
 )
```

Parameters

labelIdx

Index of the label.

Remarks

Look at the documentation of ELocator and EInterestPointLocator for their definition of proximity.

ELocatorMetrics::GetLabelFScore

F-Score for the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelFScore(
 const std::string& label
 )

float GetLabelFScore(
 int labelId
 )
```

Parameters

label

Label

labelId

Label index

ELocatorMetrics::GetLabelIntersectionOverUnion

This method is deprecated.

Deprecated. Same as ELocatorMetrics::GetLabelAverageProximity.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelIntersectionOverUnion(
 int labelIdx
 )
```

Parameters

labelIdx

Index of the label.

ELocatorMetrics::GetLabelPrecision

Precision for the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelPrecision(
 const std::string& label
 )

float GetLabelPrecision(
 int labelId
 )
```

Parameters

label

Label

labelId

Label index

ELocatorMetrics::GetLabelRecall

Recall for the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetLabelRecall(
 const std::string& label
 )

float GetLabelRecall(
 int labelId
 )
```

Parameters

label
Label
labelId
Label index

ELocatorMetrics::GetNumCorrectlyDetectedObjects

Number of correctly detected objects.
A correctly detected object is a predicted object that has a "proximity" higher than ELocatorBase::SameLabelMaxObjectProximity with a ground truth object of the same label.
A label can be specified to obtain the number of correctly detected objects for that label. Otherwise, the number of correctly detected objects is for all the labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumCorrectlyDetectedObjects(
 )

OEV_UINT32 GetNumCorrectlyDetectedObjects(
 const std::string& label
 )

OEV_UINT32 GetNumCorrectlyDetectedObjects(
 int labelId
 )
```

Parameters

label
Label
labelId
Index of the label

ELocatorMetrics::GetNumDetectedObjects

Number of detected objects.
A label can be specified to obtain the number of detected objects for that label. Otherwise, the number of detected objects is for all the labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumDetectedObjects(
 )

OEV_UINT32 GetNumDetectedObjects(
 const std::string& label
 )
```

```
OEV_UINT32 GetNumDetectedObjects(
 int labelId
 )
```

Parameters

label
Label

labelId
Index of the label

ELocatorMetrics::GetNumUndetectedObjects

Number of undetected objects.
An undetected object is a ground truth object that is not matched to any predicted object of the same label with a proximity higher than ELocatorBase::SameLabelMaxObjectProximity.
A label can be specified to obtain the number of undetected objects for that label. Otherwise, the number of undetected objects is for all the labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumUndetectedObjects(
 )

OEV_UINT32 GetNumUndetectedObjects(
 const std::string& label
 )

OEV_UINT32 GetNumUndetectedObjects(
 int labelId
 )
```

Parameters

label
Label

labelId
Index of the label

ELocatorMetrics::GetWeightedFScore

Weighted average of the F-Score for each label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetWeightedFScore(
 )
```

```
float GetWeightedFScore(
 const std::vector<float>& weights
 )
```

Parameters

weights

Label weights

ELocatorMetrics::GetWeightedPrecision

Weighted average of the precision (proportion of detected objects that are correct) for each label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetWeightedPrecision(
 )
float GetWeightedPrecision(
 const std::vector<float>& weights
 )
```

Parameters

weights

Label weights

ELocatorMetrics::GetWeightedRecall

Weighted average of the recall (true positive rate) for each label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetWeightedRecall(
 )
float GetWeightedRecall(
 const std::vector<float>& weights
 )
```

Parameters

weights

Label weights

ELocatorMetrics::GetImageAccuracy

Image accuracy: the proportion of images that are correctly detected to contain or not objects, regardless of their labels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetImageAccuracy() const
```

ELocatorMetrics::GetIntersectionOverUnion

This property is deprecated.

Deprecated. Same as ELocatorMetrics::AverageProximity.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetIntersectionOverUnion() const
```

ELocatorMetrics::IsValid

Indicates whether the object contains at least one result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsValid(
 )
```

ELocatorMetrics::Load

Loads a locator metric. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ELocatorMetrics::GetNumBadlyPredictedImagesWithObjects

Number of images with objects that are badly predicted as containing no object. This is the number of false negatives at the image level.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetNumBadlyPredictedImagesWithObjects() const
```

ELocatorMetrics::GetNumBadlyPredictedImagesWithoutObjects

Number of images without objects that are badly predicted as containing objects. This is the number of false positives a the image level.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetNumBadlyPredictedImagesWithoutObjects() const
```

ELocatorMetrics::GetNumCorrectlyPredictedImagesWithObjects

Number of images containing objects that are correctly predicted as containing objects, regardless of the labels of the objects. This is the number of true positives at the image level.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT32 GetNumCorrectlyPredictedImagesWithObjects() const
```

ELocatorMetrics::GetNumCorrectlyPredictedImagesWithoutObjects

Number of images without objects that are correctly predicted as containing no object. This is the number of true negatives at the image level.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT32 GetNumCorrectlyPredictedImagesWithoutObjects() const
```

ELocatorMetrics::GetNumLabels

Number of labels recognized by the ELocator tool that produced these metrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumLabels() const
```

ELocatorMetrics::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ELocatorMetrics& operator=(
 const ELocatorMetrics& other
 )
```

Parameters

other

Reference to the ELocatorMetrics object used for the assignment

ELocatorMetrics::GetPrecision

Precision (proportion of detected objects that are correct).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetPrecision() const
```

ELocatorMetrics::GetRecall

Recall (true positive rate, proportion of ground truth object that are correctly detected).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetRecall() const
```

ELocatorMetrics::Save

Saves a locator metric. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.147. ELocatorObject Class

Object for a ELocator tool.

Derived Class(es):ELocatorPredictedObject

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Draw	Draws the object with its label.
ELocatorObject	Constructs a ELocatorObject.
GetFeatures	Features of this locator object.
GetHeight	Height of the object.
GetLabel	Label of the object.
GetObjectSize	Object size for EasyLocate Interest points. This property is set by a EClassificationDataset or EInterestPointLocator instance.
GetOrgX	Top-left corner X origin of the object.
GetOrgY	Top-left corner y origin of the object.

GetPositionX	X position of an object. When the object has a size, the position if the center of the bounding box.
GetPositionY	Y position of an object without size. When the object has a size, the position if the center of the bounding box.
GetRectangleRegion	Rectangle region for the object. The region must be axis-aligned. When the object has not the ELocatorFeature_Size feature, the width and height of the rectangle are equal to ELocatorObject property.
GetWidth	Width of the object.
HasFeature	Checks that the object has a locator feature.
IsValid	Whether the locator object is valid.
operator!=	Inequality operator.
operator=	Assignment operator
operator==	Equality operator.
SetHeight	Height of the object.
SetLabel	Label of the object.
SetOrgX	Top-left corner X origin of the object.
SetOrgY	Top-left corner y origin of the object.
SetOriginAndSize	Sets the position and size of an object using its top left origin, width, and height.
SetPositionX	X position of an object. When the object has a size, the position if the center of the bounding box.
SetPositionY	Y position of an object without size. When the object has a size, the position if the center of the bounding box.
SetRectangleRegion	Rectangle region for the object. The region must be axis-aligned. When the object has not the ELocatorFeature_Size feature, the width and height of the rectangle are equal to ELocatorObject property.
SetWidth	Width of the object.

ELocatorObject::Draw

Draws the object with its label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Draw(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 EDrawAdapter* graphicsContext,
 bool drawLabel,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicsContext,
 bool drawLabel,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

-

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawLabel

Whether to draw the label of the object or not

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELocatorObject::ELocatorObject

Constructs a ELocatorObject.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ELocatorObject(
 )

void ELocatorObject(
 float posX,
 float posY,
 const std::string& label
 )

void ELocatorObject(
 float orgX,
 float orgY,
 float width,
 float height,
 const std::string& label
 )

void ELocatorObject(
 const ELocatorObject& other
 )
```

Parameters

posX
X position of the object

posY
Y position of the object

label
Label of the object

orgX
X origin of the object

orgY
Y origin of the object

width
Width of the object

height
Height of the object

other
Reference to the ELocatorObject object that should be copied

ELocatorObject::GetFeatures

Features of this locator object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetFeatures() const
```

ELocatorObject::HasFeature

Checks that the object has a locator feature.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasFeature(
 Euresys::Open_eVision::EasyDeepLearning::ELocatorFeature feature
 )
```

Parameters

feature

Feature to check.

ELocatorObject::GetHeight

ELocatorObject::SetHeight

Height of the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetHeight() const
void SetHeight(float height)
```

ELocatorObject::IsValid

Whether the locator object is valid.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsValid(
 )
```

ELocatorObject::GetLabel

ELocatorObject::SetLabel

Label of the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetLabel() const
void SetLabel(const std::string& label)
```

ELocatorObject::GetObjectSize

Object size for EasyLocate Interest points. This property is set by a EClassificationDataset or EInterestPointLocator instance.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetObjectSize() const
```

ELocatorObject::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool operator!=(
 const ELocatorObject& other
 )
```

Parameters

other

Other object to compare to.

ELocatorObject::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ELocatorObject& operator=(
 const ELocatorObject& other
 )
```

Parameters

other

Reference to the ELocatorObject object used for the assignment

ELocatorObject::operator==

Equality operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool operator==(
 const ELocatorObject& other
 )
```

Parameters

other

Other object to compare to.

ELocatorObject::GetOrgX

ELocatorObject::SetOrgX

Top-left corner X origin of the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetOrgX() const
void SetOrgX(float orgX)
```

ELocatorObject::GetOrgY

ELocatorObject::SetOrgY

Top-left corner y origin of the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetOrgY() const
void SetOrgY(float orgY)
```

ELocatorObject::GetPositionX

ELocatorObject::SetPositionX

X position of an object. When the object has a size, the position if the center of the bounding box.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetPositionX() const
void SetPositionX(float posX)
```

ELocatorObject::GetPositionY

ELocatorObject::SetPositionY

Y position of an object without size. When the object has a size, the position if the center of the bounding box.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetPositionY() const
void SetPositionY(float posY)
```

ELocatorObject::GetRectangleRegion

ELocatorObject::SetRectangleRegion

Rectangle region for the object. The region must be axis-aligned.
When the object has not the ELocatorFeature_Size feature, the width and height of the rectangle are equal to ELocatorObject property.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ERectangleRegion GetRectangleRegion() const
void SetRectangleRegion(const ERectangleRegion& region)
```

ELocatorObject::SetOriginAndSize

Sets the position and size of an object using its top left origin, width, and height.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SetOriginAndSize(
 float orgX,
 float orgY,
 float width,
 float height
 )
```

Parameters

orgX
X origin of the object

orgY
Y origin of the object

width
Width of the object

height
Height of the object

ELocatorObject::GetWidth

ELocatorObject::SetWidth

Width of the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetWidth() const
void SetWidth(float width)
```

4.148. ELocatorPredictedObject Class

Object predicted by a ELocator tool.

Base Class:ELocatorObject
Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Draw	Draws the object with its label and probability.
ELocatorPredictedObject	Copy constructor.
GetProbability	Probability of the object.
operator=	Assignment operator.

ELocatorPredictedObject::Draw

Draws the object with its label and probability.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Draw(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicsContext,
 bool drawLabel,
 bool drawProbability,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicsContext,
 bool drawLabel,
 bool drawProbability,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

-

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawLabel

Whether to draw the label of the object or not

drawProbability

Whether to draw the probability of the object or not

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELocatorPredictedObject::ELocatorPredictedObject

Copy constructor.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void ELocatorPredictedObject(
 const ELocatorPredictedObject& other
 )
```

Parameters

other

Object to copy

ELocatorPredictedObject::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ELocatorPredictedObject& operator=(
 const ELocatorPredictedObject& other
 )
```

Parameters

other

Object to copy

ELocatorPredictedObject::GetProbability

Probability of the object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetProbability() const
```

4.149. ELocatorResult Class

Result of a ELocator tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Draw	Draws the detected objects with their label and score.
ELocatorResult	Constructs an ELocatorResult object.
GetDetectedObjects	Detected objects.
GetDetectionThreshold	Detection threshold set by the ELocator tool.
GetGroundtruthObjects	Groundtruth objects if ELocatorResult::HasGroundtruth is equal to true.
GetLabel	i-th label recognized by the ELocator.
GetLabelColor	Color of a label.
GetLocatorFeatures	Locator features of the result.
GetNumDetectedObjects	Number of detected objects in the image.
GetNumLabels	Number of labels the ELocator can recognize.
GetObjectSize	Object size for prediction made by a EInterestPointLocator.
HasGroundtruth	Whether the result has a groundtruth.
IsValid	Whether the result is valid and was produced by a ELocator objet.

Load	Loads a locator result. The given ESerializer must have been created for reading.
operator=	Assignement operator.
RemoveGroundtruth	Removes the groundtruth.
Save	Saves a locator result. The given ESerializer must have been created for writing.

ELocatorResult::GetDetectedObjects

Detected objects.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorPredictedObject>
GetDetectedObjects() const
```

ELocatorResult::GetDetectionThreshold

Detection threshold set by the ELocator tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetDetectionThreshold() const
```

ELocatorResult::Draw

Draws the detected objects with their label and score.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Draw(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 EDrawAdapter* graphicsContext,
 bool drawLabel,
 bool drawProbability,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicsContext,
 bool drawLabel,
 bool drawProbability,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

-

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning factor, in pixels. By default, no panning occurs.

panY

Vertical panning factor, in pixels. By default, no panning occurs.

drawLabel

Whether to draw the label of each detected object or not

drawProbability

Whether to draw the probability of each detected object or not

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ELocatorResult::ELocatorResult

Constructs an ELocatorResult object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void ELocatorResult(
 )
void ELocatorResult(
 const ELocatorResult& other
 )
```

Parameters

other

ELocatorResult object

ELocatorResult::GetLabel

i-th label recognized by the ELocator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetLabel(
 int i
 )
```

Parameters

i

Label index between 0 and ELocatorResult::NumLabels

ELocatorResult::GetLabelColor

Color of a label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ERGBColor GetLabelColor(
 int i
 )
ERGBColor GetLabelColor(
 const std::string& label
 )
```

Parameters

i

Index of the label for which to get the color (between 0 and ELocatorResult::NumLabels - 1)

label

Label for which to get the color

Remarks

The label color is controlled at the tool level. To change a color in a result, change the label color in the tool.

ELocatorResult::GetNumDetectedObjects

Number of detected objects in the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumDetectedObjects(
 )

int GetNumDetectedObjects(
 const std::string& label
 )

int GetNumDetectedObjects(
 int labelId
 )
```

Parameters

label

Label for which to count the detected objects

labelId

Index of label for which to count the detected objects

ELocatorResult::GetGroundtruthObjects

Groundtruth objects if ELocatorResult::HasGroundtruth is equal to true.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::vector<Euresys::Open_eVision::EasyDeepLearning::ELocatorObject>
GetGroundtruthObjects() const
```

ELocatorResult::HasGroundtruth

Whether the result has a groundtruth.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool HasGroundtruth(
 )
```

ELocatorResult::IsValid

Whether the result is valid and was produced by a ELocator objet.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsValid(
 )
```

ELocatorResult::Load

Loads a locator result. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ELocatorResult::GetLocatorFeatures

Locator features of the result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetLocatorFeatures() const
```

ELocatorResult::GetNumLabels

Number of labels the ELocator can recognize.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetNumLabels() const
```

ELocatorResult::GetObjectSize

Object size for prediction made by a EInterestPointLocator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int GetObjectSize() const
```

ELocatorResult::operator=

Assignement operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ELocatorResult& operator=(
 const ELocatorResult& other
 )
```

Parameters

other

ELocatorResult object.

ELocatorResult::RemoveGroundtruth

Removes the groundtruth.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void RemoveGroundtruth(
 )
```

ELocatorResult::Save

Saves a locator result. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

4.150. EMailBarcode Class

Manages a complete context for a Mail Barcode.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the bounding box of the mail barcode.
EMailBarcode	Constructs an EMailBarcodeReader context.
GetChecksumOk	Returns if the checksum was successfully validated.
GetComponentStrings	Returns the list of semantic parts included in the mail barcode text.
GetOrientation	Returns the orientation of the mail barcode.
GetPosition	Returns the position of the mail barcode in the Image/ROI.
GetSymbology	Returns the symbology of the mail barcode.
GetText	Returns the full decoded text of the mail barcode.
operator=	Assignment operator

EMailBarcode::GetChecksumOk

Returns if the checksum was successfully validated.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetChecksumOk() const
```

EMailBarcode::GetComponentStrings

Returns the list of semantic parts included in the mail barcode text.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EStringPair> GetComponentStrings() const
```

EMailBarcode::Draw

Draws the bounding box of the mail barcode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* adapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

adapter

-

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

hDC

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMailBarcode::EMailBarcode

Constructs an EMailBarcodeReader context.

Namespace: Euresys::Open_eVision

```
[C++]
void EMailBarcode(
 const EMailBarcode& other
 )
```

Parameters

other

-

EMailBarcode::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
EMailBarcode& operator=(
 const EMailBarcode& other
 )
```

Parameters

other

-

EMailBarcode::GetOrientation

Returns the orientation of the mail barcode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EMailBarcodeOrientation GetOrientation() const
```

EMailBarcode::GetPosition

Returns the position of the mail barcode in the Image/ROI.

Namespace: Euresys::Open_eVision

[C++]

```
ERectangle GetPosition() const
```

EMailBarcode::GetSymbology

Returns the symbology of the mail barcode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EMailBarcodeSymbologies GetSymbology() const
```

EMailBarcode::GetText

Returns the full decoded text of the mail barcode.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetText() const
```

4.151. EMailBarcodeReader Class

Manages a complete context for a Mail Barcode Reader.

Namespace: Euresys::Open_eVision

Methods

EMailBarcodeReader	Constructs an EMailBarcodeReader context.
GetEnableClutteredBarcodes	Enables cluttered barcode (barcode with fused bars) support.
GetEnableDottedBarcodes	Enables dotted barcode support.
GetExpectedOrientations	Expected barcode orientations for the Mail Barcode detection.
GetExpectedSymbologies	Expected symbologies for the Mail Barcode detection.
GetValidateChecksum	Configures the reader to return barcodes even if their checksum is incorrect.
Load	Loads the settings of the EMailBarcodeReader object, from disk.
operator=	Assignment operator
Read	Locates and decodes mail barcodes.
Save	Saves the current settings of the EMailBarcodeReader object.
SetEnableClutteredBarcodes	Enables cluttered barcode (barcode with fused bars) support.
SetEnableDottedBarcodes	Enables dotted barcode support.
SetExpectedOrientations	Expected barcode orientations for the Mail Barcode detection.
SetExpectedSymbologies	Expected symbologies for the Mail Barcode detection.
SetValidateChecksum	Configures the reader to return barcodes even if their checksum is incorrect.

EMailBarcodeReader::EMailBarcodeReader

Constructs an EMailBarcodeReader context.

Namespace: Euresys::Open_eVision

[C++]

```
void EMailBarcodeReader(
 )
```

```
void EMailBarcodeReader(
 const EMailBarcodeReader& other
 )
```

Parameters

other

-

EMailBarcodeReader::GetEnableClutteredBarcodes

EMailBarcodeReader::SetEnableClutteredBarcodes

Enables cluttered barcode (barcode with fused bars) support.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableClutteredBarcodes() const
void SetEnableClutteredBarcodes(bool enable)
```

EMailBarcodeReader::GetEnableDottedBarcodes

EMailBarcodeReader::SetEnableDottedBarcodes

Enables dotted barcode support.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableDottedBarcodes() const
void SetEnableDottedBarcodes(bool enable)
```

EMailBarcodeReader::GetExpectedOrientations

EMailBarcodeReader::SetExpectedOrientations

Expected barcode orientations for the Mail Barcode detection.

Namespace: Euresys::Open_eVision

[C++]

```
int GetExpectedOrientations() const
```

```
void SetExpectedOrientations(int orientations)
```

Remarks

The value is a combination of the members of the EMailBarcodeOrientation enumerate.

EMailBarcodeReader::GetExpectedSymbologies

EMailBarcodeReader::SetExpectedSymbologies

Expected symbologies for the Mail Barcode detection.

Namespace: Euresys::Open_eVision

[C++]

```
int GetExpectedSymbologies() const
void SetExpectedSymbologies(int symbologies)
```

Remarks

The value is a combination of the members of the EMailBarcodeSymbologies enumerate.

EMailBarcodeReader::Load

Loads the settings of the EMailBarcodeReader object, from disk.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 ESerializer* serializer
 )
void Load(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

A string containing the full path to the file.

EMailBarcodeReader::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
EMailBarcodeReader& operator=(
 const EMailBarcodeReader& other
 )
```

Parameters

other

-

EMailBarcodeReader::Read

Locates and decodes mail barcodes.

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::EMailBarcode> Read(
 const EROIBW8& roi
 )
```

Parameters

roi

The ROI/Image in which to search for mail barcodes.

Remarks

This method returns the list of the detected barcodes.

EMailBarcodeReader::Save

Saves the current settings of the EMailBarcodeReader object.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 ESerializer* serializer
 )
void Save(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

A string containing the full path to the file.

Remarks

It is advised to use a file extension that is non-standard (for instance *.mbr).

EMailBarcodeReader::GetValidateChecksum

EMailBarcodeReader::SetValidateChecksum

Configures the reader to return barcodes even if their checksum is incorrect.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetValidateChecksum() const
void SetValidateChecksum(bool validateChecksum)
```

4.152. EMatcher Class

Manages a complete matching context in EasyMatch.

Remarks

A matching context consists of a learned pattern and of the parameters required to locate one or more instances of the pattern in a search field.

Namespace: Euresys::Open_eVision

Methods

ClearImage	Releases the pointer to the image object that has been passed to the EMatcher object.
CopyLearntPattern	Copies the learned pattern in the supplied image. If no pattern has been learned, an exception with code EError_NoPatternLearnt will be thrown.
CopyTo	Copies all the data of the current EMatcher object into another EMatcher object and returns it.
DrawPosition	Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

DrawPositions	Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
DrawPositionsWithCurrentPen	Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
DrawPositionWithCurrentPen	Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.
EMatcher	Constructs a matching context.
GetAdvancedLearning	Toggle advanced learning.
GetAngleStep	Gets the angle step used at the given reduction.
GetContrastMode	Contrast mode.
GetCorrelationMode	Correlation mode.
GetDontCareThreshold	"Don't care" threshold.
GetEnableEarlyCandidateRejection	Whether to remove bad candidates in the early phases of the processing or not. Enabled by default.
GetExtension	Gets the extension of the matching ROI, i.e. the horizontal and vertical distances, in pixels, that the found pattern occurrences may fall outside the matching ROI. Such occurrences partially outside the ROI have their score corrected by the ratio between the pattern area outside the ROI and the pattern area inside.
GetFilteringMode	Filtering mode.
GetFinalReduction	Index of the last reduction.
GetInitialMinScore	Minimum score applied as a selection criterion in the early stages of the matching process.
GetInterpolate	Interpolation mode.
GetIsotropicScale	Flag indicating whether isotropic (as opposed to anisotropic) scaling is used.
GetMaxAngle	Maximum angle, in the current angle unit.
GetMaxInitialPositions	Maximum number of positions at the first stage of the matching process.
GetMaxOverlap	Overlapping tolerance
GetMaxPositions	Maximum number of positions.
GetMaxScale	Maximum scale factor for isotropic scaling.
GetMaxScaleX	Maximum horizontal scale factor for anisotropic scaling.
GetMaxScaleY	Maximum vertical scale factor for anisotropic scaling.
GetMinAngle	Minimum angle, in the current angle unit.
GetMinReducedArea	Minimum reduced area parameter.
GetMinScale	Minimum scale factor for isotropic scaling.

GetMinScaleX	Minimum horizontal scale factor for anisotropic scaling.
GetMinScaleY	Minimum vertical scale factor for anisotropic scaling.
GetMinScore	Minimum score.
GetNumPositions	Number of good matches found, as defined by EMatcher::MinScore and EMatcher::MaxPositions properties.
GetNumReductions	Number of reduction steps used in the matching process.
GetPatternHeight	Learned pattern height.
GetPatternLearnt	Returns true after a learning operation has been successfully performed, indicating that the EMatcher object is ready for matching, and false otherwise.
GetPatternType	Pixel type of the learned pattern.
GetPatternWidth	Learned pattern width.
GetPixelDimensions	Gets the physical pixel dimensions.
GetPosition	Returns an EMatchPosition object containing the position coordinates.
GetPositions	Returns a vector of EMatchPosition objects, each containing the position coordinates and other matching results.
GetScaleStep	Gets the scale step used at the given reduction.
GetScaleXStep	Gets the scale step used on the X axis at the given reduction.
GetScaleYStep	Gets the scale step used on the Y axis at the given reduction.
GetVersion	Version number of the EMatcher object.
LearnPattern	Learns a pattern to subsequently match in an image.
Load	Loads the EMatcher. The given ESerializer must have been created for reading.
Match	Matches the pattern against an image.
operator=	Copies all the data from another EMatcher object into the current EMatcher object
Save	Saves the EMatcher. The given ESerializer must have been created for writing.
SetAdvancedLearning	Toggle advanced learning.
SetContrastMode	Contrast mode.
SetCorrelationMode	Correlation mode.
SetDontCareThreshold	"Don't care" threshold.
SetEnableEarlyCandidateRejection	Whether to remove bad candidates in the early phases of the processing or not. Enabled by default.
SetExtension	Sets the extension of the matching ROI, i.e. puts the horizontal and vertical distances, in pixels, that the found pattern occurrences may fall outside the matching ROI. Such occurrences partially outside the ROI have their score corrected by the ratio between the pattern area outside the ROI and the pattern area inside.
SetFilteringMode	Filtering mode.

SetFinalReduction	Index of the last reduction.
SetInitialMinScore	Minimum score applied as a selection criterion in the early stages of the matching process.
SetInterpolate	Interpolation mode.
SetMaxAngle	Maximum angle, in the current angle unit.
SetMaxInitialPositions	Maximum number of positions at the first stage of the matching process.
SetMaxOverlap	Overlapping tolerance
SetMaxPositions	Maximum number of positions.
SetMaxScale	Maximum scale factor for isotropic scaling.
SetMaxScaleX	Maximum horizontal scale factor for anisotropic scaling.
SetMaxScaleY	Maximum vertical scale factor for anisotropic scaling.
SetMinAngle	Minimum angle, in the current angle unit.
SetMinReducedArea	Minimum reduced area parameter.
SetMinScale	Minimum scale factor for isotropic scaling.
SetMinScaleX	Minimum horizontal scale factor for anisotropic scaling.
SetMinScaleY	Minimum vertical scale factor for anisotropic scaling.
SetMinScore	Minimum score.
SetPixelDimensions	Sets the physical pixel dimensions.

EMatcher::GetAdvancedLearning

EMatcher::SetAdvancedLearning

Toggle advanced learning.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetAdvancedLearning() const
void SetAdvancedLearning(bool bState)
```

Remarks

When enabled, the advanced learning process will try to optimize learning parameters like the Minimum Reduced Area. Advanced learning uses the whole image context (parent EImage for an EROI) to optimize the parameters. The learning will take more time (from 1x to 5x longer) but the matching probability could be improved. The improvement strongly depends on the pattern source image. The advanced learning is automatically disabled when the method EMatcher::MinReducedArea is called.
In addition, advanced learning does not work when using EMatcher::DontCareThreshold and a masked image. However, it is compatible with the overload of EMatcher::LearnPattern taking an ERegion, which allows to do the same as EMatcher::DontCareThreshold.

EMatcher::ClearImage

Releases the pointer to the image object that has been passed to the EMatcher object.

Namespace: Euresys::Open_eVision

[C++]

```
void ClearImage(
 )
```

Remarks

It is the way to tell to the EMatcher object that its pointer is not valid anymore. The EMatcher::Match method keeps a copy of the image pointer given as parameter. So, if the user deletes this pointer, the EMatcher object should be informed.

EMatcher::GetContrastMode

EMatcher::SetContrastMode

Contrast mode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EMatchContrastMode GetContrastMode() const
void SetContrastMode(Euresys::Open_eVision::EMatchContrastMode eMode)
```

Remarks

By default, the contrast mode is set to EMatchContrastMode_Normal.

EMatcher::CopyLearntPattern

Copies the learned pattern in the supplied image. If no pattern has been learned, an exception with code NoPatternLearnt will be thrown.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyLearntPattern(
 EImageBW8& image
 )
void CopyLearntPattern(
 EImageC24& image
 )
```

Parameters

image

Pointer to the image in which the learned pattern will be returned.

EMatcher::CopyTo

Copies all the data of the current EMatcher object into another EMatcher object and returns it.

Namespace: Euresys::Open_eVision

```
[C++]
void CopyTo(
 EMatcher& other
 )
```

Parameters

other

Reference to the EMatcher object in which the current EMatcher object parameters are to be copied.

EMatcher::GetCorrelationMode

EMatcher::SetCorrelationMode

Correlation mode.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::ECorrelationMode GetCorrelationMode() const
void SetCorrelationMode(Euresys::Open_eVision::ECorrelationMode eMode)
```

Remarks

This property tells what normalization rule is used to correlate the pattern to the image. By default, the correlation mode is set to ECorrelationMode_Normalized.

EMatcher::GetDontCareThreshold

EMatcher::SetDontCareThreshold

"Don't care" threshold.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetDontCareThreshold() const

void SetDontCareThreshold(OEV_UINT32 un32Threshold)
```

Remarks

If the pattern cannot be inscribed in a rectangle because there are foreign objects in a close neighborhood, mismatches can be avoided by using "don't care" pixels: all the pattern pixels whose value is strictly below DontCareThreshold will be ignored. By default, this property is set to 0: no "don't care" pixel exists.

Note. When you use the "don't care" feature, either the pattern is well contrasted from its background -then set DontCareThreshold to an appropriate thresholding value- or it is not -then set the background pixels of the pattern to some low value (by a masking operation) and set DontCareThreshold to this low value plus one.

In addition, EMatcher::AdvancedLearning does not work when using a don't care threshold and a masked image. However, it is compatible with the overload of EMatcher::LearnPattern taking an ERegion, which allows to do the same as EMatcher::DontCareThreshold.

EMatcher::DrawPosition

Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPosition(
 EDrawAdapter* graphicContext,
 OEV_UINT32 index,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawPosition(
 HDC graphicContext,
 OEV_UINT32 index,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawPosition(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 index,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

index

Occurrence index, in range 0..NumPositions-1.

bCorner

true if the corner mark is to be drawn. false by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatcher::DrawPositions

Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPositions(
 EDrawAdapter* graphicContext,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawPositions(
 HDC graphicContext,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawPositions(
 HDC graphicContext,
 const ERGBColor& color,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

bCorner

true if the corner mark is to be drawn. false by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all occurrences (to vary the colors, draw the objects separately using the EMatcher::DrawPosition method instead). Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatcher::DrawPositionsWithCurrentPen

This method is deprecated.

Draws a graphical representation of all occurrences of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

Namespace: Euresys::Open_eVision

```
[C++]

void DrawPositionsWithCurrentPen(
 HDC graphicContext,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

bCorner

true if the corner mark is to be drawn. false by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all occurrences (to vary the colors, draw the objects separately using the EMatcher::DrawPosition method instead). Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatcher::DrawPositionWithCurrentPen

This method is deprecated.

Draws a graphical representation of a given occurrence of the pattern in the image, using a rectangle and possibly a small line segment in the upper-left corner.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawPositionWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 index,
 bool bCorner,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

index

Occurrence index, in range 0..NumPositions-1.

bCorner

true if the corner mark is to be drawn. false by default. (This mark is useful when large rotations are allowed.)

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatcher::EMatcher

Constructs a matching context.

Namespace: Euresys::Open_eVision

[C++]

```
void EMatcher(
 )
void EMatcher(
 OEV_UINT32 maxNumDOF
 )
void EMatcher(
 const EMatcher& other
 )
```

Parameters

maxNumDOF

Maximum number of degrees of freedom (this number must be comprised between 2 and 5).

other

Another EMatcher object to be copied in the new EMatcher object.

Remarks

With the default constructor (no argument), all parameters are initialized to their respective default values. The copy constructor constructs a matching context based on a pre-existing EMatcher object. The last constructor constructs a matching context with a specified number of degrees of freedom. maximumNumberOfDegreesOfFreedom is the maximum number of degrees of freedom that the EMatcher object being constructed will be allowed to use during its life. All other parameters are initialized to their respective default values. The degrees of freedom for a matching operation are the *translation* (2 modes), the *rotation* (1 mode), the *isotropic scaling* (1 mode) and the *anisotropic scaling* (1 more mode). There is no way to modify this parameter for an existing EMatcher object. The default value for maximumNumberOfDegreesOfFreedom is 5, that is the maximum value. The minimum value for the number of degrees of freedom is 2, in order to allow, at least, the pattern translation.

EMatcher::GetEnableEarlyCandidateRejection

EMatcher::SetEnableEarlyCandidateRejection

Whether to remove bad candidates in the early phases of the processing or not. Enabled by default.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableEarlyCandidateRejection() const
void SetEnableEarlyCandidateRejection(bool earlyCandidateRejection)
```

Remarks

To reduce the number of candidates, those failing to fill some criterion are removed. This makes processing faster, a downside is that some good candidates can be erroneously removed. We recommend setting this parameter to false if you observe that a slight change in the input conditions make processing fail. For example: changing the search ROI in the image slightly.
After disabling the early candidate rejection, you will probably have to increase EMatcher::MaxInitialPositions as more candidates will now be kept.

EMatcher::GetFilteringMode

EMatcher::SetFilteringMode

Filtering mode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EFilteringMode GetFilteringMode() const
void SetFilteringMode(Euresys::Open_eVision::EFilteringMode eFilteringMode)
```

Remarks

To achieve acceptable time performance, EasyMatch works by sub-sampling the pattern in the early phases of the processing. The filtering mode parameter allows to select the pre-processing type applied to the image before the decimation: averaging or low-pass filtering. By default, this property is set to EFilteringMode_Uniform, whereas the EFilteringMode_LowPass mode is indicated if the image presents sharp gray-level transitions.

EMatcher::GetFinalReduction

EMatcher::SetFinalReduction

Index of the last reduction.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetFinalReduction() const
void SetFinalReduction(OEV_UINT32 un32FinalReduction)
```

Remarks

The pattern matching process is comprised of a few passes (typically 4) during which the position accuracy is improved by a factor of 2 (for all degrees of freedom). This is called a *reduction*. By default, the computation continues until an accuracy of one pixel is obtained. To speed up the process, the last reduction passes can be dropped, so lowering the accuracy. Anyway, the interpolation mode can then still be used. By default, this property is set to 0 (pixel accuracy). Value 1 corresponds to 2-pixels accuracy, 2 to 4, and so on. The range of values that can be used for this property is 0 to NumReductions-1.

EMatcher::GetAngleStep

Gets the angle step used at the given reduction.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngleStep(
 OEV_UINT32 reduction
 )
```

Parameters

reduction

Offset of the reduction in the pyramid, must be between EMatcher::FinalReduction and EMatcher::NumReductions - 1.

EMatcher::GetExtension

Gets the extension of the matching ROI, i.e. the horizontal and vertical distances, in pixels, that the found pattern occurrences may fall outside the matching ROI. Such occurrences partially outside the ROI have their score corrected by the ratio between the pattern area outside the ROI and the pattern area inside.

Namespace: Euresys::Open_eVision

[C++]

```
void GetExtension(
 int& n32ExtensionX,
 int& n32ExtensionY
 )
```

Parameters

n32ExtensionX

extension outside the matching ROI along its Width, in pixels.

n32ExtensionY

extension outside the matching ROI along its Height, in pixels.

EMatcher::GetPixelDimensions

Gets the physical pixel dimensions.

Namespace: Euresys::Open_eVision

```
[C++]
void GetPixelDimensions(
 float& width,
 float& height
 )
```

Parameters

width
Width of a pixel.

height
Height of a pixel.

EMatcher::GetPosition

Returns an EMatchPosition object containing the position coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
EMatchPosition GetPosition(
 OEV_UINT32 index
 )
```

Parameters

index
0-based index to the desired position. The positions are ordered by decreasing score.

EMatcher::GetScaleStep

Gets the scale step used at the given reduction.

Namespace: Euresys::Open_eVision

```
[C++]
float GetScaleStep(
 OEV_UINT32 reduction
 )
```

Parameters

reduction

Offset of the reduction in the pyramid, must be between EMatcher::FinalReduction and EMatcher::NumReductions - 1.

EMatcher::GetScaleXStep

Gets the scale step used on the X axis at the given reduction.

Namespace: Euresys::Open_eVision

```
[C++]
float GetScaleXStep(
 OEV_UINT32 reduction
 )
```

Parameters

reduction

Offset of the reduction in the pyramid, must be between EMatcher::FinalReduction and EMatcher::NumReductions - 1.

EMatcher::GetScaleYStep

Gets the scale step used on the Y axis at the given reduction.

Namespace: Euresys::Open_eVision

```
[C++]
float GetScaleYStep(
 OEV_UINT32 reduction
 )
```

Parameters

reduction

Offset of the reduction in the pyramid, must be between EMatcher::FinalReduction and EMatcher::NumReductions - 1.

EMatcher::GetInitialMinScore

EMatcher::SetInitialMinScore

Minimum score applied as a selection criterion in the early stages of the matching process.

Namespace: Euresys::Open_eVision

[C++]

```
float GetInitialMinScore() const
void SetInitialMinScore(float f32InitialMinScore)
```

Remarks

When the search for matches starts, EasyMatch considers a set of candidate positions that it progressively refines. When they later appear to be bad candidates, they are rejected. Though it is the minimum score level that is used to reject bad matching positions at the final step of the matching process, the "initial minimum score" parameter is used to eliminate bad positions (whose score is not high enough) in the early stages of the matching processing. If the matching process is achieved in one step, only the minimum score parameter will be considered. By default, this property is set to -1.

EMatcher::GetInterpolate

EMatcher::SetInterpolate

Interpolation mode.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetInterpolate() const
void SetInterpolate(bool bInterpolate)
```

Remarks

By default, matching is done with a one-pixel precision for all degrees of freedom (translation, rotation and scaling). You can use an additional interpolation process to achieve sub-pixel accuracy. This generally leads to an improvement of the sub-pixel accuracy by a factor larger than 10. This is possible only when the found instances match closely the model. A score higher than 0.99 indicates that the instances are a close match of the model. In other words, the instance is considered to be more accurate when the score is higher. The added computational cost is low. By default, this property is set to false.

EMatcher::GetIsotropicScale

Flag indicating whether isotropic (as opposed to anisotropic) scaling is used.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetIsotropicScale() const
```

Remarks

true if isotropic (as opposed to anisotropic) scaling is used, i.e. when the scale factors in both the X and Y directions are equal.

EMatcher::LearnPattern

Learns a pattern to subsequently match in an image.

Namespace: Euresys::Open_eVision

[C++]

```
void LearnPattern(
 EROIBW8* pattern
 )

void LearnPattern(
 EROIC24* pattern
 )

void LearnPattern(
 EROIBW8* pattern,
 const ERegion& region
 )

void LearnPattern(
 EROIC24* pattern,
 const ERegion& region
 )
```

Parameters

pattern

Pattern to learn.

region

Region in the pattern image to consider

Remarks

The maximum size for a pattern is (sqrt(Minimum Reduced Area) - 1) * 2^8. For the default Minimum Reduced Area of 64, this corresponds to a 1792x1792 maximum size. Increasing the Minimum Reduced Area enables larger pattern size but reduces the processing speed.
When a region is given, the method will ignore the pixels outside the region by internally setting them to 0 and setting EMatcher::DontCareThreshold to 1 if it wasn't set by the user. Thus, when working with an ERegion, all pixels that are black in the pattern will be discarded.

EMatcher::Load

Loads the EMatcher. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )
```

```
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EMatcher::Match

Matches the pattern against an image.

Namespace: Euresys::Open_eVision

[C++]

```
void Match(
 EROIBW8* image
 )

void Match(
 EROIBW8* image,
 const ERegion& region
 )

void Match(
 EROIC24* image
 )

void Match(
 EROIC24* image,
 const ERegion& region
 )
```

Parameters

image

Pointer to the image/ROI within which the pattern will be searched for.

region

Region within which the pattern will be searched for.

Remarks

The matching results can be obtained by means of the EMatcher::NumPositions and EMatcher::GetPosition members.

EMatcher::GetMaxAngle

EMatcher::SetMaxAngle

Maximum angle, in the current angle unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxAngle() const
void SetMaxAngle(float f32MaxAngle)
```

Remarks

The rotation of the pattern is allowed within the range (-1 <= MinAngle < MaxAngle <= 1 revolution). By default, both remain 0.

EMatcher::GetMaxInitialPositions

EMatcher::SetMaxInitialPositions

Maximum number of positions at the first stage of the matching process.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMaxInitialPositions() const
void SetMaxInitialPositions(OEV_UINT32 un32MaxInitialPositions)
```

Remarks

When the search for matches starts, EasyMatch considers a set of candidate positions that it progressively refines. When they later appear to be bad candidates, they are rejected. Eventually, a maximum of EMatcher::MaxPositions is returned. In some circumstances, when the image contains features roughly similar to the pattern, these can confuse the matching process, resulting in false matches. To overcome this situation, increasing the number of initial positions will help. By default, this property is set to 0, indicating that the value of EMatcher::MaxPositions should be used instead.

EMatcher::GetMaxOverlap

EMatcher::SetMaxOverlap

Overlapping tolerance

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxOverlap() const
void SetMaxOverlap(float f32MaxOverlap)
```

Remarks

0.0 means all found patterns must be disconnected, 1.0 means they can fully overlap. Default: 1.0.

EMatcher::GetMaxPositions

EMatcher::SetMaxPositions

Maximum number of positions.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMaxPositions() const
void SetMaxPositions(OEV_UINT32 un32MaxPositions)
```

Remarks

Indicates how many matching positions have to be returned at a maximum. This number corresponds to the expected maximum number of occurrences of the pattern (it is sometimes advisable to use a few additional positions). By default, this property is set to 1.

EMatcher::GetMaxScale

EMatcher::SetMaxScale

Maximum scale factor for isotropic scaling.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxScale() const
void SetMaxScale(float f32Scale)
```

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant Set member. The scaling of the pattern is allowed within the range (0.5 <= MinScale < MaxScale <= 2). By default, both remain 1. The same holds for anisotropic scale factors.

EMatcher::GetMaxScaleX

EMatcher::SetMaxScaleX

Maximum horizontal scale factor for anisotropic scaling.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxScaleX() const
void SetMaxScaleX(float f32MaxScaleX)
```

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant Set member. The scaling of the pattern is allowed within the range (0.5 <= MinScaleX < MaxScaleX <= 2). By default, both remain 1. The same holds for anisotropic scale factors.

EMatcher::GetMaxScaleY

EMatcher::SetMaxScaleY

Maximum vertical scale factor for anisotropic scaling.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxScaleY() const
void SetMaxScaleY(float f32MaxScaleY)
```

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant Set member. The scaling of the pattern is allowed within the range (0.5 <= MinScaleY < MaxScaleY <= 2). By default, both remain 1. The same holds for anisotropic scale factors.

EMatcher::GetMinAngle

EMatcher::SetMinAngle

Minimum angle, in the current angle unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinAngle() const
void SetMinAngle(float f32MinAngle)
```

Remarks

The rotation of the pattern is allowed within the range (-1 <= MinAngle < MaxAngle <= 1 revolution). By default, both remain 0.

EMatcher::GetMinReducedArea

EMatcher::SetMinReducedArea

Minimum reduced area parameter.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinReducedArea() const
void SetMinReducedArea(OEV_UINT32 un32Area)
```

Remarks

To achieve acceptable time performance, EasyMatch works by under-sampling the pattern in the early phases of the processing. This property tells how many pixels of the pattern are kept, at a minimum. By default, this property is set to 64, which is the right choice in most situations. Higher values are not recommended. Lower values can speed up the processing, but sometimes cause the matching process fail to find the best matches. To circumvent this problem, you can use guard positions, that is find more matches than expected and keep the good ones only.

Note. Changing this property invalidates any previous learning.

EMatcher::GetMinScale

EMatcher::SetMinScale

Minimum scale factor for isotropic scaling.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinScale() const
void SetMinScale(float f32Scale)
```

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant Set member. The scaling of the pattern is allowed within the range (0.5 <= MinScale < MaxScale <= 2). By default, both remain 1. The same holds for anisotropic scale factors.

EMatcher::GetMinScaleX

EMatcher::SetMinScaleX

Minimum horizontal scale factor for anisotropic scaling.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinScaleX() const

void SetMinScaleX(float f32MinScaleX)
```

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant Set member. The scaling of the pattern is allowed within the range (0.5 <= MinScaleX < MaxScaleX <= 2). By default, both remain 1. The same holds for anisotropic scale factors.

EMatcher::GetMinScaleY

EMatcher::SetMinScaleY

Minimum vertical scale factor for anisotropic scaling.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinScaleY() const

void SetMinScaleY(float f32MinScaleY)
```

Remarks

Two scaling modes are allowed: in isotropic mode, the scale factor is identical in all directions; in anisotropic mode, the scale factors differ in the horizontal and vertical directions. To select the appropriate mode, it suffices to call the relevant Set member. The scaling of the pattern is allowed within the range (0.5 <= MinScaleY < MaxScaleY <= 2). By default, both remain 1. The same holds for anisotropic scale factors.

EMatcher::GetMinScore

EMatcher::SetMinScore

Minimum score.

Namespace: Euresys::Open_eVision

```
[C++]
float GetMinScore() const
void SetMinScore(float f32MinScore)
```

Remarks

This property indicates what score a match must reach to be considered as good, and to be returned in the list of positions; this selection criterion is applied at the final stage of the matching process. One good way to select the appropriate EMatcher::MinScore in a given context is to set it to -1 (any match will be retained), set EMatcher::MaxPositions to more than needed, and examine the returned scores after a matching. By default, this property is set to -1.

EMatcher::GetNumPositions

Number of good matches found, as defined by EMatcher::MinScore and EMatcher::MaxPositions properties.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumPositions() const
```

EMatcher::GetNumReductions

Number of reduction steps used in the matching process.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumReductions() const
```

Remarks

These depend on the actual pattern size, and on the MinReducedArea property. The FinalReduction property, used to speed up matching when coarse location is sufficient, must be set in range 0..NumReductions-1.

EMatcher::operator=

Copies all the data from another EMatcher object into the current EMatcher object

Namespace: Euresys::Open_eVision

[C++]

```
EMatcher& operator=(
 const EMatcher& other
 )
```

Parameters

other

EMatcher object to be copied

EMatcher::GetPatternHeight

Learned pattern height.

Namespace: Euresys::Open_eVision

[C++]

```
int GetPatternHeight() const
```

EMatcher::GetPatternLearnt

Returns true after a learning operation has been successfully performed, indicating that the EMatcher object is ready for matching, and false otherwise.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetPatternLearnt() const
```

EMatcher::GetPatternType

Pixel type of the learned pattern.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EImageType GetPatternType() const
```

EMatcher::GetPatternWidth

Learned pattern width.

Namespace: Euresys::Open_eVision

[C++]

```
int GetPatternWidth() const
```

EMatcher::GetPositions

Returns a vector of EMatchPosition objects, each containing the position coordinates and other matching results.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EMatchPosition> GetPositions() const
```

EMatcher::Save

Saves the EMatcher. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EMatcher::SetExtension

Sets the extension of the matching ROI, i.e. puts the horizontal and vertical distances, in pixels, that the found pattern occurrences may fall outside the matching ROI. Such occurrences partially outside the ROI have their score corrected by the ratio between the pattern area outside the ROI and the pattern area inside.

Namespace: Euresys::Open_eVision

[C++]

```
void SetExtension(
 int n32ExtensionX,
 int n32ExtensionY
 )
```

Parameters

n32ExtensionX

extension outside the matching ROI along its Width, in pixels.

n32ExtensionY

extension outside the matching ROI along its Height, in pixels.

EMatcher::SetPixelDimensions

Sets the physical pixel dimensions.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixelDimensions(
 float width,
 float height
 )
```

Parameters

width

Width of a pixel.

height

Height of a pixel.

Remarks

When an image has been acquired in such a way that the pixels are "non-square" the physical width and height of the area covered by a pixel are unequal, a form of anisotropy results. When rotated, the objects become skewed; rectangles become parallelograms. In such a situation, the pixel aspect ratio must be known to compensate it during matching. The aspect ratio is given by specifying the true width and height of a pixel. The specification of the pixel dimensions is only useful when rotation is used. Only the aspect ratio matters, so that relative width and height values can be given. By default, the pixel width and height are set to 1.0.

EMatcher::GetVersion

Version number of the EMatcher object.

Namespace: Euresys::Open_eVision

```
[C++]
static OEV_UINT32 GetVersion()
```

4.153. EMatrixCode Class

This class is deprecated.

Holds all the information regarding a single Data Matrix code: its decoded string, its grading, its errors,...

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the EMatrixCode corner points and symbol finder pattern (when the symbol size has been correctly detected).
DrawErrors	Draws all symbol finder pattern cells where errors were detected and corrected.
DrawErrorsWithCurrentPen	Draws the detected errors.
DrawWithCurrentPen	Draws the EMatrixCode corner points and symbol finder pattern (when the symbol size has been correctly detected).
EMatrixCode	Constructs a EMatrixCode context.
GetAngle	MatrixCode angle.
GetAxialNonUniformity	Measured axial non-uniformity value (1.0 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetAxialNonUniformityGrade	Measured axial non-uniformity grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetCellDefects	Measured cell defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetCenter	EMatrixCode center.
GetContrast	Measured symbol contrast value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

GetContrastGrade	Measured symbol contrast grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetContrastType	Symbol contrast type, as defined by EMatrixCodeContrastMode.
GetCorner	Gets a matrix code corner.
GetDataMatrixCellHeight	Measured data matrix cell height value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetDataMatrixCellWidth	Measured data matrix cell width value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetDecodedDataElement	Gets a character value of the matrix code.
GetDecodedString	Decoded Data Matrix symbol string.
GetFamily	ECC symbol family, as defined by EFamily.
GetFinderPatternDefects	Measured finder pattern defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetFlipping	Symbol flipping type, as defined by EFlipping.
GetFound	true if a EMatrixCode object has been found in the ROI supplied to EMatrixCodeReader::Read, even if it could not be successfully decoded.
GetHorizontalMarkGrowth	Measured horizontal mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetHorizontalMarkMisplacement	Measured horizontal mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetIso15415GradingParameters	ISO/IEC 15415 grading parameters
GetIso29158GradingParameters	ISO/IEC 29158 grading parameters
GetLocationThreshold	Absolute threshold (0 to 255 scale) that was used during location of the symbol.
GetLogicalSize	Symbol logical size, as defined by ELogicalSize.
GetLogicalSizeHeight	For a logical size of S1xS2, LogicalSizeHeight is the integer S1.
GetLogicalSizeWidth	For a logical size of S1xS2, LogicalSizeWidth is the integer S2.
GetMeasuredPrintGrowth	Raw, un-normalized print quality parameter (1.0 to 0 scale), after a read operation, provided that the print quality assessment has been enabled.
GetNumErrors	Number of errors that were detected and corrected while decoding the symbol.

GetOverallGrade	Overall symbol grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetPrintGrowth	Normalized measured print growth value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetPrintGrowthGrade	Measured print growth grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetReadingThreshold	Absolute threshold (0 to 255 scale) that was used during reading of the symbol.
GetSemiT10GradingParameters	Semi T10-0701 grading parameters
GetSymbolContrastSNR	Measured symbol contrast signal to noise ratio value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetUnusedErrorCorrection	Measured unused error correction value (1.0 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetUnusedErrorCorrectionGrade	Measured unused error correction grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.
GetVerticalMarkGrowth	Measured vertical mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
GetVerticalMarkMisplacement	Measured vertical mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.
IsGS1	true if the decoded string uses the GS1 standard
Load	Saves a EMatrixCode. The given ESerializer must have been created for writing.
operator=	Copies all the data from another EMatrixCode object into the current EMatrixCode object
Save	Loads a EMatrixCode. The given ESerializer must have been created for reading.
SetCorner	Sets a matrix code corner.

EMatrixCode::GetAngle

This property is deprecated.

MatrixCode angle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
```

Remarks

EMatrixCode::GetAxialNonUniformity

This property is deprecated.

Measured axial non-uniformity value (1.0 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAxialNonUniformity() const
```

EMatrixCode::GetAxialNonUniformityGrade

This property is deprecated.

Measured axial non-uniformity grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
int GetAxialNonUniformityGrade() const
```

EMatrixCode::GetCellDefects

This property is deprecated.

Measured cell defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCellDefects() const
```

Remarks

Read-only.

EMatrixCode::GetCenter

This property is deprecated.

EMatrixCode center.

Namespace: Euresys::Open_eVision

```
[C++]
const EPoint& GetCenter() const
```

EMatrixCode::GetContrast

This property is deprecated.

Measured symbol contrast value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetContrast() const
```

Remarks

This property is computed as the difference of the reference gray levels over their arithmetic average. Values range between 0 and 1.0.

EMatrixCode::GetContrastGrade

This property is deprecated.

Measured symbol contrast grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
int GetContrastGrade() const
```

EMatrixCode::GetContrastType

This property is deprecated.

Symbol contrast type, as defined by EMatrixCodeContrastMode.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EMatrixCodeContrastMode GetContrastType() const
```

EMatrixCode::GetDataMatrixCellHeight

This property is deprecated.

Measured data matrix cell height value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDataMatrixCellHeight() const
```

Remarks

Read-only.

EMatrixCode::GetDataMatrixCellWidth

This property is deprecated.

Measured data matrix cell width value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDataMatrixCellWidth() const
```

Remarks

Read-only.

EMatrixCode::GetDecodedString

This property is deprecated.

Decoded Data Matrix symbol string.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetDecodedString() const
```

EMatrixCode::Draw

This method is deprecated.

Draws the EMatrixCode corner points and symbol finder pattern (when the symbol size has been correctly detected).

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

The reference corner has a bold cross marking. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawErrors

This method is deprecated.

Draws all symbol finder pattern cells where errors were detected and corrected.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawErrors(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawErrors(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawErrors(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

This member is intended to be called in conjunction with EMatrixCode::Draw. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawErrorsWithCurrentPen

This method is deprecated.

Draws the detected errors.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawErrorsWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

-

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawWithCurrentPen

This method is deprecated.

Draws the EMatrixCode corner points and symbol finder pattern (when the symbol size has been correctly detected).

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

The reference corner has a bold cross marking. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::EMatrixCode

This method is deprecated.

Constructs a EMatrixCode context.

Namespace: Euresys::Open_eVision

[C++]

```
void EMatrixCode(
 )

void EMatrixCode(
 const EMatrixCode& other
 )
```

Parameters

other

Another EMatrixCode object to be copied in the new EMatrixCode object.

Remarks

The default constructor constructs an uninitialized EMatrixCode object. All properties are initialized to their respective default values. The copy constructor constructs a EMatrixCode context based on a pre-existing EMatrixCode object. All properties and internal data are copied.

EMatrixCode::GetFamily

This property is deprecated.

ECC symbol family, as defined by EFamily.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EFamily GetFamily() const
```

EMatrixCode::GetFinderPatternDefects

This property is deprecated.

Measured finder pattern defects value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetFinderPatternDefects() const
```

Remarks

Read-only.

EMatrixCode::GetFlipping

This property is deprecated.

Symbol flipping type, as defined by EFlipping.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EFlipping GetFlipping() const
```

EMatrixCode::GetFound

This property is deprecated.

true if a EMatrixCode object has been found in the ROI supplied to EMatrixCodeReader::Read, even if it could not be successfully decoded.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetFound() const
```

Remarks

If this property is false, it is still possible that an unlocalized matrix code exists in the image. However, if Found is false, no other property should be read.

EMatrixCode::GetCorner

This method is deprecated.

Gets a matrix code corner.

Namespace: Euresys::Open_eVision

```
[C++]
const EPoint& GetCorner(
 int index
 )
```

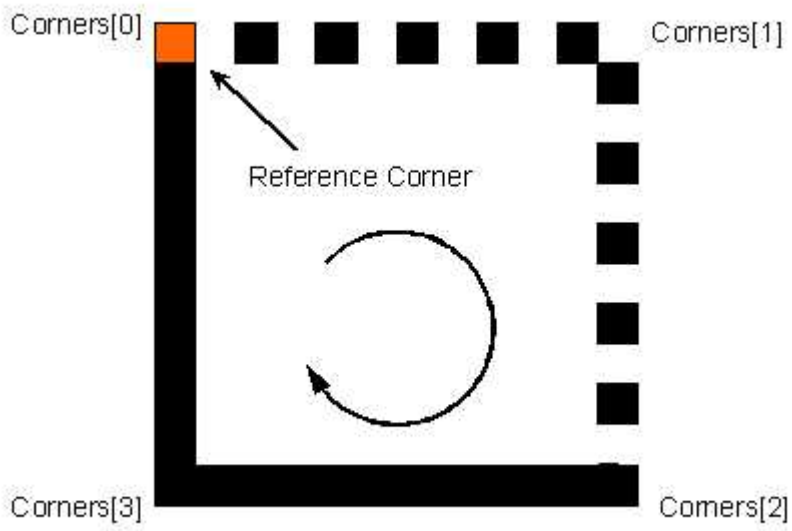
Parameters

index

Index of the matrix code corner.

Remarks

The indices of the corners use the following convention:



EMatrixCode::GetDecodedDataElement

This method is deprecated.

Gets a character value of the matrix code.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT8 GetDecodedDataElement(
 int index
 )
```

Parameters

index

Index of the character value.

Remarks

This property makes it possible to see information not coded as ASCII characters.

EMatrixCode::GetHorizontalMarkGrowth

This property is deprecated.

Measured horizontal mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetHorizontalMarkGrowth() const
```

Remarks

Read-only.

EMatrixCode::GetHorizontalMarkMisplacement

This property is deprecated.

Measured horizontal mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetHorizontalMarkMisplacement() const
```

Remarks

Read-only.

EMatrixCode::IsGS1

This method is deprecated.

true if the decoded string uses the GS1 standard

Namespace: Euresys::Open_eVision

[C++]

```
bool IsGS1(
 )
```

EMatrixCode::GetIso15415GradingParameters

This property is deprecated.

ISO/IEC 15415 grading parameters

Namespace: Euresys::Open_eVision

[C++]

```
EMatrixCodeIso15415GradingParameters GetIso15415GradingParameters() const
```

Remarks

Read-only.

EMatrixCode::GetIso29158GradingParameters

This property is deprecated.

ISO/IEC 29158 grading parameters

Namespace: Euresys::Open_eVision

[C++]

```
EMatrixCodeIso29158GradingParameters GetIso29158GradingParameters() const
```

Remarks

Read-only.

EMatrixCode::Load

This method is deprecated.

Saves a EMatrixCode. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 ESerializer* serializer
 )

void Load(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

A string containing the full path to the file.

EMatrixCode::GetLocationThreshold

This property is deprecated.

Absolute threshold (0 to 255 scale) that was used during location of the symbol.

Namespace: Euresys::Open_eVision

[C++]

```
int GetLocationThreshold() const
```

EMatrixCode::GetLogicalSize

This property is deprecated.

Symbol logical size, as defined by ELogicalSize.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ELogicalSize GetLogicalSize() const
```

EMatrixCode::GetLogicalSizeHeight

This property is deprecated.

For a logical size of S1xS2, LogicalSizeHeight is the integer S1.

Namespace: Euresys::Open_eVision

[C++]

```
int GetLogicalSizeHeight() const
```

EMatrixCode::GetLogicalSizeWidth

This property is deprecated.

For a logical size of S1xS2, LogicalSizeWidth is the integer S2.

Namespace: Euresys::Open_eVision

[C++]

```
int GetLogicalSizeWidth() const
```

EMatrixCode::GetMeasuredPrintGrowth

This property is deprecated.

Raw, un-normalized print quality parameter (1.0 to 0 scale), after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMeasuredPrintGrowth() const
```

Remarks

This property is computed as the measured area of the active cells (same color as the finder pattern) over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of 1.0, regardless the symbol size.

EMatrixCode::GetNumErrors

This property is deprecated.

Number of errors that were detected and corrected while decoding the symbol.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumErrors() const
```

Remarks

Such errors may be due to symbol degradation by scratches, blur, non-uniform illumination or slight changes in size.

EMatrixCode::operator=

This method is deprecated.

Copies all the data from another EMatrixCode object into the current EMatrixCode object

Namespace: Euresys::Open_eVision

[C++]

```
EMatrixCode& operator=(
 const EMatrixCode& other
 )
```

Parameters

other

EMatrixCode object to be copied

EMatrixCode::GetOverallGrade

This property is deprecated.

Overall symbol grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
int GetOverallGrade() const
```

EMatrixCode::GetPrintGrowth

This property is deprecated.

Normalized measured print growth value, as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetPrintGrowth() const
```

Remarks

The use of this property is a bit tricky: first a raw measure of the print growth is provided as EMatrixCode::MeasuredPrintGrowth. Then, the measurement is normalized from the EMatrixCodeReader::MinimumPrintGrowth /EMatrixCodeReader::MaximumPrintGrowth/EMatrixCodeReader::NominalPrintGrowth properties of the EMatrixCodeReader instance, which must be provided by the user. The normalized EMatrixCode::PrintGrowth ranges around 0.

EMatrixCode::GetPrintGrowthGrade

This property is deprecated.

Measured print growth grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
int GetPrintGrowthGrade() const
```

Remarks

Read-only.

EMatrixCode::GetReadingThreshold

This property is deprecated.

Absolute threshold (0 to 255 scale) that was used during reading of the symbol.

Namespace: Euresys::Open_eVision

```
[C++]
int GetReadingThreshold() const
```

Remarks

Read-only.

EMatrixCode::Save

This method is deprecated.

Loads a EMatrixCode. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 ESerializer* serializer
 )
void Save(
 const std::string& path
 )
```

Parameters

serializer

The ESerializer object that is read from.

path

A string containing the full path to the file.

EMatrixCode::GetSemiT10GradingParameters

This property is deprecated.

Semi T10-0701 grading parameters

Namespace: Euresys::Open_eVision

```
[C++]
EMatrixCodeSemiT10GradingParameters GetSemiT10GradingParameters() const
```

Remarks

Read-only.

EMatrixCode::SetCorner

This method is deprecated.

Sets a matrix code corner.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCorner(
 int index,
 const EPoint& corner
 )
```

Parameters

index

Index of the matrix code corner.

corner

New corner.

EMatrixCode::GetSymbolContrastSNR

This property is deprecated.

Measured symbol contrast signal to noise ratio value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSymbolContrastSNR() const
```

Remarks

Read-only.

EMatrixCode::GetUnusedErrorCorrection

This property is deprecated.

Measured unused error correction value (1.0 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

[C++]

```
float GetUnusedErrorCorrection() const
```

Remarks

The EMatrixCode::UnusedErrorCorrection property takes into account the number of redundant bits used for error correction only; no erasure nor error detection bits are considered.

EMatrixCode::GetUnusedErrorCorrectionGrade

This property is deprecated.

Measured unused error correction grading (4 to 0 scale), as defined by the AIM standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
int GetUnusedErrorCorrectionGrade() const
```

Remarks

Read-only.

EMatrixCode::GetVerticalMarkGrowth

This property is deprecated.

Measured vertical mark growth value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetVerticalMarkGrowth() const
```

Remarks

Read-only.

EMatrixCode::GetVerticalMarkMisplacement

This property is deprecated.

Measured vertical mark misplacement value, as defined by the Semi T10 standard, after a read operation, provided that the print quality assessment has been enabled.

Namespace: Euresys::Open_eVision

```
[C++]
float GetVerticalMarkMisplacement() const
```

Remarks

Read-only.

4.154. EMatrixCode Class

Holds all the information regarding a single Data Matrix code: its decoded string, its grading, its errors and more.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

Methods

DrawErrors	Draws the detected errors.
DrawErrorsWithCurrentPen	Draws the detected errors using the pen currently set in the graphical context.
DrawGrid	Draws the detected Data Matrix grid.
DrawGridWithCurrentPen	Draws the detected Data Matrix grid using the pen currently set in the graphical context.
DrawPosition	Draws the Data Matrix Position. This includes the outer edges of the code as well as the timing patterns.
DrawPositionWithCurrentPen	Draws the Data Matrix Position using the pen currently set in the graphical context. This includes the outer edges of the code as well as the timing patterns.
EMatrixCode	Creates an EMatrixCode object.
GetCellColor	Detected color of a cell.
GetCellCorrectedColor	Corrected color of a cell.
GetCellPosition	Position of a cell of the Data Matrix
GetDecodedString	Decoded Data Matrix symbol string.
GetDecodedStringStream	-
GetECC000Family	Retrieves the ECC000 Family for non-ECC200 Matrix Codes.
GetErrors	Retrieves the position of the errors detected in the Data Matrix symbol.
GetIsECC200	Indicates if the Data Matrix is ECC200 or not.
GetIsGraded	Returns true if the Matrix Code has been graded
GetIsGS1	Returns true if the decoded string uses the GS1 standard
GetIso15415GradingParameters	ISO/IEC 15415 grading parameters
GetIso29158GradingParameters	ISO/IEC TR 29158 grading parameters
GetIsReliable	Returns true if the code has been succesfully decoded and false otherwise.
GetPosition	Position of the Data Matrix
GetReliabilityScore	Returns a confidence score on the fact that the code is actually a data matrix.
GetSemiT10GradingParameters	Semi T10-0701 grading parameters
GetSymbolHeight	Data Matrix Symbol Height (Number of cells in the vertical direction)
GetSymbolPolarity	Symbol polarity as defined by ESymbolPolarity.
GetSymbolWidth	Data Matrix Symbol Width (Number of cells in the horizontal direction)

operator=	Assignment operator

EMatrixCode::GetDecodedString

Decoded Data Matrix symbol string.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
std::string GetDecodedString() const
```

EMatrixCode::GetDecodedStringStream

-

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
std::vector<int> GetDecodedStringStream() const
```

EMatrixCode::DrawErrors

Draws the detected errors.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void DrawErrors(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawErrors(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

Handle of the graphics context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawErrorsWithCurrentPen

This method is deprecated.

Draws the detected errors using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void DrawErrorsWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawGrid

Draws the detected Data Matrix grid.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void DrawGrid(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawGrid(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

Handle of the graphics context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawGridWithCurrentPen

This method is deprecated.

Draws the detected Data Matrix grid using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void DrawGridWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawPosition

Draws the Data Matrix Position. This includes the outer edges of the code as well as the timing patterns.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void DrawPosition(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawPosition(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

Handle of the graphics context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::DrawPositionWithCurrentPen

This method is deprecated.

Draws the Data Matrix Position using the pen currently set in the graphical context. This includes the outer edges of the code as well as the timing patterns.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void DrawPositionWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EMatrixCode::GetECC000Family

Retrieves the ECC000 Family for non-ECC200 Matrix Codes.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
Euresys::Open_eVision::EasyMatrixCode2::ECC000Family GetECC000Family() const
```

EMatrixCode::EMatrixCode

Creates an EMatrixCode object.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void EMatrixCode(
 )

void EMatrixCode(
 const EMatrixCode& other
 )
```

Parameters

other

The reference EMatrixCode instance to copy this one from.

EMatrixCode::GetErrors

Retrieves the position of the errors detected in the Data Matrix symbol.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
std::vector<Euresys::Open_eVision::EMatrixPosition> GetErrors() const
```

EMatrixCode::GetCellColor

Detected color of a cell.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
Euresys::Open_eVision::ECellColor GetCellColor(
 int x,
 int y
 )

Euresys::Open_eVision::ECellColor GetCellColor(
 EMatrixPosition position
 )
```

Parameters

x

The horizontal index of the cell.

y

The vertical index of the cell.

position

The position of the cell in the code.

Remarks

Returns an ECellColor, corresponding to the cell color as detected in the searched area.

EMatrixCode::GetCellCorrectedColor

Corrected color of a cell.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
Euresys::Open_eVision::ECellColor GetCellCorrectedColor(
 int x,
 int y
 )

Euresys::Open_eVision::ECellColor GetCellCorrectedColor(
 EMatrixPosition position
 )
```

Parameters

x

The horizontal index of the cell.

y

The vertical index of the cell.

position

The position of the cell in the code.

Remarks

Returns the ECellColor corresponding to the theorethical cell color (the color that the cell should have in the searched area).

EMatrixCode::GetCellPosition

Position of a cell of the Data Matrix

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]

EQuadrangle GetCellPosition(
 int x,
 int y
 )

EQuadrangle GetCellPosition(
 EMatrixPosition position
 )
```

Parameters

x

The horizontal index of the cell.

y

The vertical index of the cell.

position

The position of the cell in the code.

EMatrixCode::GetIsECC200

Indicates if the Data Matrix is ECC200 or not.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]

bool GetIsECC200() const
```

EMatrixCode::GetIsGraded

Returns true if the Matrix Code has been graded

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
bool GetIsGraded() const
```

EMatrixCode::GetIsGS1

Returns true if the decoded string uses the GS1 standard

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
bool GetIsGS1() const
```

EMatrixCode::GetIso15415GradingParameters

ISO/IEC 15415 grading parameters

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
EMatrixCodeIso15415GradingParameters GetIso15415GradingParameters() const
```

EMatrixCode::GetIso29158GradingParameters

ISO/IEC TR 29158 grading parameters

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
EMatrixCodeIso29158GradingParameters GetIso29158GradingParameters() const
```

EMatrixCode::GetIsReliable

Returns true if the code has been succesfully decoded and false otherwise.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
bool GetIsReliable() const
```

EMatrixCode::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EMatrixCode& operator=(
 const EMatrixCode& other
 )
```

Parameters

other

The EMatrixCode instance to assign.

EMatrixCode::GetPosition

Position of the Data Matrix

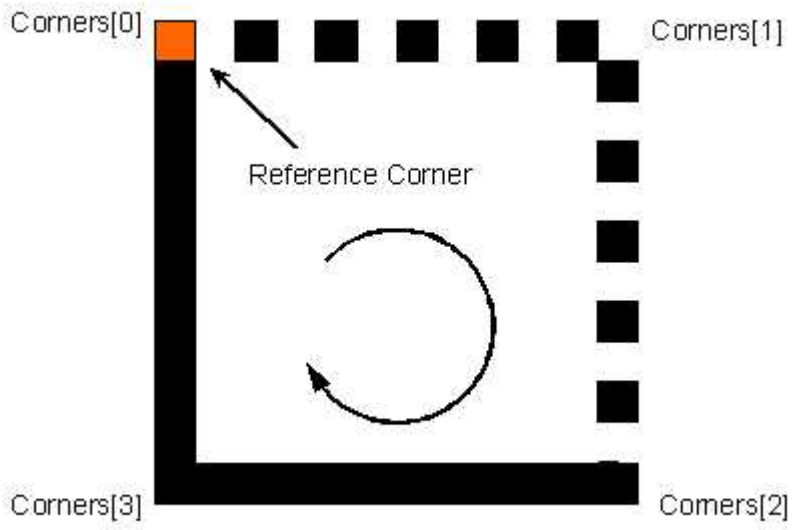
Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EQuadrangle GetPosition() const
```

Remarks

The order of the corners of the EQuadrangle uses the following convention:



EMatrixCode::GetReliabilityScore

Returns a confidence score on the fact that the code is actually a data matrix.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
float GetReliabilityScore() const
```

Remarks

Returns 1.0f if the code has been succesfully decoded.

EMatrixCode::GetSemiT10GradingParameters

Semi T10-0701 grading parameters

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EMatrixCodeSemiT10GradingParameters GetSemiT10GradingParameters() const
```

EMatrixCode::GetSymbolHeight

Data Matrix Symbol Height (Number of cells in the vertical direction)

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
int GetSymbolHeight() const
```

EMatrixCode::GetSymbolPolarity

Symbol polarity as defined by ESymbolPolarity.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
Euresys::Open_eVision::ESymbolPolarity GetSymbolPolarity() const
```

EMatrixCode::GetSymbolWidth

Data Matrix Symbol Width (Number of cells in the horizontal direction)

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
int GetSymbolWidth() const
```

4.155. EMatrixCodeGrid Class

Represents a grid of Data Matrix Codes

Namespace: Euresys::Open_eVision::EasyMatrixCode2

Methods

EMatrixCodeGrid	Creates an EMatrixCodeGrid object.
GetCellEnabled	Returns true if Cell is enabled and false otherwise
GetNumCols	Returns the number of columns in the grid
GetNumRows	Returns the number of rows in the grid
GetResults	Returns the MatrixCodes
operator=	Assignment operator
SetEnableAll	Enable/Disable all cells
SetEnableCell	Enable/Disable Cell
SetEnableColumn	Enable/Disable Column
SetEnableRow	Enable/Disable Row

EMatrixCodeGrid::EMatrixCodeGrid

Creates an EMatrixCodeGrid object.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void EMatrixCodeGrid(
 )

void EMatrixCodeGrid(
 const EMatrixCodeGrid& other
 )

void EMatrixCodeGrid(
 OEV_UINT32 numCols,
 OEV_UINT32 numRows
 )
```

Parameters

other

The reference EMatrixCodeGrid instance to copy this one from.

numCols

The number of columns in the grid.

numRows

The number of rows in the grid.

EMatrixCodeGrid::SetEnableAll

Enable/Disable all cells

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void SetEnableAll(bool enable)
```

Remarks

By default, all grid cells are enabled.

EMatrixCodeGrid::GetCellEnabled

Returns true if Cell is enabled and false otherwise

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
bool GetCellEnabled(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
```

Parameters

column

-

row

-

Remarks

By default, all grid cells are enabled.

EMatrixCodeGrid::GetResults

Returns the MatrixCodes

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
std::vector<Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode> GetResults(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
std::vector<Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode> GetResults(
 )
```

Parameters

column

The column of a cell.

row

The row of a cell.

EMatrixCodeGrid::GetNumCols

Returns the number of columns in the grid

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
OEV_UINT32 GetNumCols() const
```

EMatrixCodeGrid::GetNumRows

Returns the number of rows in the grid

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
OEV_UINT32 GetNumRows() const
```

EMatrixCodeGrid::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EMatrixCodeGrid& operator=(
 const EMatrixCodeGrid& other
 )
```

Parameters

other

The EMatrixCodeGrid instance to assign.

EMatrixCodeGrid::SetEnableCell

Enable/Disable Cell

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void SetEnableCell(
 OEV_UINT32 column,
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

column

-

row

-

enable

-

Remarks

By default, all grid cells are enabled.

EMatrixCodeGrid::SetEnableColumn

Enable/Disable Column

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void SetEnableColumn(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

EMatrixCodeGrid::SetEnableRow

Enable/Disable Row

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void SetEnableRow(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

4.156. EMatrixCodeReader Class

This class is deprecated.

A EMatrixCodeReader instance is a tool that processes an ROI and returns a EMatrixCode instance.
Deprecated, use EMatrixCodeReader instead.

Remarks

A EMatrixCodeReader has properties that allow customizing the reading process. One of these properties is EMatrixCodeReader::SearchParams, which is an instance of ESearchParamsType. This object embeds the parameter space where a Data Matrix code will be searched and has an interface of its own. The EMatrixCodeReader class is found in the Euresys namespace

Namespace: Euresys::Open_eVision

Methods

EMatrixCodeReader	Default constructor for EMatrixCodeReader objects.
GetComputeGrading	Allows to choose whether the grading properties of the EMatrixCode object will be computed by the EMatrixCodeReader::Reset, EMatrixCodeReader::Read or EMatrixCodeReader::LearnMore methods.
GetLearnMaskElement	Allows to know which decoded parameters are learnt when the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods are called.
GetMaxHeightWidthRatio	Maximum value for a Data Matrix aspect ratio
GetMaximumPrintGrowth	Maximum reference value in use for normalization of the PrintGrowth quality indicator.
GetMinimumPrintGrowth	Minimum reference value in use for normalization of the PrintGrowth quality indicator.
GetNominalPrintGrowth	Nominal reference value in use for normalization of the PrintGrowth quality indicator.
GetSearchParams	Parameter space that the algorithm uses to read a Data Matrix code from an ROI.
GetTimeOut	Time-out for the EMatrixCodeReader::Learn, EMatrixCodeReader::LearnMore and EMatrixCodeReader::Read methods.
Learn	Tries to locate, decode and read the Data Matrix code in the given ROI.
LearnMore	Tries to locate, decode and read the Data Matrix code in the given ROI.
Load	Saves a EMatrixCodeReader. The given ESerializer must have been created for writing.
Read	Tries to locate, decode and read the Data Matrix code in the given ROI.
Reset	Resets the parameter search space to its default: the full range of all parameters.
Save	Loads a EMatrixCodeReader. The given ESerializer must have been created for reading.
SetComputeGrading	Allows to choose whether the grading properties of the EMatrixCode object will be computed by the EMatrixCodeReader::Reset, EMatrixCodeReader::Read or EMatrixCodeReader::LearnMore methods.
SetIso29158CalibrationParameters	Sets ISO/IEC 29158 calibration paramters
SetLearnMaskElement	Allows to choose which decoded parameters are learnt when the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods are called.
SetMaxHeightWidthRatio	Maximum value for a Data Matrix aspect ratio
SetMaximumPrintGrowth	Maximum reference value in use for normalization of the PrintGrowth quality indicator.

SetMinimumPrintGrowth	Minimum reference value in use for normalization of the PrintGrowth quality indicator.
SetNominalPrintGrowth	Nominal reference value in use for normalization of the PrintGrowth quality indicator.
SetTimeOut	Time-out for the EMatrixCodeReader::Learn, EMatrixCodeReader::LearnMore and EMatrixCodeReader::Read methods.

EMatrixCodeReader::GetComputeGrading

EMatrixCodeReader::SetComputeGrading

This property is deprecated.

Allows to choose whether the grading properties of the EMatrixCode object will be computed by the EMatrixCodeReader::Reset, EMatrixCodeReader::Read or EMatrixCodeReader::LearnMore methods.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetComputeGrading()

void SetComputeGrading(bool value)
```

Remarks

Default: false.

EMatrixCodeReader::EMatrixCodeReader

This method is deprecated.

Default constructor for EMatrixCodeReader objects.

Namespace: Euresys::Open_eVision

[C++]

```
void EMatrixCodeReader(
 )
```

EMatrixCodeReader::GetLearnMaskElement

This method is deprecated.

Allows to know which decoded parameters are learnt when the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods are called.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetLearnMaskElement(
 Euresys::Open_eVision::ELearnParam index
 )
```

Parameters

index

Parameter identifier, as defined in ELearnParam

EMatrixCodeReader::Learn

This method is deprecated.

Tries to locate, decode and read the Data Matrix code in the given ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EMatrixCode Learn(
 const EROIBW8& roi
 )
```

Parameters

roi

ROI in which the Data Matrix has to be found.

Remarks

If successful, it adds the parameters of the Data Matrix code found into the internal learning database. The decoding results can be found in the returned EMatrixCode object. The addition of the parameters of the Data Matrix code found into the internal learning database means that subsequent Read operations will be faster, but can only be performed on similar Data Matrix codes/conditions. Only the parameters that are tagged for learning are remembered for the subsequent EMatrixCodeReader::Read operations (see EMatrixCodeReader::SetLearnMaskElement). See the EMatrixCode::Found property for information about the outcome of the Learn process.

EMatrixCodeReader::LearnMore

This method is deprecated.

Tries to locate, decode and read the Data Matrix code in the given ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EMatrixCode LearnMore(
 const EROIBW8& roi
 )
```

Parameters

roi

ROI in which the Data Matrix has to be found.

Remarks

If successful, it cumulates the parameters of the Data Matrix code found with those already present in the internal learning database. The decoding results can be found in the returned EMatrixCode object. The cumulation of the parameters of the Data Matrix code found with those already present in the internal learning database means that subsequent Read operations will be faster, but can only be performed on similar Data Matrix codes/conditions. Only the parameters that are tagged for learning are remembered for the subsequent EMatrixCodeReader::Read operations (see EMatrixCodeReader::SetLearnMaskElement). See the EMatrixCode::Found property for information about the outcome of the LearnMore process.

EMatrixCodeReader::Load

This method is deprecated.

Saves a EMatrixCodeReader. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 ESerializer* serializer
 )

void Load(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

A string containing the full path to the file.

EMatrixCodeReader::GetMaxHeightWidthRatio

EMatrixCodeReader::SetMaxHeightWidthRatio

This property is deprecated.

Maximum value for a Data Matrix aspect ratio

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxHeightWidthRatio()
void SetMaxHeightWidthRatio(float value)
```

Remarks

This property allows controlling what kind of objects are considered as potential MatrixCode instances in the image. When objects are found in the image, only those where the bounding box has an aspect ratio smaller than this value are taken into account for digitization and decoding. The default value is 3.8, and should be adjusted if the MatrixCode cells in your image are non-square, or if your matrix code uses a very non-square symbology such as 32x8. The supplied value must lie between 0.0 and 5.0.

EMatrixCodeReader::GetMaximumPrintGrowth

EMatrixCodeReader::SetMaximumPrintGrowth

This property is deprecated.

Maximum reference value in use for normalization of the PrintGrowth quality indicator.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaximumPrintGrowth()
void SetMaximumPrintGrowth(float value)
```

Remarks

Default: 2.0. After the standard, parameter PrintGrowth must be computed as normalized value related to three references: minimum, nominal and maximum values have to be provided. Parameter MeasuredPrintGrowth is the raw, un-normalized print quality parameter. It is computed as the measured area of the active cells over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of unity. The PrintGrowth is derived from the MeasuredPrintGrowth by means of the three normalization parameters.

EMatrixCodeReader::GetMinimumPrintGrowth

EMatrixCodeReader::SetMinimumPrintGrowth

This property is deprecated.

Minimum reference value in use for normalization of the PrintGrowth quality indicator.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumPrintGrowth()
void SetMinimumPrintGrowth(float value)
```

Remarks

Default: 0.0. After the standard, parameter PrintGrowth must be computed as normalized value related to three references: minimum, nominal and maximum values have to be provided. Parameter MeasuredPrintGrowth is the raw, un-normalized print quality parameter. It is computed as the measured area of the active cells over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of unity. The PrintGrowth is derived from the MeasuredPrintGrowth by means of the three normalization parameters.

EMatrixCodeReader::GetNominalPrintGrowth

EMatrixCodeReader::SetNominalPrintGrowth

This property is deprecated.

Nominal reference value in use for normalization of the PrintGrowth quality indicator.

Namespace: Euresys::Open_eVision

[C++]

```
float GetNominalPrintGrowth()
void SetNominalPrintGrowth(float value)
```

Remarks

Default: 1.0. After the standard, parameter PrintGrowth must be computed as normalized value related to three references: minimum, nominal and maximum values have to be provided. Parameter MeasuredPrintGrowth is the raw, un-normalized print quality parameter. It is computed as the measured area of the active cells over the area of ideal square cells, having sides equal to the pitches (such cells perfectly tile the symbol). Its value typically is on the order of unity. The PrintGrowth is derived from the MeasuredPrintGrowth by means of the three normalization parameters.

EMatrixCodeReader::Read

This method is deprecated.

Tries to locate, decode and read the Data Matrix code in the given ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EMatrixCode Read(
 const EROIBW8& roi
 )
```

Parameters

roi

ROI in which the Data Matrix has to be found.

Remarks

The decoding results can be found in the returned EMatrixCode object. See the EMatrixCode::Found property for information about the outcome of the Read process.

Note. This function throws an exception if the matrix code in the given ROI can not be read.

EMatrixCodeReader::Reset

This method is deprecated.

Resets the parameter search space to its default: the full range of all parameters.

Namespace: Euresys::Open_eVision

[C++]

```
void Reset(
 )
```

Remarks

This does not modify the learning mask.

EMatrixCodeReader::Save

This method is deprecated.

Loads a EMatrixCodeReader. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 ESerializer* serializer
 )
```

```
void Save(
 const std::string& path
 )
```

Parameters

serializer

The ESerializer object that is read from.

path

A string containing the full path to the file.

EMatrixCodeReader::GetSearchParams

This property is deprecated.

Parameter space that the algorithm uses to read a Data Matrix code from an ROI.

Namespace: Euresys::Open_eVision

```
[C++]
ESearchParamsType& GetSearchParams()
```

Remarks

It can be modified through automatic learning (using the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods) or by using EMatrixCodeReader::SearchParams properties and methods.

EMatrixCodeReader::SetIso29158CalibrationParameters

This method is deprecated.

Sets ISO/IEC 29158 calibration paramters

Namespace: Euresys::Open_eVision

```
[C++]
void SetIso29158CalibrationParameters(
 float Rcal,
 float MLcal,
 float SRcal,
 float SRtarget
 )
```

Parameters

Rcal

Reported reflectance value, from a calibration standard.

MLcal

Mean of the light from a histogram of the calibrated standard.

SRcal

System response parameters(such as exposure and/or again) used to create an image of the calibration standard.

SRtarget

System response parameters(such as exposure and/or again) used to create an image of the symbole under test.

EMatrixCodeReader::SetLearnMaskElement

This method is deprecated.

Allows to choose which decoded parameters are learnt when the EMatrixCodeReader::Learn or EMatrixCodeReader::LearnMore methods are called.

Namespace: Euresys::Open_eVision

[C++]

```
void SetLearnMaskElement(
 Euresys::Open_eVision::ELearnParam index,
 bool value
 )
```

Parameters

index

Parameter identifier, as defined in ELearnParam.

value

true to enable the parameter for learning.

Remarks

In order to enable a parameter for learning, you need to set corresponding item of LearnMask to true. Default: all items are set to true.

EMatrixCodeReader::GetTimeOut

EMatrixCodeReader::SetTimeOut

This property is deprecated.

Time-out for the EMatrixCodeReader::Learn, EMatrixCodeReader::LearnMore and EMatrixCodeReader::Read methods.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTimeOut()

void SetTimeOut(OEV_UINT32 value)
```

Remarks

If the processing time of one of these functions becomes longer than the set time-out, the processing is stopped and an exception is thrown. In that case, the error code of the exception is EError_TimeoutReached. The time-out is set in microseconds. This time-out is not a real time-out. The processing is stopped as soon as possible after the time-out has been reached. This means that the time elapsed effectively in the method can be greater than the time-out in itself.

4.157. EMatrixCodeReader Class

An EMatrixCodeReader instance can detect, decode and grade matrixcodes in an ROI, it returns a vector of EMatrixCode instances.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

Methods

EMatrixCodeReader	Creates an EMatrixCodeReader object.
GetComputeGrading	Allows to choose whether the grading properties of the EMatrixCode object will be computed by the EMatrixCodeReader::Read method. The default setting for this property is `false`.
GetEnableDMRE	Enables or disables the decoding of Datamatrix Rectangular Extension (DMRE).
GetEnablePermissiveDecoding	Enables or disables the decoding of codes containing partially incorrect information.
GetEnableReturnUnreliableCodes	Enable or disable unreliable codes to be returned.
GetIso29158CalibrationParameters	ISO/IEC TR 29158 calibration parameters.
GetLearnPerformed	Returns true if some context information has been learned from a call to EMatrixCodeReader::Learn, and false otherwise.
GetMatrixCodeDimensionsRange	Sets or disables the range, in pixels, that the datamatrices sides dimensions must have to be detected.
GetMaxNumCodes	Maximum number of data matrices to find in a single Image/ROI.
GetReadMode	The EReadMode used for the EMatrixCodeReader::Read method. The default value for this property is EReadMode_Speed.
GetReadResults	Outputs the EMatrixCode that were found by the EMatrixCodeReader::Read method.
GetTimeOut	The timeout for the EMatrixCodeReader::Read and EMatrixCodeReader::Learn method in microseconds.
Learn	Learns the optimal parameter settings for detecting datamatrices in the given ROI.
Load	Load the configuration for this EMatrixCodeReader instance.
operator=	Assignment operator

Read	Tries to locate, decode and read data matrix codes in the given ROI.
ResetLearning	Forgets the learned parameter settings and resets their default values.
Save	Save the configuration for this EMatrixCodeReader instance.
SetComputeGrading	Allows to choose whether the grading properties of the EMatrixCode object will be computed by the EMatrixCodeReader::Read method. The default setting for this property is false.
SetEnableDMRE	Enables or disables the decoding of Datamatrix Rectangular Extension (DMRE).
SetEnablePermissiveDecoding	Enables or disables the decoding of codes containing partially incorrect information.
SetEnableReturnUnreliableCodes	Enable or disable unreliable codes to be returned.
SetIso29158CalibrationParameters	ISO/IEC TR 29158 calibration parameters.
SetMatrixCodeDimensionsRange	Sets or disables the range, in pixels, that the datamatrices sides dimensions must have to be detected.
SetMaxNumCodes	Maximum number of data matrices to find in a single Image/ROI.
SetReadMode	The EReadMode used for the EMatrixCodeReader::Read method. The default value for this property is EReadMode_Speed.
SetTimeOut	The timeout for the EMatrixCodeReader::Read and EMatrixCodeReader::Learn method in microseconds.
StopProcess	Stops the EMatrixCodeReader::Read and/or EMatrixCodeReader::Learn process as soon as possible.
UnsetMatrixCodeDimensionsRange	-

EMatrixCodeReader::GetComputeGrading

EMatrixCodeReader::SetComputeGrading

Allows to choose whether the grading properties of the EMatrixCode object will be computed by the EMatrixCodeReader::Read method. The default setting for this property is false.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
bool GetComputeGrading() const

void SetComputeGrading(bool computeGrading)
```

EMatrixCodeReader::EMatrixCodeReader

Creates an EMatrixCodeReader object.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void EMatrixCodeReader(
 )
void EMatrixCodeReader(
 const EMatrixCodeReader& other
 )
```

Parameters

other

Another EMatrixCodeReader object to be copied in the new EMatrixCodeReader object.

EMatrixCodeReader::GetEnableDMRE

EMatrixCodeReader::SetEnableDMRE

Enables or disables the decoding of Datamatrix Rectangular Extension (DMRE).

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
bool GetEnableDMRE() const
void SetEnableDMRE(bool enable)
```

EMatrixCodeReader::GetEnablePermissiveDecoding

EMatrixCodeReader::SetEnablePermissiveDecoding

Enables or disables the decoding of codes containing partially incorrect information.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
bool GetEnablePermissiveDecoding() const
void SetEnablePermissiveDecoding(bool enable)
```

EMatrixCodeReader::GetEnableReturnUnreliableCodes

EMatrixCodeReader::SetEnableReturnUnreliableCodes

Enable or disable unreliable codes to be returned.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
bool GetEnableReturnUnreliableCodes() const

void SetEnableReturnUnreliableCodes(bool enable)
```

Remarks

By default, unreliable codes are not returned. Unreliable codes are objects which are likely to be data matrices, but which could not be decoded. These unreliably decoded codes will return false upon calling EMatrixCode.

EMatrixCodeReader::GetIso29158CalibrationParameters

EMatrixCodeReader::SetIso29158CalibrationParameters

ISO/IEC TR 29158 calibration parameters.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EMatrixCodeIso29158CalibrationParameters GetIso29158CalibrationParameters() const

void SetIso29158CalibrationParameters(EMatrixCodeIso29158CalibrationParameters params)
```

EMatrixCodeReader::Learn

Learns the optimal parameter settings for detecting datamatrices in the given ROI.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void Learn(
 const EROIBW8& roi
 )

void Learn(
 const EROIBW8& roi,
 const ERegion& region
 )
```

Parameters

roi

The ROI in which the data matrix codes have to be found.

region

-

Remarks

Throws an exception if EMatrixCodeReader returns true and the ROI dimensions are different from the ones used on previous calls.

EMatrixCodeReader::GetLearnPerformed

Returns true if some context information has been learned from a call to EMatrixCodeReader::Learn, and false otherwise.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
bool GetLearnPerformed() const
```

EMatrixCodeReader::Load

Load the configuration for this EMatrixCodeReader instance.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The path from which to load the configuration.

serializer

The serializer.

EMatrixCodeReader::GetMatrixCodeDimensionsRange

EMatrixCodeReader::SetMatrixCodeDimensionsRange

Sets or disables the range, in pixels, that the datamatrices sides dimensions must have to be detected.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EIntegerRange GetMatrixCodeDimensionsRange() const

void SetMatrixCodeDimensionsRange(const EIntegerRange& range)
```

Remarks

If this parameter is set by the user, it supersedes the value learned by the EMatrixCodeReader::Learn method. If this parameter is set after a EMatrixCodeReader::Learn, EMatrixCodeReader::ResetLearning will be called.

EMatrixCodeReader::GetMaxNumCodes

EMatrixCodeReader::SetMaxNumCodes

Maximum number of data matrices to find in a single Image/ROI.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
OEV_UINT32 GetMaxNumCodes() const

void SetMaxNumCodes(OEV_UINT32 maxNumCodes)
```

Remarks

By default, this parameter is set to 1.

EMatrixCodeReader::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
EMatrixCodeReader& operator=(
 const EMatrixCodeReader& other
 )
```

Parameters

other

EMatrixCodeReader object to be copied.

EMatrixCodeReader::Read

Tries to locate, decode and read data matrix codes in the given ROI.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
std::vector<Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode> Read(
 const EROIBW8& roi
 )

std::vector<Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode> Read(
 const EROIBW8& roi,
 const ERegion& region
 )

std::vector<Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode> Read(
 const EROIBW8& roi,
 int numCellsX,
 int numCellsY,
 float extension
 )

EMatrixCodeGrid Read(
 const EROIBW8& roi,
 const ERectangleRegion& area,
 int numCellsX,
 int numCellsY,
 float extension
 )

EMatrixCodeGrid Read(
 const EROIBW8& roi,
 const ERectangleRegion& area,
 const EMatrixCodeGrid& grid,
 float extension
 )
```

Parameters

roi

The ROI in which the data matrix codes have to be found.

region

Region into the search field where the data matrix codes have to be found.

numCellsX

Number of grid cells in the X direction

numCellsY

Number of grid cells in the Y direction

extension

Extension of the grid cells to allow cell overlap. For instance, 0.0f means no extension and 0.1f means a 10% cell size extension.

area

Rectangular Region used as the full grid area

grid

Grid with cell disabling capabilities

Remarks

Throws an exception if EMatrixCodeReader returns true and the ROI dimensions are different from the ones used on previous calls. The grid overload allows you to disable some cells of the grid if those cells are not supposed to contain datamatrix codes. See the EMatrixCodeGrid class documentation for more information.

EMatrixCodeReader::GetReadMode

EMatrixCodeReader::SetReadMode

The EReadMode used for the EMatrixCodeReader::Read method. The default value for this property is Speed.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
Euresys::Open_eVision::EasyMatrixCode2::EReadMode GetReadMode() const
void SetReadMode(Euresys::Open_eVision::EasyMatrixCode2::EReadMode mode)
```

EMatrixCodeReader::GetReadResults

Outputs the EMatrixCode that were found by the EMatrixCodeReader::Read method.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

[C++]

```
std::vector<Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode> GetReadResults() const
```

EMatrixCodeReader::ResetLearning

Forgets the learned parameter settings and resets their default values.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void ResetLearning(
 )
```

EMatrixCodeReader::Save

Save the configuration for this EMatrixCodeReader instance.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The path to which to save the configuration.

serializer

The serializer.

EMatrixCodeReader::StopProcess

Stops the EMatrixCodeReader::Read and/or EMatrixCodeReader::Learn process as soon as possible.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void StopProcess(
 )
```

Remarks

When this method is called the process is stopped at the first checkpoint.

EMatrixCodeReader::GetTimeOut

EMatrixCodeReader::SetTimeOut

The timeout for the EMatrixCodeReader::Read and EMatrixCodeReader::Learn method in microseconds.

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
OEV_UINT64 GetTimeOut() const
void SetTimeOut(OEV_UINT64 timeout)
```

Remarks

If the processing time of one of these methods becomes longer than the set time-out period, the process is stopped.
The EMatrixCodeReader::Read method will return all the codes it has decoded up to that point.
The EMatrixCodeReader::Learn method will only learn from those codes it has found within the time-out period.
Note that the time-out period is not exact: the process is stopped at the first checkpoint after the time-out period has elapsed.

EMatrixCodeReader::UnsetMatrixCodeDimensionsRange

-

Namespace: Euresys::Open_eVision::EasyMatrixCode2

```
[C++]
void UnsetMatrixCodeDimensionsRange(
 )
```

4.158. EMeasurementUnit Class

The measurement units that are supported by Open eVision.

Remarks

Measurement units are used to represent physical units, such as "meter" or "inch", and ease conversions between different unit systems. They are used to build dimensional values. The following length measurement units are predefined: EUnit_um (microns), EUnit_mm, EUnit_cm, EUnit_dm, EUnit_m, EUnit_Dm, EUnit_Hm, EUnit_Km, EUnit_mil (1/1000 inch), EUnit_inch, EUnit_foot, EUnit_yard, EUnit_mile.

Namespace: Euresys::Open_eVision

Methods

ConversionFactorTo	Returns the factor needed to convert from this measurement unit to a second one.
EMeasurementUnit	Constructs a measurement unit.
GetMagnitude	Relative magnitude of this unit, with respect to a standard unit (e.g. 1 mm = 0.001 m).
GetName	Pointer to a NULL-terminated string containing the unit abbreviation.
GetStockMeasurementUnit	Returns a predefined measurement unit.
SetMagnitude	Relative magnitude of this unit, with respect to a standard unit (e.g. 1 mm = 0.001 m).
SetName	Pointer to a NULL-terminated string containing the unit abbreviation.

EMeasurementUnit::ConversionFactorTo

Returns the factor needed to convert from this measurement unit to a second one.

Namespace: Euresys::Open_eVision

[C++]

```
float ConversionFactorTo(
 const EMeasurementUnit& Unit
 )
```

Parameters

Unit

Reference to the second measurement unit.

EMeasurementUnit::EMeasurementUnit

Constructs a measurement unit.

Namespace: Euresys::Open_eVision

[C++]

```
void EMeasurementUnit(
 float magnitude,
 const std::string& name
 )
```

Parameters

magnitude

Relative magnitude of this unit with respect to a standard (e.g. 1 mm = 0.001 m).

name

Unit abbreviation (e.g. "mm").

EMeasurementUnit::GetStockMeasurementUnit

Returns a predefined measurement unit.

Namespace: Euresys::Open_eVision

```
[C++]
EMeasurementUnit* GetStockMeasurementUnit(
 Euresys::Open_eVision::EStockMeasurementUnit unit
 )
```

Parameters

unit

Enum defining the predefined measurement unit.

EMeasurementUnit::GetMagnitude

EMeasurementUnit::SetMagnitude

Relative magnitude of this unit, with respect to a standard unit (e.g. 1 mm = 0.001 m).

Namespace: Euresys::Open_eVision

```
[C++]
float GetMagnitude() const
void SetMagnitude(float f32Magnitude)
```

EMeasurementUnit::GetName

EMeasurementUnit::SetName

Pointer to a NULL-terminated string containing the unit abbreviation.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetName() const
```

```
void SetName(const std::string& name)
```

4.159. EMemorySerializer Class

Handles and EMemorySerializer context

Base Class:ESerializer
Namespace: Euresys::Open_eVision

Methods

Close	Closes the serializer
GetBuffer	Address of the internal buffer.
GetBufferSize	Size of the internal buffer
GetCurrentPosition	Current position in the buffer
GetWriting	Checks if the serializer is in writing mode

EMemorySerializer::GetBuffer

Address of the internal buffer.

Namespace: Euresys::Open_eVision

[C++]

```
const void* GetBuffer() const
```

EMemorySerializer::GetBufferSize

Size of the internal buffer

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetBufferSize() const
```

EMemorySerializer::Close

Closes the serializer

Namespace: Euresys::Open_eVision

[C++]

```
void Close(
 )
```

EMemorySerializer::GetCurrentPosition

Current position in the buffer

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetCurrentPosition()
```

EMemorySerializer::GetWriting

Checks if the serializer is in writing mode

Namespace: Euresys::Open_eVision

[C++]

```
bool GetWriting() const
```

4.160. EMesh Class

Represents a 3D meshed object (https://en.wikipedia.org/wiki/Triangle_mesh).

Namespace: Euresys::Open_eVision::Easy3D

Methods

Clear	Empties the Mesh.
ComputePlaneBehind	Returns the E3DPlane with given normal on which the EMesh lays. This is useful when projecting an EMesh on an EZMap.
EMesh	Creates an EMesh object.
GetNormals	Returns a pointer to the array (of E3DPoint) representing the list of normals. If the EMesh object has no triangle mesh data, this method returns NULL.
GetPointCloud	Returns the point cloud of the EMesh object.
GetTriangleCount	Returns the number of triangles. If the EMesh object has no triangle mesh data, the method returns 0.

GetTriangleIndexes	Returns a pointer to the index array (an array of 32 bit signed integers) representing the list of triangles. Indexes are referring to the array of E3DPoint. A triangle is defined by 3 indexes (T1a T1b T1c T2a T2b T2c T3a ....). The triangle index array size is a multiple of 3. Index values must be in [0, N[where N is the number of points in the point cloud. If the EMesh object has no triangle mesh data, this method returns NULL.
Load	Loads the EMesh. Supported formats are Open eVision proprietary and STL. The given ESerializer must have been created for reading.
LoadSTL	Loads a triangle mesh from a STL file (https://en.wikipedia.org/wiki/STL_(file_format))
operator=	Assignment operator.
Save	Saves the EMesh. Supported formats are Open eVision proprietary and STL. The given ESerializer must have been created for writing.
SaveSTL	Saves the triangle mesh to an STL file (https://en.wikipedia.org/wiki/STL_(file_format)).

EMesh::Clear

Empties the Mesh.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Clear(
 )
```

EMesh::ComputePlaneBehind

Returns the E3DPlane with given normal on which the EMesh lays. This is useful when projecting an EMesh on an EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane ComputePlaneBehind(
 E3DPoint normal,
 float margin
 )
```

Parameters

normal

The normal of the plane.

margin

Let P be the plane, A point of the EMesh closest to P and B the of the EMesh furthest to B. margin = dist(P, A) / dist(P, B). Must be positive. Default: 0.02.

EMesh::EMesh

Creates an EMesh object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EMesh(
 )
void EMesh(
 const EPointCloud& pointCloud,
 const std::vector<int>& triangleIndexes
 )
void EMesh(
 const EPointCloud& pointCloud,
 const std::vector<int>& triangleIndexes,
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& normals
 )
void EMesh(
 const EMesh& other
 )
```

Parameters

pointCloud

A point cloud

triangleIndexes

The triangle mesh is a list of indexes, referring to the array contained in "pointCloud".
A triangle is defined by 3 indexes (T1a T1b T1c T2a T2b T2c T3a ....).
Index values must be in [0, N[with N the number of points in the EPointCloud.
The triangle index array size is a multiple of 3.

normals

Normals of each of the faces of the mesh. Must be the third of the size of "triangleIndexes".
If not provided, they will be computed automatically from the faces by assuming the vertices are listed in counter-clock-wise order from outside (which is the convention used by the stl file format).

other

Another EMesh.

EMesh::Load

Loads the EMesh. Supported formats are Open eVision proprietary and STL. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EMesh::LoadSTL

Loads a triangle mesh from a STL file (https://en.wikipedia.org/wiki/STL_(file_format))

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadSTL(
 const std::string& path
 )
```

Parameters

path

The path to the STL file.

EMesh::GetNormals

Returns a pointer to the array (of E3DPoint) representing the list of normals.
If the EMesh object has no triangle mesh data, this method returns NULL.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const void* GetNormals() const
```

EMesh::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EMesh& operator=(
 const EMesh& other
 )
```

Parameters

other

-

EMesh::GetPointCloud

Returns the point cloud of the EMesh object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EPointCloud& GetPointCloud() const
```

EMesh::Save

Saves the EMesh. Supported formats are Open eVision proprietary and STL. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EMesh::SaveSTL

Saves the triangle mesh to an STL file (https://en.wikipedia.org/wiki/STL_(file_format)).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveSTL(
 const std::string& path,
 bool binary
 )
```

Parameters

path

The path to the STL file.

binary

Optional parameter, activates the binary file format (default is true).

EMesh::GetTriangleCount

Returns the number of triangles.
If the EMesh object has no triangle mesh data, the method returns 0.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetTriangleCount() const
```

EMesh::GetTriangleIndexes

Returns a pointer to the index array (an array of 32 bit signed integers) representing the list of triangles. Indexes are referring to the array of E3DPoint.
A triangle is defined by 3 indexes (T1a T1b T1c T2a T2b T2c T3a ....). The triangle index array size is a multiple of 3.
Index values must be in [0, N[where N is the number of points in the point cloud.
If the EMesh object has no triangle mesh data, this method returns NULL.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const void* GetTriangleIndexes() const
```

4.161. EMeshToZMapConverter Class

Computes an EZMap from an EMesh. The value of the pixels of the ZMap are the distance between the 3D points and the reference plane.
All 3D points under the reference plane are discarded.
Various options can be set with methods EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::SetFillMode, EMeshToZMapConverter::SetMapXYResolution, EMeshToZMapConverter::MapZResolution, EMeshToZMapConverter::OrientationVector...
When the conversion is called without defining specific parameters, the algorithm uses the following options:
- The reference plane is the horizontal plane.
- The orientation vector is selected automatically.
- The origin is set as the lowest left position of the projected point cloud on the reference plane.
- The resolution (the dimensions of the Z map) is estimated to have approximately one Point Cloud point per ZMap pixels.
- The scale is calculated from the point cloud ranges and the estimated resolution.
- The fill mode is enabled and the method is set to 'EFillUndefinedPixelsDirection_Local' (see method EDepthMap8::FillUndefinedPixels).

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	Computes an EZMap from a world space EMesh. The value of the pixels of the ZMap are the distance between the 3D triangles and the reference plane. Various options can be set with methods EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector, EMeshToZMapConverter::SetMapSize, ...
EMeshToZMapConverter	Creates an EMeshToZMapConverter object.
EnableFillMode	Enables or disables fill mode. Fill mode parameters are defined by method EMeshToZMapConverter::SetFillMode or EMeshToZMapConverter::SetFillModeMedian. Fill mode is enabled by default. If fill mode is disabled, undefined pixels may remain in the EZMap.
GetExtension	Sets a metric value used to enlarge the point cloud 3D domain. That value affects X,Y and Z directions and can be used to generate an EZMap with borders of undefined pixels. Default value is 0, which means a ZMap without border.
GetFillUndefinedPixelsDirection	Gets the undefined pixel fill direction (see EFillUndefinedPixelsDirection).
GetFillUndefinedPixelsMethod	Gets the undefined pixel fill method (see EFillUndefinedPixelsMethod).
GetMapHeight	Gets the required height (number of pixels) of the generated EZMap. By default, the required size is not set.

GetMapWidth	Gets the required width (number of pixels) of the generated EZMap. By default, the required size is not set.
GetMapXResolution	Gets the resolution of the EZMap pixels along the X axis.
GetMapYResolution	Gets the resolution of the EZMap pixels along the Y axis.
GetMapZResolution	Gets/sets the EZMap Z resolution, in world space units per gray value. The resolution is used to compute the transformation of the world Z position to an integer 8, 16 or 32 bits pixel value.
GetOrientationVector	Sets an explicit orientation for the EZMap. Overrides the orientation mode given by the method EMeshToZMapConverter::OrientationVectorMode.
GetOrientationVectorMode	Chooses the EZMap orientation from a list of predefined axis, automatic mode or user defined vector. Use EMeshToZMapConverter::OrientationVector to set an explicit orientation vector for the ZMap.
GetOrigin	Chooses the EZMap origin. It is the 3D world position used as origin of the ZMap upper left pixel (0,0).
GetReferencePlane	Sets the E3DPlane reference plane. The resulting EZMap is the distance of the 3D points above that plane. 3D points below the reference plane are discarded.
GetReferencePlaneMode	Sets an axis aligned reference plane. Overrides the explicit reference plane given by the method EMeshToZMapConverter::ReferencePlane.
GetWorldToZMapTransform	Explicitly sets the world to ZMap transformation. EMeshToZMapConverter::WorldToZMapTransform overrides the settings done by EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector and EMeshToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the world positions are transformed to the EZMap space. The matrix must be a rigid transformation (translation and rotation only). The resolution of the ZMap are defined by the EMeshToZMapConverter::SetMapXYResolution and EMeshToZMapConverter::MapZResolution methods.
GetZMaptoWorldTransform	Explicitly sets the ZMap to World transformation. "SetZMaptoWorldTransform" overrides the settings done by EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector and EMeshToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the EZMap positions are transformed to the World space. The matrix must be a rigid transformation (translation and rotation only). The resolutions of the World are defined by the EMeshToZMapConverter::SetMapXYResolution and EMeshToZMapConverter::MapZResolution methods.

IsFillModeEnabled	Tells if the fill mode is enabled or not. Use EMeshToZMapConverter::EnableFillMode to toggle the fill mode and EMeshToZMapConverter::SetFillMode or EMeshToZMapConverter::SetFillModeMedian to set the filling parameters.
Load	Loads the converter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator
operator==	Comparison operator
Save	Saves the converter configuration. The given ESerializer must have been created for writing.
SetExtension	Sets a metric value used to enlarge the point cloud 3D domain. That value affects X,Y and Z directions and can be used to generate an EZMap with borders of undefined pixels. Default value is 0, which means a ZMap without border.
SetFillMode	Inpainting options used to fill the "holes" in the EZMap. A hole exists when no 3D point is projected at that pixel position in the ZMap.
SetFillModeMedian	Inpainting options used to fill the "holes" in the EZMap using a median rectangular kernel of odd size. A hole exists when no 3D point is projected at that pixel position in the ZMap.
SetMapSize	Sets the required size of the generated EZMap; expressed in number of pixels for width and height dimensions. By default, the required size is not set.
SetMapXYResolution	Sets the resolution (possibly anisotropic) of the EZMap pixels along the X and Y axes, in world space units per pixel (e.g mm/pixel). The resolution is used to compute the ZMap size (width and height), depending on the projected point cloud on the reference plane.
SetMapZResolution	Gets/sets the EZMap Z resolution, in world space units per gray value. The resolution is used to compute the transformation of the world Z position to an integer 8, 16 or 32 bits pixel value.
SetOrientationVector	Sets an explicit orientation for the EZMap. Overrides the orientation mode given by the method EMeshToZMapConverter::OrientationVectorMode.
SetOrientationVectorMode	Chooses the EZMap orientation from a list of predefined axis, automatic mode or user defined vector. Use EMeshToZMapConverter::OrientationVector to set an explicit orientation vector for the ZMap.
SetOrigin	Chooses the EZMap origin. It is the 3D world position used as origin of the ZMap upper left pixel (0,0).
SetReferencePlane	Sets the E3DPlane reference plane. The resulting EZMap is the distance of the 3D points above that plane. 3D points below the reference plane are discarded.
SetReferencePlaneMode	Sets an axis aligned reference plane. Overrides the explicit reference plane given by the method EMeshToZMapConverter::ReferencePlane.

SetWorldToZMapTransform	Explicitly sets the world to ZMap transformation. EMeshToZMapConverter::WorldToZMapTransform overrides the settings done by EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector and EMeshToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the world positions are transformed to the EZMap space. The matrix must be a rigid transformation (translation and rotation only). The resolution of the ZMap are defined by the EMeshToZMapConverter::SetMapXYResolution and EMeshToZMapConverter::MapZResolution methods.
SetZMaptoWorldTransform	Explicitly sets the ZMap to World transformation. "SetZMaptoWorldTransform" overrides the settings done by EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector and EMeshToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the EZMap positions are transformed to the World space. The matrix must be a rigid transformation (translation and rotation only). The resolutions of the World are defined by the EMeshToZMapConverter::SetMapXYResolution and EMeshToZMapConverter::MapZResolution methods.
UnsetMapSize	Unsets the resolution. Lets the conversion decides for the optimum resolution, depending on the projected point cloud and pixel scale.
UnsetMapXYResolution	Unsets the X and Y resolutions. Lets the conversion decide the optimal resolution, depending on the projected point cloud and EZMap size.
UnsetMapZResolution	Unsets the ZMap Z resolution. Lets the conversion decide the optimal Z resolution.
UnsetOrigin	Lets the conversion process decides for the EZMap origin position (based on projected point cloud on the reference plane). Use EMeshToZMapConverter::Origin to enable and choose the ZMap origin.
UnsetWorldToZMapTransform	Disables the explicit world to ZMap transformation, set with EMeshToZMapConverter::WorldToZMapTransform.

EMeshToZMapConverter::Convert

Computes an EZMap from a world space EMesh. The value of the pixels of the ZMap are the distance between the 3D triangles and the reference plane. Various options can be set with methods EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector, EMeshToZMapConverter::SetMapSize, ...

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Convert(
 const EMesh& obj,
 EZMap8& zmap
 )
void Convert(
 const EMesh& obj,
 EZMap16& zmap
 )
void Convert(
 const EMesh& obj,
 EZMap32f& zmap
 )
void Convert(
 const EMesh& obj,
 EZMap* zmap
 )
```

Parameters

obj

The input 3D mesh.

zmap

The generated ZMap in 8, 16 or 32 bits format.

EMeshToZMapConverter::EMeshToZMapConverter

Creates an EMeshToZMapConverter object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EMeshToZMapConverter(
 )
void EMeshToZMapConverter(
 const EMeshToZMapConverter& other
 )
```

Parameters

other

Reference to the EMeshToZMapConverter object used for the initialization.

EMeshToZMapConverter::EnableFillMode

Enables or disables fill mode. Fill mode parameters are defined by method EMeshToZMapConverter::SetFillMode or EMeshToZMapConverter::SetFillModeMedian.
Fill mode is enabled by default. If fill mode is disabled, undefined pixels may remain in the EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EnableFillMode(
 bool state
 )
```

Parameters

state

Set to true to enable fill mode.

EMeshToZMapConverter::GetExtension

EMeshToZMapConverter::SetExtension

Sets a metric value used to enlarge the point cloud 3D domain.
That value affects X,Y and Z directions and can be used to generate an EZMap with borders of undefined pixels.
Default value is 0, which means a ZMap without border.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetExtension() const

void SetExtension(float size)
```

EMeshToZMapConverter::GetFillUndefinedPixelsDirection

Gets the undefined pixel fill direction (see EFillUndefinedPixelsDirection).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection
GetFillUndefinedPixelsDirection() const
```

EMeshToZMapConverter::GetFillUndefinedPixelsMethod

Gets the undefined pixel fill method (see EFillUndefinedPixelsMethod).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod GetFillUndefinedPixelsMethod
() const
```

EMeshToZMapConverter::IsFillModeEnabled

Tells if the fill mode is enabled or not.
Use EMeshToZMapConverter::EnableFillMode to toggle the fill mode and EMeshToZMapConverter::SetFillMode or EMeshToZMapConverter::SetFillModeMedian to set the filling parameters.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsFillModeEnabled(
 )
```

EMeshToZMapConverter::Load

Loads the converter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EMeshToZMapConverter::GetMapHeight

Gets the required height (number of pixels) of the generated EZMap.
By default, the required size is not set.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetMapHeight() const
```

EMeshToZMapConverter::GetMapWidth

Gets the required width (number of pixels) of the generated EZMap.
By default, the required size is not set.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetMapWidth() const
```

EMeshToZMapConverter::GetMapXResolution

Gets the resolution of the EZMap pixels along the X axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetMapXResolution() const
```

EMeshToZMapConverter::GetMapYResolution

Gets the resolution of the EZMap pixels along the Y axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetMapYResolution() const
```

EMeshToZMapConverter::GetMapZResolution

EMeshToZMapConverter::SetMapZResolution

Gets/sets the EZMap Z resolution, in world space units per gray value.
The resolution is used to compute the transformation of the world Z position to an integer 8, 16 or 32 bits pixel value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMapZResolution() const

void SetMapZResolution(float scale)
```

EMeshToZMapConverter::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EMeshToZMapConverter& operator=(
 const EMeshToZMapConverter& other
 )
```

Parameters

other

Reference to the EMeshToZMapConverter object used for the assignment.

EMeshToZMapConverter::operator==

Comparison operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator==(
 const EMeshToZMapConverter& other
 )
```

Parameters

other

Reference to the EMeshToZMapConverter object used for the comparison.

EMeshToZMapConverter::GetOrientationVector

EMeshToZMapConverter::SetOrientationVector

Sets an explicit orientation for the EZMap.
Overrides the orientation mode given by the method EMeshToZMapConverter::OrientationVectorMode.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]

E3DPoint GetOrientationVector() const

void SetOrientationVector(const E3DPoint& direction)
```

Remarks

The direction should be an E3DPoint representing the expected direction of the X (width) axis of the ZMap.
That direction will be used after projection on the reference plane normal.
That direction must NOT be aligned with the reference plane normal.

EMeshToZMapConverter::GetOrientationVectorMode

EMeshToZMapConverter::SetOrientationVectorMode

Chooses the EZMap orientation from a list of predefined axis, automatic mode or user defined vector.
Use EMeshToZMapConverter::OrientationVector to set an explicit orientation vector for the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]

Euresys::Open_eVision::Easy3D::EZMapOrientationVectorMode GetOrientationVectorMode() const

void SetOrientationVectorMode(Euresys::Open_eVision::Easy3D::EZMapOrientationVectorMode mode)
```

Remarks

Choose between Automatic mode (default), world space axis or explicit user defined vector (see EZMapOrientationVectorMode).

EMeshToZMapConverter::GetOrigin

EMeshToZMapConverter::SetOrigin

Chooses the EZMap origin. It is the 3D world position used as origin of the ZMap upper left pixel (0,0).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetOrigin() const

void SetOrigin(const E3DPoint& position)
```

Remarks

That position will be projected on the reference plane.
To let the conversion chooses for the origin, call EMeshToZMapConverter::UnsetOrigin.

EMeshToZMapConverter::GetReferencePlane

EMeshToZMapConverter::SetReferencePlane

Sets the E3DPlane reference plane.
The resulting EZMap is the distance of the 3D points above that plane.
3D points below the reference plane are discarded.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane GetReferencePlane() const

void SetReferencePlane(const E3DPlane& plane)
```

EMeshToZMapConverter::GetReferencePlaneMode

EMeshToZMapConverter::SetReferencePlaneMode

Sets an axis aligned reference plane.
Overrides the explicit reference plane given by the method EMeshToZMapConverter::ReferencePlane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EZMapReferencePlaneMode GetReferencePlaneMode() const
```

```
void SetReferencePlaneMode(Euresys::Open_eVision::Easy3D::EZMapReferencePlaneMode mode)
```

Remarks

Choose between X, Y or Z reference plane (see EZMapReferencePlaneMode).
The plane offset is set automatically on the point cloud lowest 3D point.

EMeshToZMapConverter::Save

Saves the converter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

Pointer to the ESerializer created for writing.

EMeshToZMapConverter::SetFillMode

Inpainting options used to fill the "holes" in the EZMap. A hole exists when no 3D point is projected at that pixel position in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFillMode(
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

direction

Direction in which the undefined pixels are filled in a depthmap from EFillUndefinedPixelsDirection

method

Which values used to fill the undefined pixels in a depthmap from EFillUndefinedPixelsMethod

EMeshToZMapConverter::SetFillModeMedian

Inpainting options used to fill the "holes" in the EZMap using a median rectangular kernel of odd size. A hole exists when no 3D point is projected at that pixel position in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetFillModeMedian(
 OEV_UINT32 halfKernelX,
 OEV_UINT32 halfKernelY
 )
```

Parameters

halfKernelX

-

halfKernelY

-

EMeshToZMapConverter::SetMapSize

Sets the required size of the generated EZMap; expressed in number of pixels for width and height dimensions.
By default, the required size is not set.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetMapSize(
 int width,
 int height
 )
```

Parameters

width

The required width for the Generated ZMap.

height

The required height for the Generated ZMap.

EMeshToZMapConverter::SetMapXYResolution

Sets the resolution (possibly anisotropic) of the EZMap pixels along the X and Y axes, in world space units per pixel (e.g mm/pixel).
The resolution is used to compute the ZMap size (width and height), depending on the projected point cloud on the reference plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetMapXYResolution(
 float resolution
 )
void SetMapXYResolution(
 float resolutionX,
 float resolutionY
 )
```

Parameters

resolution

The resolution for the isotropic case.

resolutionX

The resolution for the X axis.

resolutionY

The resolution for the Y axis.

Remarks

The isotropic scale, for X and Y axis is in metric world units.

EMeshToZMapConverter::UnsetMapSize

Unsets the resolution. Lets the conversion decides for the optimum resolution, depending on the projected point cloud and pixel scale.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetMapSize(
 )
```

EMeshToZMapConverter::UnsetMapXYResolution

Unsets the X and Y resolutions. Lets the conversion decide the optimal resolution, depending on the projected point cloud and EZMap size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetMapXYResolution(
 )
```

EMeshToZMapConverter::UnsetMapZResolution

Unsets the ZMap Z resolution. Lets the conversion decide the optimal Z resolution.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetMapZResolution(
 )
```

EMeshToZMapConverter::UnsetOrigin

Lets the conversion process decides for the EZMap origin position (based on projected point cloud on the reference plane).
Use EMeshToZMapConverter::Origin to enable and choose the ZMap origin.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetOrigin(
 )
```

EMeshToZMapConverter::UnsetWorldToZMapTransform

Disables the explicit world to ZMap transformation, set with EMeshToZMapConverter::WorldToZMapTransform.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetWorldToZMapTransform(
 )
```

EMeshToZMapConverter::GetWorldToZMapTransform

EMeshToZMapConverter::SetWorldToZMapTransform

Explicitly sets the world to ZMap transformation.
EMeshToZMapConverter::WorldToZMapTransform overrides the settings done by EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector and EMeshToZMapConverter::Origin.
That E3DTransformMatrix transform expresses how the world positions are transformed to the EZMap space.
The matrix must be a rigid transformation (translation and rotation only).
The resolution of the ZMap are defined by the EMeshToZMapConverter::SetMapXYResolution and EMeshToZMapConverter::MapZResolution methods.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetWorldToZMapTransform() const

void SetWorldToZMapTransform(const E3DTransformMatrix& matrix)
```

EMeshToZMapConverter::GetZMaptoWorldTransform

EMeshToZMapConverter::SetZMaptoWorldTransform

Explicitly sets the ZMap to World transformation.
"SetZMaptoWorldTransform" overrides the settings done by EMeshToZMapConverter::ReferencePlane, EMeshToZMapConverter::OrientationVector and EMeshToZMapConverter::Origin.
That E3DTransformMatrix transform expresses how the EZMap positions are transformed to the World space.
The matrix must be a rigid transformation (translation and rotation only).
The resolutions of the World are defined by the EMeshToZMapConverter::SetMapXYResolution and EMeshToZMapConverter::MapZResolution methods.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetZMaptoWorldTransform() const

void SetZMaptoWorldTransform(const E3DTransformMatrix& matrix)
```

4.162. EMovingAverage Class

Temporal integration of a number of images to reduce noise.

Namespace: Euresys::Open_eVision

Methods

Average	Performs averaging of the source image with the preceding ones, and computes the de-noised image.
EMovingAverage	Constructs an EMovingAverage object.
GetSize	Queries a moving average context for the current parameter settings.
GetSrcImage	Returns the image in which you must store the next image to be averaged.
Reset	Restarts the average process as if no image had ever been handed to the EMovingAverage object.
SetSize	Initializes a moving average context with appropriate parameters.

EMovingAverage::Average

Performs averaging of the source image with the preceding ones, and computes the de-noised image.

Namespace: Euresys::Open_eVision

[C++]

```
void Average(
 EROIBW8* destinationImage
 )

void Average(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage
 )
```

Parameters

destinationImage

Pointer to the destination image.

sourceImage

Pointer to the source image.

Remarks

The overload with only the destinationImage may be used only when the buffer allocation scheme has been set to internal (see EMovingAverage::SetSize or EMovingAverage::EMovingAverage)

EMovingAverage::EMovingAverage

Constructs an EMovingAverage object.

Namespace: Euresys::Open_eVision

[C++]

```
void EMovingAverage(
 )

void EMovingAverage(
 OEV_UINT32 period,
 int width,
 int height,
 bool internalAllocationScheme
 )
```

Parameters

period

Number of images on which to integrate. A power of 2 is recommended.

width

Image width (all images used for averaging must be of the same size).

height

Image height (all images used for averaging must be of the same size).

internalAllocationScheme

Buffer allocation scheme. When true, the moving average context provides the image to be acquired into (see member EMovingAverage::SrcImage).

Remarks

The default constructor constructs a void moving average context. A void moving average context has no internal buffers allocated and cannot be used for integration. Use the EMovingAverage::SetSize member after construction, or the initializing constructor instead. The sizing constructor constructs and initializes a moving average context.

EMovingAverage::GetSize

Queries a moving average context for the current parameter settings.

Namespace: Euresys::Open_eVision

[C++]

```
void GetSize(
 OEV_UINT32& numberOfImages,
 int& imageWidth,
 int& imageHeight,
 bool& internalAllocationScheme
 )
```

Parameters

numberOfImages

Number of images on which to integrate.

imageWidth

Image width (all images used for averaging must be of the same size).

imageHeight

Image height (all images used for averaging must be of the same size).

internalAllocationScheme

Buffer allocation scheme. When true, the moving average context provides the image to be acquired into (see member EMovingAverage::SrcImage).

EMovingAverage::Reset

Restarts the average process as if no image had ever been handed to the EMovingAverage object.

Namespace: Euresys::Open_eVision

[C++]

```
void Reset(
 )
```

Remarks

The behavior is thus the same as after a EMovingAverage::SetSize operation.

EMovingAverage::SetSize

Initializes a moving average context with appropriate parameters.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSize(
 OEV_UINT32 numberOfImages,
 int imageWidth,
 int imageHeight,
 bool internalAllocationScheme
 )
```

Parameters

numberOfImages

Number of images on which to integrate. A power of 2 is recommended.

imageWidth

Image width.

imageHeight

Image height.

internalAllocationScheme

Buffer allocation scheme. When true, the moving average context provides the image to be acquired into (see member EMovingAverage::SrcImage).

EMovingAverage::GetSrcImage

Returns the image in which you must store the next image to be averaged.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW8* GetSrcImage()
```

Remarks

This method may be used only when the buffer allocation scheme has been set to internal (see EMovingAverage::SetSize or EMovingAverage::EMovingAverage)

4.163. EObject Class

This class represents an object (blob) in an encoded image.

Remarks

This class inherits from the ECodedElement class and provides additional methods to access the holes of a particular object.

The extraction of the holes is lazy. This means that the holes are not computed before they get accessed. For this reason, the first access to the holes is slower than the subsequent accesses. On the other hand, the applications that do not make use of the holes are not penalized by the cost of hole extraction.

Base Class:ECodedElement

Namespace: Euresys::Open_eVision

Methods

EObject	-
GetHole	Returns a specified hole in the object.
GetHoleCount	Returns the number of holes in the object.

operator= -

EObject::EObject

-

Namespace: Euresys::Open_eVision

```
[C++]
void EObject(
 const EObject& other
 )
```

Parameters

other

-

EObject::GetHole

Returns a specified hole in the object.

Namespace: Euresys::Open_eVision

```
[C++]
const EHole& GetHole(
 OEV_UINT32 index
 )
EHole& GetHole(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the hole of interest.

EObject::GetHoleCount

Returns the number of holes in the object.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetHoleCount() const
```

EObject::operator=

-

Namespace: Euresys::Open_eVision

```
[C++]
EObject& operator=(
 const EObject& other
 )
```

Parameters

other

-

4.164. EObjectBasedCalibrationGenerator Class

Represents an object-based 3D calibration generator.
The class performs the computation of a calibration model based on the scan of the reference object.

Base Class:ECalibrationGenerator

Namespace: Euresys::Open_eVision::Easy3D

Methods

Compute	Computes an EObjectBasedCalibrationModel from the EDepthMap of the calibration object.
EObjectBasedCalibrationGenerator	Creates a EObjectBasedCalibrationGenerator.
GetCalibrationObjectScaleX	Returns the X axis scale of the calibration object. Deprecated: you should use @EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeA@ instead.
GetCalibrationObjectScaleY	Returns the Y axis scale of the calibration object. Deprecated: you should use @EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeB@ instead.
GetCalibrationObjectScaleZ	Returns the Z axis scale of the calibration object. Deprecated: you should use @EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeC@ instead.
GetCalibrationObjectSizeA	Returns the 'A' size of the calibration object.
GetCalibrationObjectSizeB	Returns the 'B' size of the calibration object.

GetCalibrationObjectSizeC	Returns the 'C' size of the calibration object.
GetCalibrationObjectType	Gets the EObjectBasedCalibrationType used for the computation of the EObjectBasedCalibrationModel. The type of the object used for the calibration is E3DObjectBasedCalibrationType_NotDefined by default.
GetNumCalibrationPasses	Sets/Gets the number of calibration passes. Each passes will refine the calibration model but the computation will be longer. The number of passes is 1 by default.
GetPrecisionVsSpeedTradeOff	Sets/Gets the trade-off between precision and speed for the calibration. The Precision vs speed trade-off mode from EObjectBasedCalibrationPrecisionVsSpeedTradeOff is EObjectBasedCalibrationPrecisionVsSpeedTradeOff_Balanced by default.
GetRangeX	Sets/Gets the 'X' axis range for the calibration object detection in the EDepthMap. If max value is smaller than min value, then max will not be used, we use depth map width - 1 instead. If min value is smaller than 0, then min will not be used, we use 0 instead.
GetRangeY	Sets/Gets the 'Y' axis range for the calibration object detection in the depth map. If max value is smaller than min value, then max will not be used, we use depth map height - 1 instead. If min value is smaller than 0, then min will not be used, we use 0 instead.
GetRangeZ	Sets/Gets the 'Z' axis range for the calibration object detection in the depth map. if max value is smaller than min value, then max will not be used, we use the maximum float value instead. if min value is smaller than 0, then min will not be used, we use 0 instead.
Load	Loads the parameters used by the object based calibration generator.
operator=	Assignment operator.
Save	Saves the parameters used by the object based calibration generator.

SetCalibrationObjectScale	Sets the scale of your target calibration model compared to a reference model. Refer to the user guide for the EObjectBasedCalibrationModel reference design. Use that value to set the unit of the calibrated positions in the point cloud. The scale will affect the world coordinates after conversion of the EDepthMap to EPointCloud. For example, if "1 reference unit" is equal to "10 millimeters", set "10" as scale value. Then applying the calibration will produce results in millimeters. Changing these values affect the calibration object sizes defined by EObjectBasedCalibrationGenerator::SetCalibrationObjectType. Deprecated: you should use @EObjectBasedCalibrationGenerator::SetCalibrationObjectType@ instead. Note that the scaleX/Y/Z and sizeA/B/C arguments aren't the same values.
SetCalibrationObjectType	Sets the EObjectBasedCalibrationType used for the computation of the EObjectBasedCalibrationModel. The type of the object used for the calibration is E3DObjectBasedCalibrationType_NotDefined by default.
SetNumCalibrationPasses	Sets/Gets the number of calibration passes. Each passes will refine the calibration model but the computation will be longer. The number of passes is 1 by default.
SetPrecisionVsSpeedTradeOff	Sets/Gets the trade-off between precision and speed for the calibration. The Precision vs speed trade-off mode from EObjectBasedCalibrationPrecisionVsSpeedTradeOff is EObjectBasedCalibrationPrecisionVsSpeedTradeOff_Balanced by default.
SetRangeX	Sets/Gets the 'X' axis range for the calibration object detection in the EDepthMap. If max value is smaller than min value, then max will not be used, we use depth map width - 1 instead. If min value is smaller than 0, then min will not be used, we use 0 instead.
SetRangeY	Sets/Gets the 'Y' axis range for the calibration object detection in the depth map. If max value is smaller than min value, then max will not be used, we use depth map height - 1 instead. If min value is smaller than 0, then min will not be used, we use 0 instead.
SetRangeZ	Sets/Gets the 'Z' axis range for the calibration object detection in the depth map. if max value is smaller than min value, then max will not be used, we use the maximum float value instead. if min value is smaller than 0, then min will not be used, we use 0 instead.

EObjectBasedCalibrationGenerator::GetCalibrationObjectScaleX

This property is deprecated.

Returns the X axis scale of the calibration object. Deprecated: you should use @EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeA@ instead.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationObjectScaleX() const
```

EObjectBasedCalibrationGenerator::GetCalibrationObjectScaleY

This property is deprecated.

Returns the Y axis scale of the calibration object. Deprecated: you should use @EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeB@ instead.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationObjectScaleY() const
```

EObjectBasedCalibrationGenerator::GetCalibrationObjectScaleZ

This property is deprecated.

Returns the Z axis scale of the calibration object. Deprecated: you should use @EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeC@ instead.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationObjectScaleZ() const
```

EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeA

Returns the 'A' size of the calibration object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationObjectSizeA() const
```

EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeB

Returns the 'B' size of the calibration object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationObjectSizeB() const
```

EObjectBasedCalibrationGenerator::GetCalibrationObjectSizeC

Returns the 'C' size of the calibration object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationObjectSizeC() const
```

EObjectBasedCalibrationGenerator::Compute

Computes an EObjectBasedCalibrationModel from the EDepthMap of the calibration object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EObjectBasedCalibrationModel Compute(
 const EDepthMap8& dm
 )

EObjectBasedCalibrationModel Compute(
 const EDepthMap16& dm
 )

EObjectBasedCalibrationModel Compute(
 const EDepthMap32f& dm
 )
```

Parameters

dm

The input depth map of the calibration object.

EObjectBasedCalibrationGenerator::EObjectBasedCalibrationGenerator

Creates a EObjectBasedCalibrationGenerator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EObjectBasedCalibrationGenerator(
 )
void EObjectBasedCalibrationGenerator(
 const EObjectBasedCalibrationGenerator& other
 )
```

Parameters

other

The other EObjectBasedCalibrationGenerator.

EObjectBasedCalibrationGenerator::GetCalibrationObjectType

Gets the EObjectBasedCalibrationType used for the computation of the EObjectBasedCalibrationModel.
The type of the object used for the calibration is E3DObjectBasedCalibrationType_NotDefined by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EObjectBasedCalibrationType GetCalibrationObjectType(
 )
```

EObjectBasedCalibrationGenerator::Load

Loads the parameters used by the object based calibration generator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EObjectBasedCalibrationGenerator::GetNumCalibrationPasses

EObjectBasedCalibrationGenerator::SetNumCalibrationPasses

Sets/Gets the number of calibration passes.
Each passes will refine the calibration model but the computation will be longer.
The number of passes is 1 by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumCalibrationPasses() const

void SetNumCalibrationPasses(int numPasses)
```

EObjectBasedCalibrationGenerator::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EObjectBasedCalibrationGenerator& operator=(
 const EObjectBasedCalibrationGenerator& other
 )
```

Parameters

other

The other EObjectBasedCalibrationGenerator.

EObjectBasedCalibrationGenerator::GetPrecisionVsSpeedTradeOff

EObjectBasedCalibrationGenerator::SetPrecisionVsSpeedTradeOff

Sets/Gets the trade-off between precision and speed for the calibration.
The Precision vs speed trade-off mode from EObjectBasedCalibrationPrecisionVsSpeedTradeOff is EObjectBasedCalibrationPrecisionVsSpeedTradeOff_Balanced by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EObjectBasedCalibrationPrecisionVsSpeedTradeOff
GetPrecisionVsSpeedTradeOff() const
```

```
void SetPrecisionVsSpeedTradeOff(Euresys::Open_
eVision::Easy3D::EObjectBasedCalibrationPrecisionVsSpeedTradeOff tradeOff)
```

EObjectBasedCalibrationGenerator::GetRangeX

EObjectBasedCalibrationGenerator::SetRangeX

Sets/Gets the 'X' axis range for the calibration object detection in the EDepthMap.
If max value is smaller than min value, then max will not be used, we use depth map width - 1 instead.
If min value is smaller than 0, then min will not be used, we use 0 instead.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EIntegerRange GetRangeX() const

void SetRangeX(const EIntegerRange& Range_X)
```

EObjectBasedCalibrationGenerator::GetRangeY

EObjectBasedCalibrationGenerator::SetRangeY

Sets/Gets the 'Y' axis range for the calibration object detection in the depth map.
If max value is smaller than min value, then max will not be used, we use depth map height - 1 instead.
If min value is smaller than 0, then min will not be used, we use 0 instead.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EIntegerRange GetRangeY() const

void SetRangeY(const EIntegerRange& Range_Y)
```

EObjectBasedCalibrationGenerator::GetRangeZ

EObjectBasedCalibrationGenerator::SetRangeZ

Sets/Gets the 'Z' axis range for the calibration object detection in the depth map.
if max value is smaller than min value, then max will not be used, we use the maximum float value instead.
if min value is smaller than 0, then min will not be used, we use 0 instead.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EFloatRange GetRangeZ() const
void SetRangeZ(const EFloatRange& Range_Z)
```

EObjectBasedCalibrationGenerator::Save

Saves the parameters used by the object based calibration generator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EObjectBasedCalibrationGenerator::SetCalibrationObjectScale

This method is deprecated.

Sets the scale of your target calibration model compared to a reference model.
Refer to the user guide for the EObjectBasedCalibrationModel reference design.
Use that value to set the unit of the calibrated positions in the point cloud.
The scale will affect the world coordinates after conversion of the EDepthMap to EPointCloud.
For example, if "1 reference unit" is equal to "10 millimeters", set "10" as scale value.
Then applying the calibration will produce results in millimeters. Changing these values affect the calibration object sizes defined by EObjectBasedCalibrationGenerator::SetCalibrationObjectType. Deprecated: you should use @EObjectBasedCalibrationGenerator::SetCalibrationObjectType@ instead. Note that the scaleX/Y/Z and sizeA/B/C arguments aren't the same values.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetCalibrationObjectScale(
 float scale
 )
```

```
void SetCalibrationObjectScale(
 float scaleX,
 float scaleY,
 float scaleZ
 )
```

Parameters

scale

Sets the same scale on axis X, Y and Z relative to the calibration object reference size.

scaleX

Sets the scale on X axis relative to the calibration object reference size.

scaleY

Sets the scale on Y axis relative to the calibration object reference size.

scaleZ

Sets the scale on Z axis relative to the calibration object reference size.

EObjectBasedCalibrationGenerator::SetCalibrationObjectType

Sets the EObjectBasedCalibrationType used for the computation of the EObjectBasedCalibrationModel.
The type of the object used for the calibration is E3DObjectBasedCalibrationType_NotDefined by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetCalibrationObjectType(
 Euresys::Open_eVision::Easy3D::EObjectBasedCalibrationType type
 )

void SetCalibrationObjectType(
 Euresys::Open_eVision::Easy3D::EObjectBasedCalibrationType type,
 float sizeA,
 float sizeB,
 float sizeC
 )
```

Parameters

type
Sets the type of object calibration to detect in the EDepthMap.

sizeA
Sets the size 'A' of object calibration (1 by default).

sizeB
Sets the size 'B' of object calibration (1 by default).

sizeC
Sets the size 'C' of object calibration (1 by default).

Remarks

'sizeA', 'sizeB', and 'sizeC' values are in metric unit. The resulting point cloud positions after applying the calibration will follow the same metric unit. Refer to the Open eVision user guide, Easy3D calibration section, for the definition of 'A', 'B' and 'C' dimensions.

4.165. EObjectBasedCalibrationModel Class

EObjectBasedCalibrationModel is an Easy3D calibration model built from a scan of a reference object.

Base Class:ECalibrationModel

Namespace: Euresys::Open_eVision::Easy3D

Methods

EObjectBasedCalibrationModel	Creates an EObjectBasedCalibrationModel.
GetCalibrationError	Returns the estimate of the calibration error (in metric units).
GetCalibrationRelativeError	Returns the estimate of the calibration error (in relative units).
GetType	Returns the type of calibration model, see ECalibrationType.
IsInitialized	Returns true if the model is initialized.
Load	Loads the model. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the model. The given ESerializer must have been created for writing.

EObjectBasedCalibrationModel::GetCalibrationError

Returns the estimate of the calibration error (in metric units).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationError() const
```

EObjectBasedCalibrationModel::GetCalibrationRelativeError

Returns the estimate of the calibration error (in relative units).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetCalibrationRelativeError() const
```

EObjectBasedCalibrationModel::EObjectBasedCalibrationModel

Creates an EObjectBasedCalibrationModel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EObjectBasedCalibrationModel(
 )
void EObjectBasedCalibrationModel(
 const EObjectBasedCalibrationModel& other
 )
```

Parameters

other

Another EObjectBasedCalibrationModel.

EObjectBasedCalibrationModel::IsInitialized

Returns true if the model is initialized.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsInitialized(
 )
```

EObjectBasedCalibrationModel::Load

Loads the model. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EObjectBasedCalibrationModel::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EObjectBasedCalibrationModel& operator=(
 const EObjectBasedCalibrationModel& other
 )
```

Parameters

other
Another EObjectBasedCalibrationModel.

EObjectBasedCalibrationModel::Save

Saves the model. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EObjectBasedCalibrationModel::GetType

Returns the type of calibration model, see ECalibrationType.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::ECalibrationType GetType() const
```

4.166. EObjectRunsIterator Class

Iterator to the runs of a coded element.

Remarks

This class is responsible for the sequential access to the individual runs of a coded element.

A run is defined as a maximal sequence of consecutive pixels on the same run, that all belong to the same coded element.

Namespace: Euresys::Open_eVision

Methods

EObjectRunsIterator	Constructs an iterator to the runs of a coded element.
First	Rewinds to the first run of the coded element.
GetEndX	Returns the abscissa (X-coordinate) of the rightmost pixel of the run the iterator is currently over.
GetLength	Returns the lengths of the run the iterator is currently over.
GetStartX	Returns the abscissa (X-coordinate) of the leftmost pixel of the run the iterator is currently over.
GetY	Returns the ordinate (Y-coordinate) of the run the iterator is currently over.
IsDone	Tests whether all the runs have been spanned.
Next	Advance to the next run in the iterator.
operator=	Assignment operator.

EObjectRunsIterator::GetEndX

Returns the abscissa (X-coordinate) of the rightmost pixel of the run the iterator is currently over.

Namespace: Euresys::Open_eVision

```
[C++]
int GetEndX() const
```

Remarks

An exception is thrown if the iterator has reached its end. Use EObjectRunsIterator::IsDone to check this condition.

EObjectRunsIterator::EObjectRunsIterator

Constructs an iterator to the runs of a coded element.

Namespace: Euresys::Open_eVision

```
[C++]
void EObjectRunsIterator(
 const ECodedElement& codedElement
 )

void EObjectRunsIterator(
 const EObjectRunsIterator& other
 )
```

Parameters

codedElement

The coded element of interest, in the case the iterator is to be constructed from a given coded element.

other

The iterator to be copied, in the case of the copy constructor.

EObjectRunsIterator::First

Rewinds to the first run of the coded element.

Namespace: Euresys::Open_eVision

```
[C++]
void First(
 )
```

EObjectRunsIterator::GetIsDone

Tests whether all the runs have been spanned.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsDone() const
```

EObjectRunsIterator::GetLength

Returns the lengths of the run the iterator is currently over.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetLength() const
```

Remarks

An exception is thrown if the iterator has reached its end. Use EObjectRunsIterator::IsDone to check this condition.

EObjectRunsIterator::Next

Advance to the next run in the iterator.

Namespace: Euresys::Open_eVision

[C++]

```
void Next(
 )
```

Remarks

An exception is thrown if the iterator has reached its end. Use EObjectRunsIterator::IsDone to check this condition.

EObjectRunsIterator::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EObjectRunsIterator& operator=(
 const EObjectRunsIterator& other
 )
```

Parameters

other

The iterator to be copied.

EObjectRunsIterator::GetStartX

Returns the abscissa (X-coordinate) of the leftmost pixel of the run the iterator is currently over.

Namespace: Euresys::Open_eVision

[C++]

```
int GetStartX() const
```

Remarks

An exception is thrown if the iterator has reached its end. Use EObjectRunsIterator::IsDone to check this condition.

EObjectRunsIterator::GetY

Returns the ordinate (Y-coordinate) of the run the iterator is currently over.

Namespace: Euresys::Open_eVision

[C++]

```
int GetY() const
```

Remarks

An exception is thrown if the iterator has reached its end. Use EObjectRunsIterator::IsDone to check this condition.

4.167. EObjectSelection Class

This container class handles the selection of a subset of coded elements taken from a coded image.

Remarks

This class provides methods to perform a selection of objects or holes and to retrieve the features of the coded elements in the collection.

Namespace: Euresys::Open_eVision

Methods

Add	Add a coded element to the selection.
AddHole	Adds a single hole of a coded image.

AddHoles	Adds all the holes contained in an object, in a layer or in a coded image.
AddHolesOfSelectedObjects	Adds the holes of all the objects that are currently selected.
AddLayer	Adds all the objects of a single layer in a coded image.
AddObject	Adds a single object of a layer in a coded image.
AddObjects	Adds all the objects of all the layers of a given coded image.
AddObjectsUsingFloatFeature	Selects the objects that fullfil a condition on a floating-point feature.
AddObjectsUsingIntegerFeature	Selects the objects that fullfil a condition on an integer feature.
AddObjectsUsingRectangle	Adds all the objects of a coded image whose location fullfil a criterion based on a rectangle.
AddObjectsUsingRegion	Adds all the objects of a coded image whose location fullfil a criterion based on an arbitrary region.
AddObjectsUsingUnsignedIntegerFeature	Selects the objects that fullfil a condition on an unsigned integer feature.
AddObjectUsingPosition	Adds the object of a coded image that lies at a particular location.
Clear	Remove all the coded elements that are contained in the selection.
ClearFeatureCache	Clears the internal cache for the computed features.
EObjectSelection	-
FeatureAverage	Computes the average of the values of the given feature across the selection.
FeatureDeviation	Computes the standard deviation of the values of the given feature across the selection.
FeatureVariance	Computes the variance of the values of the given feature across the selection.
FloatFeatureMaximum	Computes the maximum value of the given feature across the selection.
FloatFeatureMinimum	Computes the minimum value of the given feature across the selection.
GetAttachedImage	Sets the attached image for computing the features that depend on an image.
GetElement	Index-based access to the coded elements of the selection.
GetElementCount	Returns the number of coded elements selection.
GetFeretAngle	Angle of the Feret box (in the current angle units).
GetFloatFeature	Returns the value of a floating-point feature for a selected coded element.
GetIndexOfElement	Retrieves the index of a given coded element in the selection.
GetIntegerFeature	Returns the value of an integer feature for a selected coded element.

GetUnsignedIntegerFeature	Returns the value of an unsigned integer feature for a selected coded element.
IntegerFeatureMaximum	Computes the maximum value of the given feature across the selection.
IntegerFeatureMinimum	Computes the minimum value of the given feature across the selection.
IsSelected	Tests whether a given coded element is present in the selection.
operator==	Equality operator for selection. For two object selection to be equal, they must have the exact same set of object coming from the same coded images.
Remove	Remove a coded element from the selection.
RemoveHole	Removes a single hole of a coded image.
RemoveHoles	Removes all the holes contained in an object, in a layer or in a coded image.
RemoveLayer	Removes all the objects of a single layer in a coded image.
RemoveObject	Removes a single object of a layer in a coded image.
RemoveObjectsUsingRectangle	Removes all the objects of a coded image whose location fullfil a criterion based on a rectangle.
RemoveObjectsUsingRegion	Removes all the objects of a coded image whose location fullfil a criterion based on an arbitrary region.
RemoveObjectUsingPosition	Removes the object of a coded image that lies at a particular location.
RemoveSelectedHoles	Remove all the holes that are currently present in the selection.
RemoveUsingFloatFeature	Removes the selected coded elements that fullfil a condition on a floating-point feature.
RemoveUsingIntegerFeature	Removes the selected coded elements that fullfil a condition on an unsigned integer feature.
RemoveUsingUnsignedIntegerFeature	Removes the selected coded elements that fullfil a condition on an unsigned integer feature.
RenderMask	Creates a Flexible Mask from the selection.
SetAttachedImage	Sets the attached image for computing the features that depend on an image.
SetFeretAngle	Angle of the Feret box (in the current angle units).
Sort	Sorts the selected coded elements according to the value of a feature.
ToRegion	Converts the object selection to a region.
UnsignedIntegerFeatureMaximum	Computes the maximum value of the given feature across the selection.
UnsignedIntegerFeatureMinimum	Computes the minimum value of the given feature across the selection.

EObjectSelection::Add

Add a coded element to the selection.

Namespace: Euresys::Open_eVision

```
[C++]
void Add(
 const ECodedElement& element
 )
```

Parameters

element

The coded element.

EObjectSelection::AddHole

Adds a single hole of a coded image.

Namespace: Euresys::Open_eVision

```
[C++]
void AddHole(
 ECodedImage2& codedImage,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex
 )

void AddHole(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex
 )
```

Parameters

codedImage

The coded image.

objectIndex

The index of the parent object in the layer.

holeIndex

The index of the hole in the parent object.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::AddHoles

Adds all the holes contained in an object, in a layer or in a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void AddHoles(
 ECodedImage2& codedImage
 )

void AddHoles(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex
 )

void AddHoles(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, all the layers are taken into consideration.

objectIndex

The index of the parent object. If this parameter is left unspecified, all the objects are taken into consideration.

EObjectSelection::AddHolesOfSelectedObjects

Adds the holes of all the objects that are currently selected.

Namespace: Euresys::Open_eVision

```
[C++]
void AddHolesOfSelectedObjects(
 )
```

EObjectSelection::AddLayer

Adds all the objects of a single layer in a coded image.

Namespace: Euresys::Open_eVision

```
[C++]
void AddLayer(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex
 )
void AddLayer(
 ECodedImage2& codedImage
 )
```

Parameters

codedImage

The coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::AddObject

Adds a single object of a layer in a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObject(
 ECodedImage2& codedImage,
 OEV_UINT32 objectIndex
 )
void AddObject(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )
```

Parameters

codedImage

The coded image.

objectIndex

The index of the object in the layer.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::AddObjects

Adds all the objects of all the layers of a given coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObjects(
 ECodedImage2& image
 )
```

Parameters

image

The coded image.

EObjectSelection::AddObjectsUsingFloatFeature

Selects the objects that fullfil a condition on a floating-point feature.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObjectsUsingFloatFeature(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 Euresys::Open_eVision::EFeature feature,
 float threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )
void AddObjectsUsingFloatFeature(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 Euresys::Open_eVision::EFeature feature,
 float lowBound,
 float highBound,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
void AddObjectsUsingFloatFeature(
 ECodedImage2& codedImage,
 Euresys::Open_eVision::EFeature feature,
 float threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )
void AddObjectsUsingFloatFeature(
 ECodedImage2& codedImage,
 Euresys::Open_eVision::EFeature feature,
 float lowBound,
 float highBound,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
```

Parameters

codedImage
The source coded image.

layerIndex
The index of the layer of interest. If left unspecified, all the layers are taken into consideration.

feature
The feature that serves as a filter.

threshold
The single threshold on the feature.

mode
Specifies the way the threshold is interpreted.

lowBound
The low bound of the range on the feature.

highBound
The high bound of the range on the feature.

Remarks

Most features are floating-point. These methods thus tend to be the most widely used.

EObjectSelection::AddObjectsUsingIntegerFeature

Selects the objects that fullfil a condition on an integer feature.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObjectsUsingIntegerFeature(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 Euresys::Open_eVision::EFeature feature,
 int threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )

void AddObjectsUsingIntegerFeature(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 Euresys::Open_eVision::EFeature feature,
 int lowBound,
 int highBound,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )

void AddObjectsUsingIntegerFeature(
 ECodedImage2& codedImage,
 Euresys::Open_eVision::EFeature feature,
 int threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )

void AddObjectsUsingIntegerFeature(
 ECodedImage2& codedImage,
 Euresys::Open_eVision::EFeature feature,
 int lowBound,
 int highBound,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If left unspecified, all the layers are taken into consideration.

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

lowBound

The low bound of the range on the feature.

highBound

The high bound of the range on the feature.

EObjectSelection::AddObjectsUsingRectangle

Adds all the objects of a coded image whose location fullfil a criterion based on a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObjectsUsingRectangle(
 ECodedImage2& codedImage,
 int x,
 int y,
 OEV_UINT32 width,
 OEV_UINT32 height,
 Euresys::Open_eVision::ERectangleMode mode
 )

void AddObjectsUsingRectangle(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 int x,
 int y,
 OEV_UINT32 width,
 OEV_UINT32 height,
 Euresys::Open_eVision::ERectangleMode mode
 )
```

Parameters

codedImage
The source coded image.

x
The X-coordinate of the top-left corner of the selection rectangle.

y
The Y-coordinate of the top-left corner of the selection rectangle.

width
The width of the selection rectangle.

height
The height of the selection rectangle.

mode
The comparison mode with respect to the selection rectangle.

layerIndex
If specified, only the specified layer is taken into consideration.

EObjectSelection::AddObjectsUsingRegion

Adds all the objects of a coded image whose location fullfil a criterion based on an arbitrary region.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObjectsUsingRegion(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 ERegion& region,
 Euresys::Open_eVision::ERectangleMode mode
 )
void AddObjectsUsingRegion(
 ECodedImage2& codedImage,
 ERegion& region,
 Euresys::Open_eVision::ERectangleMode mode
 )
```

Parameters

codedImage
The source coded image.

layerIndex
If specified, only the specified layer is taken into consideration.

region
The region defining the geometric domain.

mode
The comparison mode with respect to the region.

EObjectSelection::AddObjectsUsingUnsignedIntegerFeature

Selects the objects that fullfil a condition on an unsigned integer feature.

Namespace: Euresys::Open_eVision

[C++]

```
void AddObjectsUsingUnsignedIntegerFeature(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 Euresys::Open_eVision::EFeature feature,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )

void AddObjectsUsingUnsignedIntegerFeature(
 ECodedImage2& codedImage,
 Euresys::Open_eVision::EFeature feature,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )

void AddObjectsUsingUnsignedIntegerFeature(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 Euresys::Open_eVision::EFeature feature,
 OEV_UINT32 lowBound,
 OEV_UINT32 highBound,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )

void AddObjectsUsingUnsignedIntegerFeature(
 ECodedImage2& codedImage,
 Euresys::Open_eVision::EFeature feature,
 OEV_UINT32 lowBound,
 OEV_UINT32 highBound,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If left unspecified, all the layers are taken into consideration.

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

lowBound
The low bound of the range on the feature.
highBound
The high bound of the range on the feature.

EObjectSelection::AddObjectUsingPosition

Adds the object of a coded image that lies at a particular location.

Namespace: Euresys::Open_eVision

```
[C++]
void AddObjectUsingPosition(
 ECodedImage2& codedImage,
 int x,
 int y
 )
```

Parameters

codedImage
The source coded image.
x
The X-coordinate of the object.
y
The Y-coordinate of the object.

Remarks

If no object lies at the specified coordinate, the selection is not changed.

EObjectSelection::GetAttachedImage

EObjectSelection::SetAttachedImage

Sets the attached image for computing the features that depend on an image.

Namespace: Euresys::Open_eVision

```
[C++]
const EROIBW8* GetAttachedImage() const
void SetAttachedImage(const EROIBW8* image)
```

Remarks

An image must be attached before dealing with the following features: EFeature_PixelMin, EFeature_PixelMax, EFeature_WeightedGravityCenterX, EFeature_WeightedGravityCenterY, EFeature_PixelGrayAverage, EFeature_PixelGrayVariance, EFeature_PixelGrayDeviation.

EObjectSelection::Clear

Remove all the coded elements that are contained in the selection.

Namespace: Euresys::Open_eVision

```
[C++]
void Clear(
 )
```

EObjectSelection::ClearFeatureCache

Clears the internal cache for the computed features.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearFeatureCache(
 )
```

Remarks

This is useful to reduce memory consumption.

EObjectSelection::GetElementCount

Returns the number of coded elements selection.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetElementCount() const
```

EObjectSelection::EObjectSelection

-

Namespace: Euresys::Open_eVision

```
[C++]
void EObjectSelection(
 )
```

EObjectSelection::FeatureAverage

Computes the average of the values of the given feature across the selection.

Namespace: Euresys::Open_eVision

```
[C++]
float FeatureAverage(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

EObjectSelection::FeatureDeviation

Computes the standard deviation of the values of the given feature across the selection.

Namespace: Euresys::Open_eVision

```
[C++]
float FeatureDeviation(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

EObjectSelection::FeatureVariance

Computes the variance of the values of the given feature across the selection.

Namespace: Euresys::Open_eVision

```
[C++]
float FeatureVariance(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

EObjectSelection::GetFeretAngle

EObjectSelection::SetFeretAngle

Angle of the Feret box (in the current angle units).

Namespace: Euresys::Open_eVision

[C++]

```
float GetFeretAngle() const
void SetFeretAngle(float angle)
```

Remarks

A Feret angle must be set for the following features: EFeature_FeretBoxCenterX, EFeature_FeretBoxCenterY, EFeature_FeretBoxWidth, EFeature_FeretBoxHeight.

EObjectSelection::FloatFeatureMaximum

Computes the maximum value of the given feature across the selection.

Namespace: Euresys::Open_eVision

[C++]

```
float FloatFeatureMaximum(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

Remarks

Most features are floating-point. This method thus tends to be the most widely used.

EObjectSelection::FloatFeatureMinimum

Computes the minimum value of the given feature across the selection.

Namespace: Euresys::Open_eVision

[C++]

```
float FloatFeatureMinimum(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

Remarks

Most features are floating-point. This method thus tends to be the most widely used.

EObjectSelection::GetElement

Index-based access to the coded elements of the selection.

Namespace: Euresys::Open_eVision

[C++]

```
ECodedElement& GetElement(
 OEV_UINT32 index
 )

const ECodedElement& GetElement(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the coded element of interest.

EObjectSelection::GetFloatFeature

Returns the value of a floating-point feature for a selected coded element.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFloatFeature(
 OEV_UINT32 index,
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

index

The index of the selected coded element.

feature

The feature of interest.

Remarks

Most features are floating-point. This method thus tends to be the most widely used.

EObjectSelection::GetIndexOfElement

Retrieves the index of a given coded element in the selection.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetIndexOfElement(
 const ECodedElement& element
 )
```

Parameters

element

The coded element of interest.

Remarks

An exception is thrown if the coded element is not present in the selection.

EObjectSelection::GetIntegerFeature

Returns the value of an integer feature for a selected coded element.

Namespace: Euresys::Open_eVision

```
[C++]
int GetIntegerFeature(
 OEV_UINT32 index,
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

index

The index of the selected coded element.

feature

The feature of interest.

EObjectSelection::GetUnsignedIntegerFeature

Returns the value of an unsigned integer feature for a selected coded element.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetUnsignedIntegerFeature(
 OEV_UINT32 index,
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

index
The index of the selected coded element.

feature
The feature of interest.

EObjectSelection::IntegerFeatureMaximum

Computes the maximum value of the given feature across the selection.

Namespace: Euresys::Open_eVision

[C++]

```
int IntegerFeatureMaximum(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature
The feature of interest.

EObjectSelection::IntegerFeatureMinimum

Computes the minimum value of the given feature across the selection.

Namespace: Euresys::Open_eVision

[C++]

```
int IntegerFeatureMinimum(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature
The feature of interest.

EObjectSelection::IsSelected

Tests whether a given coded element is present in the selection.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsSelected(
 const ECodedElement& element
 )

bool IsSelected(
 ECodedImage2& codedImage,
 OEV_UINT32 objectIndex
 )

bool IsSelected(
 const ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )

bool IsSelected(
 const ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex
 )
```

Parameters

element

The coded element.

codedImage

The coded image.

objectIndex

The index of the object in the layer of interest.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

holeIndex

If specified, the index of the hole in the object. If unspecified, one tests the presence of the object.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::operator==

Equality operator for selection. For two object selection to be equal, they must have the exact same set of object coming from the same coded images.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const EObjectSelection& other
 )
```

Parameters

other

Object selection to compare to.

EObjectSelection::Remove

Remove a coded element from the selection.

Namespace: Euresys::Open_eVision

[C++]

```
void Remove(
 const ECodedElement& element
 )
```

Parameters

element

The coded element.

EObjectSelection::RemoveHole

Removes a single hole of a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveHole(
 ECodedImage2& codedImage,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex
 )

void RemoveHole(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex,
 OEV_UINT32 holeIndex
 )
```

Parameters

codedImage

The coded image.

objectIndex

The index of the parent object in the layer.

holeIndex

The index of the hole in the parent object.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::RemoveHoles

Removes all the holes contained in an object, in a layer or in a coded image.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveHoles(
 ECodedImage2& codedImage
 )

void RemoveHoles(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex
 )

void RemoveHoles(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )
```

Parameters

codedImage

The source coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, all the layers are taken into consideration.

objectIndex

The index of the parent object. If this parameter is left unspecified, all the objects are taken into consideration.

EObjectSelection::RemoveLayer

Removes all the objects of a single layer in a coded image.

Namespace: Euresys::Open_eVision

```
[C++]

void RemoveLayer(
 ECodedImage2& codedImage
 )

void RemoveLayer(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex
 )
```

Parameters

codedImage

The coded image.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::RemoveObject

Removes a single object of a layer in a coded image.

Namespace: Euresys::Open_eVision

```
[C++]

void RemoveObject(
 ECodedImage2& codedImage,
 OEV_UINT32 objectIndex
 )

void RemoveObject(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 OEV_UINT32 objectIndex
 )
```

Parameters

codedImage

The coded image.

objectIndex

The index of the object in the layer.

layerIndex

The index of the layer of interest. If this parameter is left unspecified, the default layer will be taken into consideration.

Remarks

This methods throws an exception if no layer index is specified and if, simultaneously, the coded image contains several layers. Indeed, in such a case, no default layer exists.

EObjectSelection::RemoveObjectsUsingRectangle

Removes all the objects of a coded image whose location fullfil a criterion based on a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveObjectsUsingRectangle(
 ECodedImage2& codedImage,
 int x,
 int y,
 OEV_UINT32 width,
 OEV_UINT32 height,
 Euresys::Open_eVision::ERectangleMode mode
 )
void RemoveObjectsUsingRectangle(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 int x,
 int y,
 OEV_UINT32 width,
 OEV_UINT32 height,
 Euresys::Open_eVision::ERectangleMode mode
 )
```

Parameters

codedImage
The source coded image.

x
The X-coordinate of the top-left corner of the selection rectangle.

y
The Y-coordinate of the top-left corner of the selection rectangle.

width
The width of the selection rectangle.

height
The height of the selection rectangle.

mode
The comparison mode with respect to the selection rectangle.

layerIndex
If specified, only the specified layer is taken into consideration.

EObjectSelection::RemoveObjectsUsingRegion

Removes all the objects of a coded image whose location fullfil a criterion based on an arbitrary region.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveObjectsUsingRegion(
 ECodedImage2& codedImage,
 OEV_UINT32 layerIndex,
 ERegion& region,
 Euresys::Open_eVision::ERectangleMode mode
 )
void RemoveObjectsUsingRegion(
 ECodedImage2& codedImage,
 ERegion& region,
 Euresys::Open_eVision::ERectangleMode mode
 )
```

Parameters

codedImage
The source coded image.

layerIndex
If specified, only the specified layer is taken into consideration.

region
The region defining the geometric domain.

mode
The comparison mode with respect to the region.

EObjectSelection::RemoveObjectUsingPosition

Removes the object of a coded image that lies at a particular location.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveObjectUsingPosition(
 ECodedImage2& codedImage,
 int x,
 int y
 )
```

Parameters

codedImage

The source coded image.

x

The X-coordinate of the object.

y

The Y-coordinate of the object.

Remarks

If no object lies at the specified coordinate, the selection is not changed.

EObjectSelection::RemoveSelectedHoles

Remove all the holes that are currently present in the selection.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveSelectedHoles(
 )
```

EObjectSelection::RemoveUsingFloatFeature

Removes the selected coded elements that fullfil a condition on a floating-point feature.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveUsingFloatFeature(
 Euresys::Open_eVision::EFeature feature,
 float threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )
```

```
void RemoveUsingFloatFeature(
 Euresys::Open_eVision::EFeature feature,
 float low,
 float high,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
```

Parameters

feature

The feature that serves as a filter.

threshold

The single threshold on the feature.

mode

Specifies the way the threshold is interpreted.

low

The low bound of the range on the feature.

high

The high bound of the range on the feature.

Remarks

Most features are floating-point. These methods thus tend to be the most widely used.

EObjectSelection::RemoveUsingIntegerFeature

Removes the selected coded elements that fullfil a condition on an unsigned integer feature.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveUsingIntegerFeature(
 Euresys::Open_eVision::EFeature feature,
 int threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )

void RemoveUsingIntegerFeature(
 Euresys::Open_eVision::EFeature feature,
 int low,
 int high,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
```

Parameters

feature
The feature that serves as a filter.
threshold
The single threshold on the feature.
mode
Specifies the way the threshold is interpreted.
low
The low bound of the range on the feature.
high
The high bound of the range on the feature.

EObjectSelection::RemoveUsingUnsignedIntegerFeature

Removes the selected coded elements that fullfil a condition on an unsigned integer feature.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveUsingUnsignedIntegerFeature(
 Euresys::Open_eVision::EFeature feature,
 OEV_UINT32 threshold,
 Euresys::Open_eVision::ESingleThresholdMode mode
 )
void RemoveUsingUnsignedIntegerFeature(
 Euresys::Open_eVision::EFeature feature,
 OEV_UINT32 low,
 OEV_UINT32 high,
 Euresys::Open_eVision::EDoubleThresholdMode mode
 )
```

Parameters

feature
The feature that serves as a filter.
threshold
The single threshold on the feature.
mode
Specifies the way the threshold is interpreted.
low
The low bound of the range on the feature.
high
The high bound of the range on the feature.

EObjectSelection::RenderMask

Creates a Flexible Mask from the selection.

Namespace: Euresys::Open_eVision

```
[C++]
void RenderMask(
 EROIBW8& result,
 int offsetX,
 int offsetY
 )
void RenderMask(
 EROIBW8& result
 )
```

Parameters

result

The image in which the generated mask will be stored.

offsetX

The X-offset that must be applied to bring the zero X-coordinate in the coded image on the first column of the result image (defaults to zero).

offsetY

The Y-offset that must be applied to bring the zero Y-coordinate in the coded image on the first row of the result image (defaults to zero).

Remarks

The size of the result image will not be changed: It must be properly sized beforehand.

This method generates an exception if several layers are encoded, in which case no default layer exists.

EObjectSelection::Sort

Sorts the selected coded elements according to the value of a feature.

Namespace: Euresys::Open_eVision

```
[C++]
void Sort(
 Euresys::Open_eVision::EFeature feature
 )
void Sort(
 Euresys::Open_eVision::EFeature feature,
 Euresys::Open_eVision::ESortDirection direction
 )
```

Parameters

feature

The feature to sort with.

direction

The sorting direction. By default, the features are sorted by increasing values.

EObjectSelection::ToRegion

Converts the object selection to a region.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion ToRegion(
 )
```

EObjectSelection::UnsignedIntegerFeatureMaximum

Computes the maximum value of the given feature across the selection.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 UnsignedIntegerFeatureMaximum(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

EObjectSelection::UnsignedIntegerFeatureMinimum

Computes the minimum value of the given feature across the selection.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 UnsignedIntegerFeatureMinimum(
 Euresys::Open_eVision::EFeature feature
 )
```

Parameters

feature

The feature of interest.

4.168. EObjectTemplateMatcher Class

The EObjectTemplateMatcher class is a tool designed to align and match the output of EasyObject to a reference template.

Namespace: Euresys::Open_eVision

Methods

BuildTemplate	Set the reference template from a previous EasyObject coded image, an object selection or a list of known positions.
EObjectTemplateMatcher	Creates an EObjectTemplateMatcher object.
GetEnableAlignment	Enable an optional preliminary global alignement (rigid transformation only: translation and rotation). Choose to enable alignment only if the selection is not aligned with the template.
GetMaximumDistance	Defines the absolute maximum distance for a match to be valid. That value is used during the closest object search. By default, infinite distance is used.
GetNumberOfPairedObjects	Return the number of paired objects
GetSelectionIndexes	For each object in the **TEMPLATE** return the paired index in the SELECTION. if the object has no match, the value -1 is used.
GetTemplateIndexes	For each object in the **SELECTION** return the paired index in the TEMPLATE. If the object has no match, the value -1 is used.
GetUnpairedObjects	Return lists of object indexes that appear only in template or only in selection.
Load	Load the EObjectTemplateMatcher configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Save the EObjectTemplateMatcher configuration. The given ESerializer must have been created for writing.
SetEnableAlignment	Enable an optional preliminary global alignement (rigid transformation only: translation and rotation). Choose to enable alignment only if the selection is not aligned with the template.
SetMaximumDistance	Defines the absolute maximum distance for a match to be valid. That value is used during the closest object search. By default, infinite distance is used.

SortPositions	Align and match the given position array with the reference template. The shortest distance between the positions is used to pair the given positions with the template's positions. After the execution of this function, the template indexes corresponding to the position array are returned by method EObjectTemplateMatcher. The position indexes for template's positions are returned by method EObjectTemplateMatcher. Indexes of positions that appears only in template or only in the given array are returned by method EObjectTemplateMatcher::GetUnpairedObjects. The complexity is O(n) where n is the number of elements in the given array.
SortSelection	Align and match the given selection with the reference template. The shortest distance between object's centers is used to pair the objects in the selection with the objects in the template. After the execution of this function, the template indexes corresponding to selection objects are returned by method EObjectTemplateMatcher. The selection indexes for template objects are returned by method EObjectTemplateMatcher. Indexes of objects that appears only in template or only in selection are returned by method EObjectTemplateMatcher::GetUnpairedObjects. The complexity is O(n) where n is the number of objects in selection.

EObjectTemplateMatcher::BuildTemplate

Set the reference template from a previous EasyObject coded image, an object selection or a list of known positions.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildTemplate(
 EObjectSelection& selection
 )

void BuildTemplate(
 ECodedImage2& objects
 )

void BuildTemplate(
 const std::vector<Euresys::Open_eVision::EPoint>& positions
 )
```

Parameters

selection

The object selection to be used as template.

objects

The coded image to be used as template.

positions

The list of positions to be used as template.

EObjectTemplateMatcher::GetEnableAlignment

EObjectTemplateMatcher::SetEnableAlignment

Enable an optional preliminary global alignement (rigid transformation only: translation and rotation). Choose to enable alignment only if the selection is not aligned with the template.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableAlignment() const

void SetEnableAlignment(bool align)
```

EObjectTemplateMatcher::EObjectTemplateMatcher

Creates an EObjectTemplateMatcher object.

Namespace: Euresys::Open_eVision

[C++]

```
void EObjectTemplateMatcher(
 )

void EObjectTemplateMatcher(
 const EObjectTemplateMatcher& other
 )
```

Parameters

other

Reference to the EObjectTemplateMatcher used for the initialization.

EObjectTemplateMatcher::GetUnpairedObjects

Return lists of object indexes that appear only in template or only in selection.

Namespace: Euresys::Open_eVision

[C++]

```
void GetUnpairedObjects(
 std::vector<int>& onlyInTemplate,
 std::vector<int>& onlyInSelection
 )
```

Parameters

onlyInTemplate
The list of object indexes that appear only in the template.

onlyInSelection
The list of object indexes that appear only in the selection.

EObjectTemplateMatcher::Load

Load the EObjectTemplateMatcher configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EObjectTemplateMatcher::GetMaximumDistance

EObjectTemplateMatcher::SetMaximumDistance

Defines the absolute maximum distance for a match to be valid. That value is used during the closest object search. By default, infinite distance is used.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaximumDistance() const

void SetMaximumDistance(float distance)
```

EObjectTemplateMatcher::GetNumberOfPairedObjects

Return the number of paired objects

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumberOfPairedObjects() const
```

EObjectTemplateMatcher::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EObjectTemplateMatcher& operator=(
 const EObjectTemplateMatcher& other
 )
```

Parameters

other

The EObjectTemplateMatcher object that should be copied.

EObjectTemplateMatcher::Save

Save the EObjectTemplateMatcher configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EObjectTemplateMatcher::GetSelectionIndexes

For each object in the **TEMPLATE** return the paired index in the SELECTION. if the object has no match, the value -1 is used.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<int> GetSelectionIndexes() const
```

EObjectTemplateMatcher::SortPositions

Align and match the given position array with the reference template.
The shortest distance between the positions is used to pair the given positions with the template's positions. After the execution of this function, the template indexes corresponding to the position array are returned by method EObjectTemplateMatcher.
The position indexes for template's positions are returned by method EObjectTemplateMatcher.
Indexes of positions that appears only in template or only in the given array are returned by method EObjectTemplateMatcher::GetUnpairedObjects.
The complexity is O(n) where n is the number of elements in the given array.

Namespace: Euresys::Open_eVision

[C++]

```
void SortPositions(
 const std::vector<Euresys::Open_eVision::EPoint>& positions
 )
```

Parameters

positions

The array of positions to be compared with the template.

EObjectTemplateMatcher::SortSelection

Align and match the given selection with the reference template.
The shortest distance between object's centers is used to pair the objects in the selection with the objects in the template. After the execution of this function, the template indexes corresponding to selection objects are returned by method EObjectTemplateMatcher.
The selection indexes for template objects are returned by method EObjectTemplateMatcher.
Indexes of objects that appears only in template or only in selection are returned by method EObjectTemplateMatcher::GetUnpairedObjects.
The complexity is O(n) where n is the number of objects in selection.

Namespace: Euresys::Open_eVision

[C++]

```
void SortSelection(
 const EObjectSelection& selection
 )
```

Parameters

selection

The object selection to be compared. The selection is unchanged.

EObjectTemplateMatcher::GetTemplateIndexes

For each object in the **SELECTION** return the paired index in the TEMPLATE. If the object has no match, the value -1 is used.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<int> GetTemplateIndexes() const
```

4.169. EOCR Class

Manages a complete context for the font-dependent printed character reader implemented in EasyOCR.

Namespace: Euresys::Open_eVision

Methods

AddChar	Add a character coordinates to the list.
AddPatternFromImage	Adds a new pattern to the font.
BuildObjects	Segments the source image, i.e. detects and labels the objects.
CharGetDstX	Returns the abscissa of the lower right corner of a recognized character.
CharGetDstY	Returns the ordinate of the lower right corner of a recognized character.
CharGetHeight	Returns the height of a recognized character.
CharGetOrgX	Returns the abscissa of the upper left corner of a recognized character.
CharGetOrgY	Returns the ordinate of the upper left corner of a recognized character.
CharGetTotalDstX	Returns the abscissa of the lower right corner of a recognized character.
CharGetTotalDstY	Returns the ordinate of the lower right corner of a recognized character.
CharGetTotalOrgX	Returns the abscissa of the upper left corner of a recognized character.
CharGetTotalOrgY	Returns the ordinate of the upper left corner of a recognized character.
CharGetWidth	Returns the width of a recognized character.
DrawChar	Draws the bounding box of a given character.
DrawChars	Draws the bounding boxes of all characters in the image.
DrawCharsWithCurrentPen	Draws the bounding boxes of all characters in the image.

DrawCharWithCurrentPen	Draws the bounding box of a given character.
DrawObjects	-
EmptyChars	Empties the list of known characters.
EOCR	Constructs an OCR context.
FindAllChars	Locates the characters by filtering the objects according to their size, and grouping them if the segmentation mode is set to ESegmentationMode_RepasteObjects.
GetCharSpacing	Spacing that separates characters.
GetCompareAspectRatio	Flag indicating whether the character aspect ratio is used to compute the recognition score.
GetConfidenceRatio	Returns the degree of confidence in the recognized character.
GetCutLargeChars	Flag indicating whether the large chars cutting mode is used.
GetFirstCharCode	Returns the code of the pattern that matches at best a recognized character.
GetFirstCharDistance	Computes the degree of similarity between the best matching pattern and a recognized character.
GetLineSpacingMode	Line spacing mode to sort the character boxes into lines. Default mode is ELineSpacingMode.
GetMatchingMode	Matching mode to use to compare characters to the template.
GetMaxCharHeight	Maximum character height.
GetMaxCharWidth	Maximum character width.
GetMinCharHeight	Minimum character height.
GetMinCharWidth	Minimum character width.
GetNoiseArea	Noise area.
GetNumChars	Number of recognized characters.
GetNumPatterns	Number of patterns in the current font.
GetPatternBitmap	Returns a pointer to an image holding the pattern of the given index. This image should not be modified.
GetPatternClass	Returns the class of a given pattern in the current font.
GetPatternCode	Returns the character code of a given pattern in the current font.
GetPatternHeight	Current pattern height, as set at the last EOCR::NewFont or EOCR::Load operation.
GetPatternWidth	Current pattern width, as set at the last EOCR::NewFont or EOCR::Load operation.
GetRelativeSpacing	Relative spacing value.
GetRelativeThreshold	Relative threshold used for image segmentation.
GetRemoveBorder	Flag indicating whether all blobs touching the ROI edges are automatically discarded.

GetRemoveNarrowOrFlat	Flag indicating whether the characters are discarded when either dimension falls below the minimum value
GetSecondCharCode	Returns the code of the pattern that matches at second best a recognized character.
GetSecondCharDistance	Computes the degree of similarity between the second best matching pattern and a recognized character.
GetSegmentationMode	Segmentation mode.
GetShiftingMode	Shifting mode to use to compare characters to the template.
GetShiftXTolerance	Horizontal translation tolerance around the nominal position when the character positions are explicitly specified.
GetShiftYTolerance	Vertical translation tolerance around the nominal position when the character positions are explicitly specified.
GetTextColor	Text color.
GetThreshold	Threshold mode used for image segmentation.
GetTrueThreshold	Absolute threshold level when using a single threshold.
HitChar	Returns true if the cursor is placed over the character specified by charIndex.
HitChars	Returns true if the cursor is placed over a character and sets the charIndex accordingly.
LearnPattern	Adds a new pattern to the font.
LearnPatterns	Adds all the patterns from the source image to the font.
Load	Loads the EOCR object configuration.
MatchChar	Reads a single character, i.e. finds the best match between the patterns in the font and the given character.
NewFont	Empties the contents of the font and sets the size of the pattern array to be used from then on.
operator=	Copies all the data from another EOCR object into the current EOCR object
ReadText	Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.
ReadTextWide	DEPRECATED: Use the EOCR::ReadText method instead as it performs similarly to this method, except it returns UTF-8 instead of wide characters. Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.
Recognize	Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.
RecognizeWide	DEPRECATED: use the EOCR::Recognize method instead as it performs similarly to this method, except it returns UTF-8 instead of wide characters. Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.
RemovePattern	Removes a given pattern from the current font.
Save	Saves the EOCR object configuration.

SetCharSpacing	Spacing that separates characters.
SetCompareAspectRatio	Flag indicating whether the character aspect ratio is used to compute the recognition score.
SetCutLargeChars	Flag indicating whether the large chars cutting mode is used.
SetLineSpacingMode	Line spacing mode to sort the character boxes into lines. Default mode is ELineSpacingMode.
SetMatchingMode	Matching mode to use to compare characters to the template.
SetMaxCharHeight	Maximum character height.
SetMaxCharWidth	Maximum character width.
SetMinCharHeight	Minimum character height.
SetMinCharWidth	Minimum character width.
SetNoiseArea	Noise area.
SetPatternClass	Sets the class of a given pattern in the current font.
SetPatternCode	Sets the character code of a given pattern in the current font.
SetRelativeSpacing	Relative spacing value.
SetRelativeThreshold	Relative threshold used for image segmentation.
SetRemoveBorder	Flag indicating whether all blobs touching the ROI edges are automatically discarded.
SetRemoveNarrowOrFlat	Flag indicating whether the characters are discarded when either dimension falls below the minimum value
SetSegmentationMode	Segmentation mode.
SetShiftingMode	Shifting mode to use to compare characters to the template.
SetShiftXTolerance	Horizontal translation tolerance around the nominal position when the character positions are explicitly specified.
SetShiftYTolerance	Vertical translation tolerance around the nominal position when the character positions are explicitly specified.
SetTextColor	Text color.
SetThreshold	Threshold mode used for image segmentation.

EOCR::AddChar

Add a character coordinates to the list.

Namespace: Euresys::Open_eVision

[C++]

```
void AddChar(
 int originX,
 int originY,
 int endX,
 int endY
 )
```

Parameters

originX

Abscissa of the upper left corner of the bounding box of the character.

originY

Ordinate of the upper left corner of the bounding box of the character.

endX

Abscissa of the bottom right corner of the bounding box of the character.

endY

Ordinate of the bottom right corner of the bounding box of the character.

Remarks

It is to use when you know the exact position of the characters to be recognized, to bypass the segmentation step, for efficiency or reliability purposes. To use this feature, simply specify the character positions by successive calls to AddChar. When this is done, the remainder of the OCR processing steps can take place. To empty the list of known characters, call EOCR::EmptyChars.

EOCR::AddPatternFromImage

Adds a new pattern to the font.

Namespace: Euresys::Open_eVision

[C++]

```
void AddPatternFromImage(
 EROIBW8* sourceImage,
 int originX,
 int originY,
 int width,
 int height,
 int code,
 OEV_UINT32 classes
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.
originX
Abscissa of the upper left corner of the bounding box of the pattern.
originY
Ordinate of the upper left corner of the bounding box of the pattern.
width
Width of the bounding box of the pattern.
height
Height of the bounding box of the pattern.
code
Character code of the pattern.
classes
Class of the pattern, as defined by EOCRClass.

Remarks

The pattern is extracted from an image by specifying a bounding rectangle.

EOCR::BuildObjects

Segments the source image, i.e. detects and labels the objects.

Namespace: Euresys::Open_eVision

[C++]

```
void BuildObjects(
 EROIBW8* sourceImage
 )
```

Parameters

sourceImage
Pointer to the source image/ROI.

Remarks

An object is a connected set of pixels above or below Threshold (according to TextColor).

EOCR::CharGetDstX

Returns the abscissa of the lower right corner of a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int CharGetDstX(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::CharGetDstY

Returns the ordinate of the lower right corner of a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int CharGetDstY(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::CharGetHeight

Returns the height of a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int CharGetHeight(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::CharGetOrgX

Returns the abscissa of the upper left corner of a recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
int CharGetOrgX(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::CharGetOrgY

Returns the ordinate of the upper left corner of a recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
int CharGetOrgY(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::CharGetTotalDstX

Returns the abscissa of the lower right corner of a recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
int CharGetTotalDstX(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR::CharGetTotalDstY

Returns the ordinate of the lower right corner of a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int CharGetTotalDstY(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR::CharGetTotalOrgX

Returns the abscissa of the upper left corner of a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int CharGetTotalOrgX(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR::CharGetTotalOrgY

Returns the ordinate of the upper left corner of a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int CharGetTotalOrgY(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

The coordinates are computed with respect to the parent image rather than the ROI.

EOCR::CharGetWidth

Returns the width of a recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
int CharGetWidth(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::GetCharSpacing

EOCR::SetCharSpacing

Spacing that separates characters.

Namespace: Euresys::Open_eVision

[C++]

```
int GetCharSpacing()
void SetCharSpacing(int n32CharSpacing)
```

Remarks

When two objects are separated by a vertical gap larger or equal than the spacing, they are considered to belong to distinct characters. Note that two objects can still be separated if their merging will be wider than EOCR::MaxCharWidth. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetCompareAspectRatio

EOCR::SetCompareAspectRatio

Flag indicating whether the character aspect ratio is used to compute the recognition score.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetCompareAspectRatio()
void SetCompareAspectRatio(bool bCompareAspectRatio)
```

Remarks

When true, the character aspect ratio is used to compute the recognition score.

EOCR::GetCutLargeChars

EOCR::SetCutLargeChars

Flag indicating whether the large chars cutting mode is used.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetCutLargeChars()
void SetCutLargeChars(bool bCutLargeChars)
```

Remarks

If true, candidate characters larger than MaxWidth are split into as many parts as required. If false, large characters are discarded. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::DrawChar

Draws the bounding box of a given character.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawChar(
 EDrawAdapter* graphicContext,
 OEV_UINT32 charIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawChar(
 HDC graphicContext,
 OEV_UINT32 charIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawChar(
 HDC graphicContext,
 const ERGBColor& color,
 OEV_UINT32 charIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

charIndex

Character index, in range 0..NumChars-1.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR::DrawChars

Draws the bounding boxes of all characters in the image.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawChars(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawChars(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawChars(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all characters (to vary the colors, draw the objects separately using the EOCR::DrawChar method instead). Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR::DrawCharsWithCurrentPen

This method is deprecated.

Draws the bounding boxes of all characters in the image.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawCharsWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used for all characters (to vary the colors, draw the objects separately using the EOCR::DrawChar method instead). Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR::DrawCharWithCurrentPen

This method is deprecated.

Draws the bounding box of a given character.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawCharWithCurrentPen(
 HDC graphicContext,
 OEV_UINT32 charIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

charIndex

Character index, in range 0..NumChars-1.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Drawing is done in the device context associated to the desired window. The current pen is used. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR::DrawObjects

-

Namespace: Euresys::Open_eVision

[C++]

```
void DrawObjects(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::ESelectionFlag eSelection,
 float f32ZoomX,
 float f32ZoomY,
 float f32PanX,
 float f32PanY
 )
```

Parameters

graphicContext

-

eSelection

-

f32ZoomX

-

f32ZoomY

-

f32PanX

-

f32PanY

-

EOCR::EmptyChars

Empties the list of known characters.

Namespace: Euresys::Open_eVision

```
[C++]
void EmptyChars(
 )
```

Remarks

It is to use when you know the exact position of the characters to be recognized. See member EOCR::AddChar.

EOCR::EOCR

Constructs an OCR context.

Namespace: Euresys::Open_eVision

```
[C++]
void EOCR(
 )

void EOCR(
 const EOCR& other
 )
```

Parameters

other

Another EOCR object to be copied in the new EOCR object.

Remarks

By default, the Threshold property is set to 128.

EOCR::FindAllChars

Locates the characters by filtering the objects according to their size, and grouping them if the segmentation mode is set to RepasteObjects.

Namespace: Euresys::Open_eVision

[C++]

```
void FindAllChars(
 EROIBW8* sourceImage
 )
```

Parameters

sourceImage

Pointer to the source image/ROI. This parameter is not used and kept only for compatibility purposes.

Remarks

The characters are sorted from left to right. This operation must be performed after a call to EOCR::BuildObjects and before a call to EOCR::ReadText.

EOCR::GetConfidenceRatio

Returns the degree of confidence in the recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
float GetConfidenceRatio(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

A value of 100 % means there is no difference between the recognized character and the best matching pattern. A value of 0 % means there is no way to distinguish between the best and second best matching pattern. The computation of the confidence ratio is based on the first and second characters distance parameters (see EOCR::GetFirstCharDistance and EOCR::GetSecondCharDistance).

EOCR::GetFirstCharCode

Returns the code of the pattern that matches at best a recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFirstCharCode(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::GetFirstCharDistance

Computes the degree of similarity between the best matching pattern and a recognized character.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFirstCharDistance(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

Returns 0 for a perfect match and 1 for a total mismatch.

EOCR::GetPatternBitmap

Returns a pointer to an image holding the pattern of the given index. This image should not be modified.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW8* GetPatternBitmap(
 int index
 )
```

Parameters

index

Pattern number (in range 0.. GetNumPatterns() -1).

EOCR::GetPatternClass

Returns the class of a given pattern in the current font.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetPatternClass(
 int index
 )
```

Parameters

index

Pattern number (in range 0..NumPatterns-1).

EOCR::GetPatternCode

Returns the character code of a given pattern in the current font.

Namespace: Euresys::Open_eVision

```
[C++]
int GetPatternCode(
 int index
 )
```

Parameters

index

Pattern number (in range 0..NumPatterns-1).

EOCR::GetSecondCharCode

Returns the code of the pattern that matches at second best a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
int GetSecondCharCode(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

EOCR::GetSecondCharDistance

Computes the degree of similarity between the second best matching pattern and a recognized character.

Namespace: Euresys::Open_eVision

```
[C++]
float GetSecondCharDistance(
 int index
 )
```

Parameters

index

Character number (in range 0..NumChars-1).

Remarks

Returns 0 for a perfect match and 1 for a total mismatch.

EOCR::HitChar

Returns true if the cursor is placed over the character specified by charIndex.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitChar(
 int cursorX,
 int cursorY,
 OEV_UINT32 charIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

cursorX

Current cursor coordinates.

cursorY

Current cursor coordinates.

charIndex

Index of the character to be hit.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

If zooming and/or panning were used when drawing the ROI, the same values must be used with EOCR::HitChar.

EOCR::HitChars

Returns true if the cursor is placed over a character and sets the charIndex accordingly.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitChars(
 int cursorX,
 int cursorY,
 OEV_UINT32& charIndex,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

cursorX

Current cursor coordinates.

cursorY

Current cursor coordinates.

charIndex

Index of the character hit.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

If zooming and/or panning were used when drawing the ROI, the same values must be used with EOCR::HitChars.

EOCR::LearnPattern

Adds a new pattern to the font.

Namespace: Euresys::Open_eVision

[C++]

```
void LearnPattern(
 EROIBW8* sourceImage,
 OEV_UINT32 charIndex,
 int code,
 OEV_UINT32 classes,
 bool autoSegmentation
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

charIndex

Index of the character (ASCII or Unicode) to learn.

code

Character code of the pattern.

classes

Class of the pattern, as defined by EOCRClass.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default true) or bypassed (false).

Remarks

The pattern is selected by its index value, as assigned by the EOCR::FindAllChars process.

EOCR::LearnPatterns

Adds all the patterns from the source image to the font.

Namespace: Euresys::Open_eVision

[C++]

```
void LearnPatterns(
 EROIBW8* sourceImage,
 const std::string& text,
 OEV_UINT32 singleClass,
 bool autoSegmentation
 )

void LearnPatterns(
 EROIBW8* sourceImage,
 const std::string& text,
 const std::vector<OEV_UINT32>& classes,
 bool autoSegmentation
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

text

String containing character codes of the patterns.

singleClass

If specified, all the characters of the string are associated with the same class(es), that is specified by singleClass.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default true) or bypassed (false).

classes

If specified, the i-th character of the string is associated with the class specified by the i-th element of the vector classes.

Remarks

Patterns are ordered by their index value, as assigned by the EOCR::FindAllChars process.

EOCR::GetLineSpacingMode

EOCR::SetLineSpacingMode

Line spacing mode to sort the character boxes into lines. Default mode is ELineSpacingMode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ELineSpacingMode GetLineSpacingMode()
void SetLineSpacingMode(Euresys::Open_eVision::ELineSpacingMode eLineSpacingMode)
```

EOCR::Load

Loads the EOCR object configuration.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 ESerializer* serializer
 )
void Load(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

The path from which to load the configuration.

Remarks

The given ESerializer must have been created for reading.

EOCR::MatchChar

Reads a single character, i.e. finds the best match between the patterns in the font and the given character.

Namespace: Euresys::Open_eVision

```
[C++]
void MatchChar(
 EROIBW8* sourceImage,
 OEV_UINT32 classes,
 int index,
 int shiftX,
 int shiftY
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

classes

Logical mask obtained by combining the values of EOCRClass, to specify to what classes the character may belong.

index

Character number (in range 0..NumChars-1).

shiftX

Horizontal translation applied to the character.

shiftY

Vertical translation applied to the character.

Remarks

A shift can be apply to the character. This operation can only be performed after a call to EOCR::FindAllChars.

EOCR::GetMatchingMode

EOCR::SetMatchingMode

Matching mode to use to compare characters to the template.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EMatchingMode GetMatchingMode()
void SetMatchingMode(Euresys::Open_eVision::EMatchingMode eMatchingMode)
```

Remarks

By default, the root-mean-square error method is used. These modes are meant to be used without thresholding, that is when one of the EOCRColor_DarkOnLight and EOCRColor_LightOnDark colors are used.

EOCR::GetMaxCharHeight

EOCR::SetMaxCharHeight

Maximum character height.

Namespace: Euresys::Open_eVision

[C++]

```
int GetMaxCharHeight()
void SetMaxCharHeight(int n32MaxCharHeight)
```

Remarks

A character larger than the maximum width or higher than the maximum height is discarded. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetMaxCharWidth

EOCR::SetMaxCharWidth

Maximum character width.

Namespace: Euresys::Open_eVision

[C++]

```
int GetMaxCharWidth()
```

```
void SetMaxCharWidth(int n32MaxCharWidth)
```

Remarks

A character larger than the maximum width or higher than the maximum height is discarded. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetMinCharHeight

EOCR::SetMinCharHeight

Minimum character height.

Namespace: Euresys::Open_eVision

[C++]

```
int GetMinCharHeight()
void SetMinCharHeight(int n32MinCharHeight)
```

Remarks

A character both narrower than the minimum width and lower than the minimum height is discarded by default (see EOCR::RemoveNarrowOrFlat). The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetMinCharWidth

EOCR::SetMinCharWidth

Minimum character width.

Namespace: Euresys::Open_eVision

[C++]

```
int GetMinCharWidth()
void SetMinCharWidth(int n32MinCharWidth)
```

Remarks

A character both narrower than the minimum width and lower than the minimum height is discarded by default (see EOCR::RemoveNarrowOrFlat). The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::NewFont

Empties the contents of the font and sets the size of the pattern array to be used from then on.

Namespace: Euresys::Open_eVision

[C++]

```
void NewFont(
 OEV_UINT32 patternWidth,
 OEV_UINT32 patternHeight
 )
```

Parameters

patternWidth

Width of the normalized pattern representation, in pixels.

patternHeight

Height of the normalized pattern representation, in pixels.

Remarks

A larger pattern array improves the recognition sensitivity, at the expense of increased processing time.

EOCR::GetNoiseArea

EOCR::SetNoiseArea

Noise area.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNoiseArea()

void SetNoiseArea(int n32NoiseArea)
```

Remarks

When a blob has an area smaller than this value, it is ignored. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetNumChars

Number of recognized characters.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumChars()
```

EOCR::GetNumPatterns

Number of patterns in the current font.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumPatterns()
```

EOCR::operator=

Copies all the data from another EOCR object into the current EOCR object

Namespace: Euresys::Open_eVision

[C++]

```
EOCR& operator=(
 const EOCR& other
 )
```

Parameters

other

EOCR object to be copied

EOCR::GetPatternHeight

Current pattern height, as set at the last EOCR::NewFont or EOCR::Load operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetPatternHeight()
```

EOCR::GetPatternWidth

Current pattern width, as set at the last EOCR::NewFont or EOCR::Load operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetPatternWidth()
```

EOCR::ReadText

Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.

Namespace: Euresys::Open_eVision

[C++]

```
std::string ReadText(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 OEV_UINT32 classes,
 bool autoSegmentation
 )

std::string ReadText(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 const std::vector<OEV_UINT32>& classes,
 bool autoSegmentation
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical masks obtained by combining the values of EOCRClass, to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default true) or bypassed (false).

Remarks

This operation can only be performed after a call to EOCR::FindAllChars.

EOCR::ReadTextWide

This method is deprecated.

DEPRECATED: Use the EOCR::ReadText method instead as it performs similarly to this method, except it returns UTF-8 instead of wide characters. Reads one or more rows of characters, i.e. finds the best match between the patterns in the font and the segmented characters.

Namespace: Euresys::Open_eVision

[C++]

```
std::wstring ReadTextWide(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 OEV_UINT32 classes,
 bool autoSegmentation
 )

std::wstring ReadTextWide(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 const std::vector<OEV_UINT32>& classes,
 bool autoSegmentation
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical masks obtained by combining the values of EOCRClass, to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

autoSegmentation

Boolean indicating whether the calculation of the true threshold has to be forced (default true) or bypassed (false).

Remarks

This operation can only be performed after a call to EOCR::FindAllChars.

EOCR::Recognize

Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.

Namespace: Euresys::Open_eVision

[C++]

```
std::string Recognize(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 OEV_UINT32 classes
 )
```

```
std::string Recognize(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 const std::vector<OEV_UINT32>& classes
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical mask obtained by combining the values of EOCRClass, to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

Remarks

This method does the same as a sequence of EOCR::BuildObjects/EOCR::FindAllChars/EOCR::ReadText,

EOCR::RecognizeWide

This method is deprecated.

DEPRECATED: use the EOCR::Recognize method instead as it performs similarly to this method, except it returns UTF-8 instead of wide characters. Achieves all processing phases (blob analysis, character segmentation and pattern recognition) in a single operation.

Namespace: Euresys::Open_eVision

[C++]

```
std::wstring RecognizeWide(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 OEV_UINT32 classes
 )

std::wstring RecognizeWide(
 EROIBW8* sourceImage,
 int maximumNumberOfCharacters,
 const std::vector<OEV_UINT32>& classes
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

maximumNumberOfCharacters

Maximum number of characters to be read.

classes

Pointer to an array of logical mask obtained by combining the values of EOCRClass, to specify to what classes the character may belong. Each mask value in the array applies to the corresponding character.

Remarks

This method does the same as a sequence of EOCR::BuildObjects/EOCR::FindAllChars/EOCR::ReadText,

EOCR::GetRelativeSpacing

EOCR::SetRelativeSpacing

Relative spacing value.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeSpacing()
void SetRelativeSpacing(float f32RelativeSpacing)
```

Remarks

This value is the ratio of the width of the spaces between characters and the character width. For example, characters 25 pixels wide separated by 10 pixels gaps correspond to a relative spacing of 10/25 = 0.4. The default value of this parameter is 0, corresponding to no spacing. This parameter is relevant only when the CutLargeChars mode is enabled. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetRelativeThreshold

EOCR::SetRelativeThreshold

Relative threshold used for image segmentation.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRelativeThreshold()
void SetRelativeThreshold(float f32Threshold)
```

Remarks

The RelativeThreshold is the fraction of the image pixels that will be set below the threshold. Only used when the EOCR::Threshold property is set to EThresholdMode_Relative. The default value is 0.5.

EOCR::GetRemoveBorder

EOCR::SetRemoveBorder

Flag indicating whether all blobs touching the ROI edges are automatically discarded.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRemoveBorder()

void SetRemoveBorder(bool bRemoveBorder)
```

Remarks

If true, all blobs touching the ROI edges are automatically discarded. By default, this feature is turned on. The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetRemoveNarrowOrFlat

EOCR::SetRemoveNarrowOrFlat

Flag indicating whether the characters are discarded when either dimension falls below the minimum value

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRemoveNarrowOrFlat()

void SetRemoveNarrowOrFlat(bool bRemoveNarrowOrFlat)
```

Remarks

If true, characters are discarded if either their width or their height is smaller than the minimum value. By default, this feature is disabled (only smaller characters in *both* height and width are discarded: the condition could be written NarrowAndFlat). The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::RemovePattern

Removes a given pattern from the current font.

Namespace: Euresys::Open_eVision

[C++]

```
void RemovePattern(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the pattern to be removed from the current font.

EOCR::Save

Saves the EOCR object configuration.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 ESerializer* serializer
 )

void Save(
 const std::string& path
 )
```

Parameters

serializer

The serializer.

path

The path from which to save the configuration.

Remarks

The given ESerializer must have been created for writing.

EOCR::GetSegmentationMode

EOCR::SetSegmentationMode

Segmentation mode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ESegmentationMode GetSegmentationMode()

void SetSegmentationMode(Euresys::Open_eVision::ESegmentationMode eSegmentationMode)
```

Remarks

The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::SetPatternClass

Sets the class of a given pattern in the current font.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPatternClass(
 int index,
 OEV_UINT32 classIdx
 )
```

Parameters

index

Pattern number (in range 0..NumPatterns-1).

classIdx

The class.

EOCR::SetPatternCode

Sets the character code of a given pattern in the current font.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPatternCode(
 int index,
 int code
 )
```

Parameters

index

Pattern number (in range 0..NumPatterns-1).

code

The character code.

EOCR::GetShiftingMode

EOCR::SetShiftingMode

Shifting mode to use to compare characters to the template.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShiftingMode GetShiftingMode()
void SetShiftingMode(Euresys::Open_eVision::EShiftingMode eShiftingMode)
```

EOCR::GetShiftXTolerance

EOCR::SetShiftXTolerance

Horizontal translation tolerance around the nominal position when the character positions are explicitly specified.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetShiftXTolerance()
void SetShiftXTolerance(OEV_UINT32 un32ShiftXTolerance)
```

Remarks

By default, no shifting is allowed (ShiftX = 0). The tolerance can be used in two ways: either each character is moved individually until the best match is found, or the set of characters is matched as a whole, until the best average error is reached. See EOCR::ShiftingMode.

EOCR::GetShiftYTolerance

EOCR::SetShiftYTolerance

Vertical translation tolerance around the nominal position when the character positions are explicitly specified.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetShiftYTolerance()
void SetShiftYTolerance(OEV_UINT32 un32ShiftYTolerance)
```

Remarks

By default, no shifting is allowed (ShiftY = 0). The tolerance can be used in two ways: either each character is moved individually until the best match is found, or the set of characters is matched as a whole, until the best average error is reached. See EOCR::ShiftingMode.

EOCR::GetTextColor

EOCR::SetTextColor

Text color.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EOCRColor GetTextColor()
void SetTextColor(Euresys::Open_eVision::EOCRColor eTextColor)
```

Remarks

The segmentation parameters *must* be the same for both learning and recognition process.

EOCR::GetThreshold

EOCR::SetThreshold

Threshold mode used for image segmentation.

Namespace: Euresys::Open_eVision

[C++]

```
int GetThreshold()
void SetThreshold(int un32Threshold)
```

Remarks

Threshold value as defined by the EThresholdMode enum. By default, the "minimum residue" mode is used to determine the threshold value. In case of an absolute threshold, the threshold value is given instead.

EOCR::GetTrueThreshold

Absolute threshold level when using a single threshold.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTrueThreshold()
```

Remarks

This value is valid only when the EOCR::BuildObjects or EOCR::ReadText method has been called beforehand.

4.170. EOCR2 Class

Manages a complete context for the font-dependent printed character reader implemented in EasyOCR2.

Namespace: Euresys::Open_eVision

Methods

AddCharactersToDatabase	Reads reference characters from disk and adds them to the database used for text recognition. The characters can be read from a trueType (.ttf/.ttc) file or from an EasyOCR2 database file.
AddClassifierForSymbol	Adds the **EOCR2Classifier** for the given specific symbol combination during the recognition instead of the one set by EOCR2.
ClearCharacterDatabase	Clears the reference character database from this EOCR2 instance.
ClearResult	Clear the current detected text if it exists.
Detect	Finds the text in an image as follows: (1) Detects potential characters in the image following the given text polarity. (2) Fits bounding boxes to the detected characters, following the given topology and character width/height. (3) Extracts the detected characters from the image. The detected characters are output as an EOCR2Text structure.
DrawDetection	Draws the bounding boxes found by the topology detection algorithm.
DrawDetectionWithCurrentPen	Draws the bounding boxes found by the topology detection algorithm with the current pen.
DrawRecognition	Draws the recognized text next to the character bounding box in the image.
DrawRecognitionWithCurrentPen	Draws the recognized text next to the character bounding box in the image with the current pen.
DrawSegmentation	Draws the blobs found by the segmentation algorithm.
DrawSegmentationWithCurrentPen	Draws the blobs found by the segmentation algorithm with the current pen.
EOCR2	Constructs an EOCR2 context.
GetAllowedCharacterTypes	Sets which character types are expected when EOCR2::EnabledTopology is false. The set of expected character types is represented by a bitwise combination of different EasyOCR2CharacterFilter.
GetCharacterDatabase	Sets the EOCR2CharacterDatabase used for recognizing text.
GetCharsHeight	Sets the expected character height in pixels.

GetCharsMaxFragmentation	Sets the **CharsMaxFragmentation** parameter for the segmentation algorithm. This will determine the minimum size a blob should be in order to be considered a potential character. A high setting will allow only larger blobs, a low setting will also allow smaller blobs. The minimum blob size to be considered a potential character is defined as: `CharsMaxFragmentation * CharsHeight * CharsWidth`
GetCharsSpacingBias	Sets the **CharSpacingBias** parameter for the topology fitting algorithm, which optimizes the spacing of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow spacing, wide spacing or is neutral.
GetCharsWidthBias	Sets the **CharsWidthBias** parameter for the topology fitting algorithm, which optimizes the width of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow boxes, wide boxes or is neutral.
GetCharsWidthRange	Sets the range of expected character widths in pixels.
GetClassifier	Sets the **EOCR2Classifier** parameter for the recognition algorithm. This will determine which classifier will be used for the recognition.
GetClassifierForSymbol	Gets the **EOCR2Classifier** for the given specific symbol combination during the recognition instead of the one set by EOCR2.
GetDetectionDelta	Sets the **DetectionDelta** parameter for the segmentation algorithm. This will determine the range of grayscale-values used to determine the stability of a blob. A low setting will make the algorithm more sensitive to noise, a high setting will make the algorithm insensitive to blobs with low contrast to the background.
GetDetectionMethod	Sets the **EOCR2DetectionMethod** parameter for the topology fitting algorithm, which will place textboxes on the segmentation results, matching the given topology.
GetEnableCutLargeCharacter	Sets whether EOCR2 detection should split or not segmented blobs into multiple characters if they are wider than the given character width range parameter. The default setting for this parameter is false and it is only applicable when the topology is not required or when the detectionMethod is set to EOCR2DetectionMethod_Proportional.
GetEnabledTopology	Sets whether the topology is required or not. If it is, EOCR2 will try to detect the topology accordingly with the EOCR2DetectionMethod parameter. The default setting for this parameter is true and topology has to be given with EOCR2.
GetEnableGPU	Sets/Gets whether EOCR2 uses a GPU to accelerate its processing.
GetEnableOffSizeCharacter	Sets whether EOCR2 allows or not the detection of characters whose size (width and height) is out of the size parameters if they are in the vicinity of characters in valid size range. The default setting for this parameter is true and it is only applicable when the topology is not required.
GetEnableSecondPassGlobalSegmentation	Sets whether EOCR2 segmentation should do or not do an extra pass to determine the best threshold using the EOCR2SegmentationMethod_Global segmentation method. The default setting for this parameter is false and it is only applicable when the segmentation method is set to EOCR2SegmentationMethod_Global.

GetGlobalSegmentationRelativeThreshold	Sets/Gets the fraction of the image pixels that will be set below the threshold used when the segmentation method is set to EOCR2SegmentationMethod_Global. It is only used when the GlobalSegmentationThresholdMode value is EThresholdMode_Relative.
GetGlobalSegmentationThresholdMode	Sets/Gets the EThresholdMode used when the segmentation method is set to EOCR2SegmentationMethod_Global. From the EThresholdMode, a threshold will be computed during the segmentation. While using EasyOCR2TextPolarity_WhiteOnBlack, pixels above or equal to the threshold will be segmented. While using EasyOCR2TextPolarity_BlackOnWhite, pixels under the threshold will be segmented.
GetGPUIndexes	Sets/gets the GPUs to use when computing.
GetMaxVariation	Sets the **maxVariation** parameter for the segmentation algorithm. This parameter determines how stable a blob in the image should be in order to be considered a potential character, a region with clearly defined edges is generally considered stable while a blurry region is not. A high setting allows more unstable blobs, a low setting allows only very stable blobs.
GetNumDetectionPasses	Sets the **NumDetectionPasses** parameter for the topology fitting algorithm, which will place textboxes on the segmentation results (blobs), matching the given topology. The first pass will consider all blobs, subsequent passes will only consider those blobs that are inside the textboxes from the previous pass, sometimes resulting in a more optimal fit.
GetReadText	Outputs an EOCR2Text structure containing the detailed detection and recognition results.
GetRelativeSpacesWidthRange	Sets the range of expected spaces between words as a fraction of the character width.
GetRepasteObjects	Sets whether EOCR2 groups or does not group the blobs believed to belong to the same character. The default setting for this parameter is true and it is only applicable when the topology is not required.
GetSegmentationMethod	Sets the **EOCR2SegmentationMethod** parameter for the segmentation algorithm, which will detect blobs in the image.
GetTextAngleRange	Sets the **TextAngleRange** parameter for the topology fitting algorithm, which will attempt to find the angle of the text in the image with respect to the horizontal. This will determine the center of the range of angles that will be tested, defined as: `TextAngleRange.min() &lt;= angle &lt;= TextAngleRange.max()`
GetTextPolarity	Sets the **TextPolarity** parameter for the segmentation algorithm. This will determine whether the algorithm searches for light blobs in a dark background or for dark blobs in a light background.
GetTimeOut	Time-out for the EOCR2::Read, EOCR2::Detect and EOCR2::Recognize methods.

GetTopology	Sets the topology of the text that should be found in the image. A modified version of Regex expressions are used, where: **.**(dot) represents any character (not including a space). **L** represents a letter. **Lu** represents an uppercase letter. **Ll** represents a lowercase letter. **N** represents a digit. **P** represents a punctuation character !"#&'()*,-./:;<>?[\]_{|}~ **S** represents the symbols $+-<=>|~ **\n** represents a line break. ' ' (space) represents a space between two words. Combinations can be made, for example: `[LN]` represents an alpha-numeric character. To specify multiple characters, simply add `{n}` at the end for n characters. If the amount of characters is uncertain, specify `{n,m}` for a minimum of n characters and a maximum of m characters. The topology `"[LuN]{3,5}PN{4} \n .{5} LL"` represents a text comprised of 2 lines: The first line has 1 word composed of 3 to 5 uppercase alpha-numeric characters, followed by a punctuation character and 4 numbers. The second line has 2 words. The first word comprises of 5 wildcard characters, the second word has 2 alphabetic characters (upper- or lowercase).
HitTestChar	Tests the cursor position for the presence of a character. If one is present under the cursor, it returns true and fills the EOCR2Char object passed as parameter.
HitTestLine	Tests the cursor position for the presence of a line. If one is present under the cursor, it returns true and fills the EOCR2Line object passed as parameter.
HitTestText	Tests the cursor position for the presence of a text. If one is present under the cursor, it returns true and fills the EOCR2Text object passed as parameter.
HitTestWord	Tests the cursor position for the presence of a word. If one is present under the cursor, it returns true and fills the EOCR2Word object passed as parameter.
Learn	Learns reference characters from a given EOCR2Text/EOCR2Line/EOCR2Word/EOCR2Char instance, containing user-specified character codes.
Load	Loads a model, containing parameter settings used for all operations in EOCR2, from disk.
operator=	Assignment operator, copies another EOCR2 instance to this one.
Read	Performs all steps required for reading text from an image: (1) Detects potential characters in the image following the given text polarity and character width/height. (2) Fits bounding boxes to the detected characters, following the given topology and character width/height. (3) Recognizes the detected characters using the given reference character database. The read text is output as a string.

Recognize	Recognizes the characters in a given EOCR2Text instance, based on a given reference font.
RemoveClassifierForSymbol	Removes the **EOCR2Classifier** for the given specific symbol combination during the recognition so the one set by EOCR2 will be used.
Save	Saves the model to disk, containing the current parameter setting used for all operations in EOCR2.
SaveCharacterDatabase	Saves the current reference character database to disk.
SetAllowedCharacterTypes	Sets which character types are expected when EOCR2::EnabledTopology is false. The set of expected character types is represented by a bitwise combination of different EasyOCR2CharacterFilter.
SetCharacterDatabase	Sets the EOCR2CharacterDatabase used for recognizing text.
SetCharsHeight	Sets the expected character height in pixels.
SetCharsMaxFragmentation	Sets the **CharsMaxFragmentation** parameter for the segmentation algorithm. This will determine the minimum size a blob should be in order to be considered a potential character. A high setting will allow only larger blobs, a low setting will also allow smaller blobs. The minimum blob size to be considered a potential character is defined as: `CharsMaxFragmentation * CharsHeight * CharsWidth`
SetCharsSpacingBias	Sets the **CharSpacingBias** parameter for the topology fitting algorithm, which optimizes the spacing of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow spacing, wide spacing or is neutral.
SetCharsWidthBias	Sets the **CharsWidthBias** parameter for the topology fitting algorithm, which optimizes the width of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow boxes, wide boxes or is neutral.
SetCharsWidthRange	Sets the range of expected character widths in pixels.
SetClassifier	Sets the **EOCR2Classifier** parameter for the recognition algorithm. This will determine which classifier will be used for the recognition.
SetDetectionDelta	Sets the **DetectionDelta** parameter for the segmentation algorithm. This will determine the range of grayscale-values used to determine the stability of a blob. A low setting will make the algorithm more sensitive to noise, a high setting will make the algorithm insensitive to blobs with low contrast to the background.
SetDetectionMethod	Sets the **EOCR2DetectionMethod** parameter for the topology fitting algorithm, which will place textboxes on the segmentation results, matching the given topology.
SetEnableCutLargeCharacter	Sets whether EOCR2 detection should split or not segmented blobs into multiple characters if they are wider than the given character width range parameter. The default setting for this parameter is false and it is only applicable when the topology is not required or when the detectionMethod is set to EOCR2DetectionMethod_Proportional.

SetEnabledTopology	Sets whether the topology is required or not. If it is, EOCR2 will try to detect the topology accordingly with the EOCR2DetectionMethod parameter. The default setting for this parameter is true and topology has to be given with EOCR2.
SetEnableGPU	Sets/Gets whether EOCR2 uses a GPU to accelerate its processing.
SetEnableOffSizeCharacter	Sets whether EOCR2 allows or not the detection of characters whose size (width and height) is out of the size parameters if they are in the vicinity of characters in valid size range. The default setting for this parameter is true and it is only applicable when the topology is not required.
SetEnableSecondPassGlobalSegmentation	Sets whether EOCR2 segmentation should do or not do an extra pass to determine the best threshold using the EOCR2SegmentationMethod_Global segmentation method. The default setting for this parameter is false and it is only applicable when the segmentation method is set to EOCR2SegmentationMethod_Global.
SetGlobalSegmentationRelativeThreshold	Sets/Gets the fraction of the image pixels that will be set below the threshold used when the segmentation method is set to EOCR2SegmentationMethod_Global. It is only used when the GlobalSegmentationThresholdMode value is EThresholdMode_Relative .
SetGlobalSegmentationThresholdMode	Sets/Gets the EThresholdMode used when the segmentation method is set to EOCR2SegmentationMethod_Global. From the EThresholdMode, a threshold will be computed during the segmentation. While using EasyOCR2TextPolarity_WhiteOnBlack, pixels above or equal to the threshold will be segmented. While using EasyOCR2TextPolarity_BlackOnWhite, pixels under the threshold will be segmented.
SetGPUIndexes	Sets/gets the GPUs to use when computing.
SetMaxVariation	Sets the **maxVariation** parameter for the segmentation algorithm. This parameter determines how stable a blob in the image should be in order to be considered a potential character, a region with clearly defined edges is generally considered stable while a blurry region is not. A high setting allows more unstable blobs, a low setting allows only very stable blobs.
SetNumDetectionPasses	Sets the **NumDetectionPasses** parameter for the topology fitting algorithm, which will place textboxes on the segmentation results (blobs), matching the given topology. The first pass will consider all blobs, subsequent passes will only consider those blobs that are inside the textboxes from the previous pass, sometimes resulting in a more optimal fit.
SetRelativeSpacesWidthRange	Sets the range of expected spaces between words as a fraction of the character width.
SetRepasteObjects	Sets whether EOCR2 groups or does not group the blobs believed to belong to the same character. The default setting for this parameter is true and it is only applicable when the topology is not required.
SetSegmentationMethod	Sets the **EOCR2SegmentationMethod** parameter for the segmentation algorithm, which will detect blobs in the image.

SetTextAngleRange	Sets the **TextAngleRange** parameter for the topology fitting algorithm, which will attempt to find the angle of the text in the image with respect to the horizontal. This will determine the center of the range of angles that will be tested, defined as: `TextAngleRange.min() <= angle <= TextAngleRange.max()`
SetTextPolarity	Sets the **TextPolarity** parameter for the segmentation algorithm. This will determine whether the algorithm searches for light blobs in a dark background or for dark blobs in a light background.
SetTimeOut	Time-out for the EOCR2::Read, EOCR2::Detect and EOCR2::Recognize methods.
SetTopology	Sets the topology of the text that should be found in the image. A modified version of Regex expressions are used, where: **.**(dot) represents any character (not including a space). **L** represents a letter. **Lu** represents an uppercase letter. **Ll** represents a lowercase letter. **N** represents a digit. **P** represents a punctuation character !"#&'()*,-./:;<>?[\]_{|}~ **S** represents the symbols $+-<=>|~ **\n** represents a line break. ' ' (space) represents a space between two words. Combinations can be made, for example: `[LN]` represents an alpha-numeric character. To specify multiple characters, simply add `{n}` at the end for n characters. If the amount of characters is uncertain, specify `{n,m}` for a minimum of n characters and a maximum of m characters. The topology "`[LuN]{3,5}PN{4} \n .{5} LL`" represents a text comprised of 2 lines: The first line has 1 word composed of 3 to 5 uppercase alpha-numeric characters, followed by a punctuation character and 4 numbers. The second line has 2 words. The first word comprises of 5 wildcard characters, the second word has 2 alphabetic characters (upper- or lowercase).

EOCR2::AddCharactersToDatabase

Reads reference characters from disk and adds them to the database used for text recognition. The characters can be read from a trueType (.ttf/.ttc) file or from an EasyOCR2 database file.

Namespace: Euresys::Open_eVision

[C++]

```
void AddCharactersToDatabase(
 const std::string& file
 )

void AddCharactersToDatabase(
 const std::string& file,
 Euresys::Open_eVision::EasyOCR2CharacterFilter filter
 )
```

Parameters

file

A string containing the path of the file.

filter

Optional parameter; an EasyOCR2CharacterFilter that tells the method which subset of the character-set should be loaded.

EOCR2::AddClassifierForSymbol

Adds the **EOCR2Classifier** for the given specific symbol combination during the recognition instead of the one set by EOCR2.

Namespace: Euresys::Open_eVision

[C++]

```
void AddClassifierForSymbol(
 const std::string& symbol,
 Euresys::Open_eVision::EOCR2Classifier classifier
 )
```

Parameters

symbol

-

classifier

-

Remarks

The only accepted EOCR2Classifier is currently EOCR2Classifier_DatabaseClassifier. Symbol combinations are the same as used in the topology, ex : Lu, P or [LlNS]. See EOCR2 for more details.

EOCR2::GetAllowedCharacterTypes

EOCR2::SetAllowedCharacterTypes

Sets which character types are expected when EOCR2::EnabledTopology is false. The set of expected character types is represented by a bitwise combination of different EasyOCR2CharacterFilter.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EasyOCR2CharacterFilter GetAllowedCharacterTypes() const

void SetAllowedCharacterTypes(Euresys::Open_eVision::EasyOCR2CharacterFilter
allowedCharacterTypes)
```

Remarks

For example, digits and uppercase letters can be expressed by EasyOCR2CharacterFilter_UpperCaseLetters | EasyOCR2CharacterFilter_Digits. The default setting for this parameter is EasyOCR2CharacterFilter_ASCII.

EOCR2::GetCharacterDatabase

EOCR2::SetCharacterDatabase

Sets the EOCR2CharacterDatabase used for recognizing text.

Namespace: Euresys::Open_eVision

[C++]

```
EOCR2CharacterDatabase GetCharacterDatabase()
void SetCharacterDatabase(const EOCR2CharacterDatabase& database)
```

Remarks

Setting a new characterDatabase will overwrite the current one.

EOCR2::GetCharsHeight

EOCR2::SetCharsHeight

Sets the expected character height in pixels.

Namespace: Euresys::Open_eVision

[C++]

```
int GetCharsHeight() const
void SetCharsHeight(int height)
```

EOCR2::GetCharsMaxFragmentation

EOCR2::SetCharsMaxFragmentation

Sets the **CharsMaxFragmentation** parameter for the segmentation algorithm. This will determine the minimum size a blob should be in order to be considered a potential character. A high setting will allow only larger blobs, a low setting will also allow smaller blobs.
The minimum blob size to be considered a potential character is defined as:
`CharsMaxFragmentation * CharsHeight * CharsWidth`

Namespace: Euresys::Open_eVision

[C++]

```
float GetCharsMaxFragmentation() const
void SetCharsMaxFragmentation(float charMaxFragmentation)
```

Remarks

This parameter should be set between 0.0 and 1.0, the default setting is 0.1.

EOCR2::GetCharsSpacingBias

EOCR2::SetCharsSpacingBias

Sets the **CharSpacingBias** parameter for the topology fitting algorithm, which optimizes the spacing of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow spacing, wide spacing or is neutral.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EasyOCR2CharSpacingBias GetCharsSpacingBias() const
void SetCharsSpacingBias(Euresys::Open_eVision::EasyOCR2CharSpacingBias
charSpacingBias)
```

Remarks

The default setting for this parameter is EasyOCR2CharSpacingBias_Neutral

EOCR2::GetCharsWidthBias

EOCR2::SetCharsWidthBias

Sets the **CharsWidthBias** parameter for the topology fitting algorithm, which optimizes the width of the bounding boxes to optimally fit the detected blobs. This will determine whether the method is biased toward finding narrow boxes, wide boxes or is neutral.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EasyOCR2CharWidthBias GetCharsWidthBias() const
void SetCharsWidthBias(Euresys::Open_eVision::EasyOCR2CharWidthBias charWidthBias)
```

Remarks

The default setting for this parameter is EasyOCR2CharWidthBias_Neutral

EOCR2::GetCharsWidthRange

EOCR2::SetCharsWidthRange

Sets the range of expected character widths in pixels.

Namespace: Euresys::Open_eVision

[C++]

```
EIntegerRange& GetCharsWidthRange()
void SetCharsWidthRange(const EIntegerRange& range)
```

Remarks

The CharsWidthRange is returned by reference, changing it will affect the internal state of the EOCR2 object.

EOCR2::GetClassifier

EOCR2::SetClassifier

Sets the **EOCR2Classifier** parameter for the recognition algorithm. This will determine which classifier will be used for the recognition.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EOCR2Classifier GetClassifier() const
void SetClassifier(Euresys::Open_eVision::EOCR2Classifier classifier)
```

Remarks

The default setting for this parameter is EOCR2Classifier_DatabaseClassifier.

EOCR2::ClearCharacterDatabase

Clears the reference character database from this EOCR2 instance.

Namespace: Euresys::Open_eVision

[C++]

```
void ClearCharacterDatabase(
 )
```

EOCR2::ClearResult

Clear the current detected text if it exists.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearResult(
 )
```

EOCR2::Detect

Finds the text in an image as follows:
(1) Detects potential characters in the image following the given text polarity.
(2) Fits bounding boxes to the detected characters, following the given topology and character width/height.
(3) Extracts the detected characters from the image.
The detected characters are output as an EOCR2Text structure.

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2Text Detect(
 const EROIBW8& srcRoi
 )

EOCR2Text Detect(
 const EROIBW8& srcRoi,
 const ERegion& region
 )
```

Parameters

srcRoi

The source image/ROI.

region

The region of interest where the detection is performed

Remarks

The variables Topology, CharHeight, CharWidth should be set before performing this operation. If the srcRoi is smaller than 3X3, an exception will be thrown.

EOCR2::GetDetectionDelta

EOCR2::SetDetectionDelta

Sets the **DetectionDelta** parameter for the segmentation algorithm. This will determine the range of grayscale-values used to determine the stability of a blob. A low setting will make the algorithm more sensitive to noise, a high setting will make the algorithm insensitive to blobs with low contrast to the background.

Namespace: Euresys::Open_eVision

```
[C++]
int GetDetectionDelta() const
void SetDetectionDelta(int delta)
```

Remarks

This parameter should be set between 0 and 127, the default setting is 12.

EOCR2::GetDetectionMethod

EOCR2::SetDetectionMethod

Sets the **EOCR2DetectionMethod** parameter for the topology fitting algorithm, which will place textboxes on the segmentation results, matching the given topology.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EOCR2DetectionMethod GetDetectionMethod() const
void SetDetectionMethod(Euresys::Open_eVision::EOCR2DetectionMethod method)
```

Remarks

The default setting for this parameter is EOCR2DetectionMethod. This parameter is ignored when the topology is not required.

EOCR2::DrawDetection

Draws the bounding boxes found by the topology detection algorithm.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawDetection(
 EDrawAdapter* drawAdapter,
 Euresys::Open_eVision::EasyOCR2DrawDetectionStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawDetection(
 HDC hdc,
 Euresys::Open_eVision::EasyOCR2DrawDetectionStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawDetection(
 HDC hdc,
 ERGBColor color,
 Euresys::Open_eVision::EasyOCR2DrawDetectionStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

style

The style in which each detection box should be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

hdc

Handle of the device context on which to draw.

color

The color in which to draw the detection.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR2::DrawDetectionWithCurrentPen

This method is deprecated.

Draws the bounding boxes found by the topology detection algorithm with the current pen.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawDetectionWithCurrentPen(
 HDC hdc,
 Euresys::Open_eVision::EasyOCR2DrawDetectionStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hdc

Handle of the device context on which to draw.

style

The style in which each detection box should be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR2::DrawRecognition

Draws the recognized text next to the character bounding box in the image.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawRecognition(
 EDrawAdapter* drawAdapter,
 Euresys::Open_eVision::EasyOCR2DrawRecognitionStyle style,
 OEV_UINT32 cHeight,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawRecognition(
 HDC hdc,
 Euresys::Open_eVision::EasyOCR2DrawRecognitionStyle style,
 OEV_UINT32 cHeight,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawRecognition(
 HDC hdc,
 ERGBColor textColor,
 ERGBColor backgroundColor,
 Euresys::Open_eVision::EasyOCR2DrawRecognitionStyle style,
 OEV_UINT32 cHeight,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

style

The style in which each recognition result should be drawn.

cHeight

The character-height with which the recognized text should be displayed.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

hdc

Handle of the device context on which to draw.

textColor

The color in which to draw the recognized text.

backgroundColor

The color of the box in which the text is displayed.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR2::DrawRecognitionWithCurrentPen

This method is deprecated.

Draws the recognized text next to the character bounding box in the image with the current pen.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawRecognitionWithCurrentPen(
 HDC hdc,
 Euresys::Open_eVision::EasyOCR2DrawRecognitionStyle style,
 OEV_UINT32 cHeight,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hdc

Handle of the device context on which to draw.

style

The style in which each recognition result should be drawn.

cHeight

The character-height with which the recognized text should be displayed.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR2::DrawSegmentation

Draws the blobs found by the segmentation algorithm.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawSegmentation(
 EDrawAdapter* drawAdapter,
 Euresys::Open_eVision::EasyOCR2DrawSegmentationStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawSegmentation(
 HDC hdc,
 Euresys::Open_eVision::EasyOCR2DrawSegmentationStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawSegmentation(
 HDC hdc,
 const ERGBColor& color,
 Euresys::Open_eVision::EasyOCR2DrawSegmentationStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

style

The style in which each blob should be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

hdc

Handle of the device context on which to draw.

color

The color in which to draw the segmentation.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR2::DrawSegmentationWithCurrentPen

This method is deprecated.

Draws the blobs found by the segmentation algorithm with the current pen.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawSegmentationWithCurrentPen(
 HDC hdc,
 Euresys::Open_eVision::EasyOCR2DrawSegmentationStyle style,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hdc

Handle of the device context on which to draw.

style

The style in which each blob should be drawn.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EOCR2::GetEnableCutLargeCharacter

EOCR2::SetEnableCutLargeCharacter

Sets whether EOCR2 detection should split or not segmented blobs into multiple characters if they are wider than the given character width range parameter. The default setting for this parameter is false and it is only applicable when the topology is not required or when the detectionMethod is set to Proportional.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableCutLargeCharacter() const
void SetEnableCutLargeCharacter(bool enableCutLargeCharacter)
```

EOCR2::GetEnabledTopology

EOCR2::SetEnabledTopology

Sets whether the topology is required or not. If it is, EOCR2 will try to detect the topology accordingly with the EOCR2DetectionMethod parameter. The default setting for this parameter is true and topology has to be given with EOCR2.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnabledTopology() const
void SetEnabledTopology(bool enableTopology)
```

EOCR2::GetEnableGPU

EOCR2::SetEnableGPU

Sets/Gets whether EOCR2 uses a GPU to accelerate its processing.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableGPU() const
void SetEnableGPU(bool enable)
```

Remarks

Currently only the recognition with a deep-learning classifier can be accelerated with GPU.

EOCR2::GetEnableOffSizeCharacter

EOCR2::SetEnableOffSizeCharacter

Sets whether EOCR2 allows or not the detection of characters whose size (width and height) is out of the size parameters if they are in the vicinity of characters in valid size range. The default setting for this parameter is true and it is only applicable when the topology is not required.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableOffSizeCharacter() const

void SetEnableOffSizeCharacter(bool enableOffSizeCharacter)
```

EOCR2::GetEnableSecondPassGlobalSegmentation

EOCR2::SetEnableSecondPassGlobalSegmentation

Sets whether EOCR2 segmentation should do or not do an extra pass to determine the best threshold using the Global segmentation method. The default setting for this parameter is false and it is only applicable when the segmentation method is set to Global.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableSecondPassGlobalSegmentation() const

void SetEnableSecondPassGlobalSegmentation(bool secondPassGlobalSegmentation)
```

Remarks

The extra pass adds computation.

EOCR2::EOCR2

Constructs an EOCR2 context.

Namespace: Euresys::Open_eVision

[C++]

```
void EOCR2(
 )

void EOCR2(
 const EOCR2& other
 )
```

Parameters

other

-

EOCR2::GetClassifierForSymbol

Gets the **EOCR2Classifier** for the given specific symbol combination during the recognition instead of the one set by EOCR2.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EOCR2Classifier GetClassifierForSymbol(
 const std::string& symbol
 )
```

Parameters

symbol

-

Remarks

Symbol combinations are the same as used in the topology, ex : Lu, P or [LlNS]. See EOCR2 for more details.

EOCR2::GetGlobalSegmentationRelativeThreshold

EOCR2::SetGlobalSegmentationRelativeThreshold

Sets/Gets the fraction of the image pixels that will be set below the threshold used when the segmentation method is set to Global. It is only used when the GlobalSegmentationThresholdMode value is Relative .

Namespace: Euresys::Open_eVision

```
[C++]
float GetGlobalSegmentationRelativeThreshold() const

void SetGlobalSegmentationRelativeThreshold(float relativeThreshold)
```

EOCR2::GetGlobalSegmentationThresholdMode

EOCR2::SetGlobalSegmentationThresholdMode

Sets/Gets the EThresholdMode used when the segmentation method is set to Global. From the EThresholdMode, a threshold will be computed during the segmentation. While using WhiteOnBlack, pixels above or equal to the threshold will be segmented. While using BlackOnWhite, pixels under the threshold will be segmented.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EThresholdMode GetGlobalSegmentationThresholdMode() const

void SetGlobalSegmentationThresholdMode(Euresys::Open_eVision::EThresholdMode
thresholdMode)
```

Remarks

The default setting for this parameter is EThresholdMode.

EOCR2::GetGPUIndexes

EOCR2::SetGPUIndexes

Sets/gets the GPUs to use when computing.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<OEV_UINT32> GetGPUIndexes() const

void SetGPUIndexes(const std::vector<OEV_UINT32>& index)
```

EOCR2::HitTestChar

Tests the cursor position for the presence of a character. If one is present under the cursor, it returns true and fills the EOCR2Char object passed as parameter.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTestChar(
 EOCR2Char& character,
 int& lineN,
 int& wordN,
 int& charN,
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

character

Returns the character if one was detected under the cursor.

lineN

Returns the line-number of the character if one was detected under the cursor.

wordN

Returns the word-number of the character if one was detected under the cursor.

charN

Returns the character-number of the character if one was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

EOCR2::HitTestLine

Tests the cursor position for the presence of a line. If one is present under the cursor, it returns true and fills the EOCR2Line object passed as parameter.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTestLine(
 EOCR2Line& line,
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

line

Object to fill if a line was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

EOCR2::HitTestText

Tests the cursor position for the presence of a text. If one is present under the cursor, it returns true and fills the EOCR2Text object passed as parameter.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTestText(
 EOCR2Text& text,
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

text

Object to fill if a text was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

EOCR2::HitTestWord

Tests the cursor position for the presence of a word. If one is present under the cursor, it returns true and fills the EOCR2Word object passed as parameter.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTestWord(
 EOCR2Word& word,
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

word

Object to fill if a word was detected under the cursor.

x

Horizontal position of the cursor.

y

Vertical position of the cursor.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

EOCR2::Learn

Learns reference characters from a given EOCR2Text/EOCR2Line/EOCR2Word/EOCR2Char instance, containing user-specified character codes.

Namespace: Euresys::Open_eVision

[C++]

```
void Learn(
 const EOCR2Char& character
 )

void Learn(
 const EOCR2Word& word
 )

void Learn(
 const EOCR2Line& line
 )

void Learn(
 const EOCR2Text& text
 )
```

Parameters

character

A single EOCR2Char character, containing the detected character from the reference image as well as the corresponding character code.

word

A single EOCR2Word word, containing the detected characters in a single word from the reference image as well as the corresponding character codes.

line

A single EOCR2Line line, containing the detected characters in a single line from the reference image as well as the corresponding character codes.

text

A complete EOCR2Text text, containing all detected characters from the reference image as well as the corresponding character codes.

Remarks

The EOCR2Text/EOCR2Line/EOCR2Word/EOCR2Char instance should contain detected characters from a reference image as well as their corresponding character codes.

EOCR2::Load

Loads a model, containing parameter settings used for all operations in EOCR2, from disk.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& modelPath
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

modelPath

A string containing the full path to the model file.

serializer

The serializer.

EOCR2::GetMaxVariation

EOCR2::SetMaxVariation

Sets the **maxVariation** parameter for the segmentation algorithm. This parameter determines how stable a blob in the image should be in order to be considered a potential character, a region with clearly defined edges is generally considered stable while a blurry region is not. A high setting allows more unstable blobs, a low setting allows only very stable blobs.

Namespace: Euresys::Open_eVision

```
[C++]
float GetMaxVariation() const
void SetMaxVariation(float maxVariation)
```

Remarks

This parameter should be set between 0.0 and 1.0, the default setting is 0.25.

EOCR2::GetNumDetectionPasses

EOCR2::SetNumDetectionPasses

Sets the **NumDetectionPasses** parameter for the topology fitting algorithm, which will place textboxes on the segmentation results (blobs), matching the given topology. The first pass will consider all blobs, subsequent passes will only consider those blobs that are inside the textboxes from the previous pass, sometimes resulting in a more optimal fit.

Namespace: Euresys::Open_eVision

```
[C++]
int GetNumDetectionPasses() const
void SetNumDetectionPasses(int nDetectionPasses)
```

Remarks

The default setting for this parameter is 1, the setting can be either 1 or 2.

EOCR2::operator=

Assignment operator, copies another EOCR2 instance to this one.

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2& operator=(
 const EOCR2& other
 )
```

Parameters

other

The EOCR2 instance to copy from.

EOCR2::Read

Performs all steps required for reading text from an image:
(1) Detects potential characters in the image following the given text polarity and character width/height.
(2) Fits bounding boxes to the detected characters, following the given topology and character width/height.
(3) Recognizes the detected characters using the given reference character database.
The read text is output as a string.

Namespace: Euresys::Open_eVision

[C++]

```
std::string Read(
 const EROIBW8& srcRoi
 )

std::string Read(
 const EROIBW8& srcRoi,
 const ERegion& region
 )
```

Parameters

srcRoi

The source image/ROI.

region

The region of interest where the reading is performed

Remarks

The variables TextPolarity, Topology, CharHeight, CharWidth should be set and a reference character database should be set before performing this operation. If the srcRoi is smaller than 3X3, an exception will be thrown.

EOCR2::GetReadText

Outputs an EOCR2Text structure containing the detailed detection and recognition results.

Namespace: Euresys::Open_eVision

[C++]

```
EOCR2Text GetReadText() const
```

EOCR2::Recognize

Recognizes the characters in a given EOCR2Text instance, based on a given reference font.

Namespace: Euresys::Open_eVision

[C++]

```
std::string Recognize(
 const EOCR2Text& text
 )

std::string Recognize(
 const EOCR2Text& text,
 const EROIBW8& srcRoi
 )
```

Parameters

text

EOCR2Text structure containing the detected characters from the image.

srcRoi

The source image/ROI.

Remarks

A reference character database should be provided before performing this operation. The overloaded function that uses an ROI is not yet implemented.

EOCR2::GetRelativeSpacesWidthRange

EOCR2::SetRelativeSpacesWidthRange

Sets the range of expected spaces between words as a fraction of the character width.

Namespace: Euresys::Open_eVision

[C++]

```
EFloatRange& GetRelativeSpacesWidthRange()

void SetRelativeSpacesWidthRange(const EFloatRange& range)
```

Remarks

This parameter only affects the detection when the detectionMethod is set to EOCR2DetectionMethod_FixedWidth or when the topology is not required. The RelativeSpacesWidthRange is returned by reference, changing it will affect the internal state of the EOCR2 object.

EOCR2::RemoveClassifierForSymbol

Removes the **EOCR2Classifier** for the given specific symbol combination during the recognition so the one set by EOCR2 will be used.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveClassifierForSymbol(
 const std::string& symbol
 )
```

Parameters

symbol

-

Remarks

Symbol combinations are the same as used in the topology, ex : Lu, P or [LINS]. See EOCR2 for more details.

EOCR2::GetRepasteObjects

EOCR2::SetRepasteObjects

Sets whether EOCR2 groups or does not group the blobs believed to belong to the same character. The default setting for this parameter is true and it is only applicable when the topology is not required.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRepasteObjects() const

void SetRepasteObjects(bool repasteObjects)
```

EOCR2::Save

Saves the model to disk, containing the current parameter setting used for all operations in EOCR2.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& modelPath
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

modelPath

A string containing the full path to the model file.

serializer

The serializer.

Remarks

It is advised to use a file extension that is non-standard (for instance *.ocr2)

EOCR2::SaveCharacterDatabase

Saves the current reference character database to disk.

Namespace: Euresys::Open_eVision

```
[C++]
void SaveCharacterDatabase(
 const std::string& file
 )
```

Parameters

file

A string containing the path of the file.

EOCR2::GetSegmentationMethod

EOCR2::SetSegmentationMethod

Sets the **EOCR2SegmentationMethod** parameter for the segmentation algorithm, which will detect blobs in the image.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EOCR2SegmentationMethod GetSegmentationMethod() const
void SetSegmentationMethod(Euresys::Open_eVision::EOCR2SegmentationMethod method)
```

Remarks

The default setting for this parameter is EOCR2SegmentationMethod_Local.

EOCR2::GetTextAngleRange

EOCR2::SetTextAngleRange

Sets the **TextAngleRange** parameter for the topology fitting algorithm, which will attempt to find the angle of the text in the image with respect to the horizontal. This will determine the center of the range of angles that will be tested, defined as:
`TextAngleRange.min() <= angle <= TextAngleRange.max()`

Namespace: Euresys::Open_eVision

[C++]

```
EFloatRange& GetTextAngleRange()
void SetTextAngleRange(const EFloatRange& range)
```

Remarks

The textAngleRange is returned by reference, changing it will affect the internal state of the EOCR2 object. The default setting for this parameter is -20 degrees to +20 degrees.

EOCR2::GetTextPolarity

EOCR2::SetTextPolarity

Sets the **TextPolarity** parameter for the segmentation algorithm. This will determine whether the algorithm searches for light blobs in a dark background or for dark blobs in a light background.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EasyOCR2TextPolarity GetTextPolarity() const
void SetTextPolarity(Euresys::Open_eVision::EasyOCR2TextPolarity polarity)
```

Remarks

Default setting is EasyOCR2TextPolarity_WhiteOnBlack.

EOCR2::GetTimeOut

EOCR2::SetTimeOut

Time-out for the EOCR2::Read, EOCR2::Detect and EOCR2::Recognize methods.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT64 GetTimeOut() const
void SetTimeOut(OEV_UINT64 value)
```

Remarks

If the processing time of one of these functions becomes longer than the set time-out, the processing is stopped and an exception is thrown. In that case, the error code of the exception is EError_TimeoutReached. The time-out is set in microseconds. This time-out is not a real time-out. The processing is stopped as soon as possible after the time-out has been reached. This means that the time elapsed effectively in the method can be greater than the time-out in itself.

EOCR2::GetTopology

EOCR2::SetTopology

Sets the topology of the text that should be found in the image. A modified version of Regex expressions are used, where:
.(dot) represents any character (not including a space).
L represents a letter.
Lu represents an uppercase letter.
Ll represents a lowercase letter.
N represents a digit.
P represents a punctuation character !"#&'()*,-./:;<>?[\]_{|}~
S represents the symbols $+-<=>|~
\n represents a line break.
' ' (space) represents a space between two words.
Combinations can be made, for example: `[LN]` represents an alpha-numeric character.
To specify multiple characters, simply add `{n}` at the end for n characters. If the amount of characters is uncertain, specify `{n,m}` for a minimum of n characters and a maximum of m characters.
The topology "`[LuN]{3,5}PN{4} \n .{5} LL`" represents a text comprised of 2 lines:
The first line has 1 word composed of 3 to 5 uppercase alpha-numeric characters, followed by a punctuation character and 4 numbers.
The second line has 2 words. The first word comprises of 5 wildcard characters, the second word has 2 alphabetic characters (upper- or lowercase).

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetTopology() const
void SetTopology(const std::string& topology)
```

4.171. EOCR2Char Class

Holds all information related to a single detected character.

Namespace: Euresys::Open_eVision

Methods

EOCR2Char	Constructs an EOCR2Char context.
GetBitmap	The bitmap associated to this character.
GetBoundingBox	The bounding box associated to this character.
GetCandidates	The list of recognition results for all reference-characters with their respective scores. The list is sorted from the best score to the worst one.
GetText	The true value of the character, this is used during the learning phase.
operator=	Copies all the data from another EOCR2Char object into the current EOCR2Char object
SetText	The true value of the character, this is used during the learning phase.
SetTextCode	The true value of the character, this is used during the learning phase.

EOCR2Char::GetBitmap

The bitmap associated to this character.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW8& GetBitmap()
```

EOCR2Char::GetBoundingBox

The bounding box associated to this character.

Namespace: Euresys::Open_eVision

[C++]

```
ERectangle GetBoundingBox()
```

EOCR2Char::GetCandidates

The list of recognition results for all reference-characters with their respective scores. The list is sorted from the best score to the worst one.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2CharacterCandidate> GetCandidates()
```

EOCR2Char::EOCR2Char

Constructs an EOCR2Char context.

Namespace: Euresys::Open_eVision

```
[C++]
void EOCR2Char(
 )
void EOCR2Char(
 const EOCR2Char& other
 )
```

Parameters

other

EOCR2Char object to be copied.

EOCR2Char::operator=

Copies all the data from another EOCR2Char object into the current EOCR2Char object

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2Char& operator=(
 const EOCR2Char& other
 )
```

Parameters

other

EOCR2Char object to be copied

EOCR2Char::GetText

EOCR2Char::SetText

The true value of the character, this is used during the learning phase.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetText()
void SetText(const std::string& text)
```

Remarks

It is also possible to set the character value using a uint16_t code, this is done with EOCR2Char::TextCode.

EOCR2Char::SetTextCode

The true value of the character, this is used during the learning phase.

Namespace: Euresys::Open_eVision

```
[C++]
void SetTextCode(OEV_UINT16 code)
```

Remarks

It is also possible to set the character value using a std::string, this is done with EOCR2Char::Text.

4.172. EOCR2CharacterCluster Class

Holds all information related to character cluster.

Namespace: Euresys::Open_eVision

Methods

AddCharacter	Adds a character to the cluster.
Clear	Clears the cluster.
EOCR2CharacterCluster	Constructs an EOCR2CharacterCluster context.
GetCharacter	Returns a single character from the cluster.
GetCharacterCount	Returns the number of characters in this cluster.
GetCharacters	Returns all characters from this cluster.
GetCode	The ASCII code of the cluster.
operator=	Copies all the data from another EOCR2CharacterCluster object into the current EOCR2CharacterCluster object
RemoveCharacter	Removes a character from the cluster.
SetCode	The ASCII code of the cluster.

EOCR2CharacterCluster::AddCharacter

Adds a character to the cluster.

Namespace: Euresys::Open_eVision

[C++]

```
void AddCharacter(
 EOCR2DatabaseCharacter& character
 )
```

Parameters

character

The EOCR2DatabaseCharacter to be added to the database.

EOCR2CharacterCluster::GetCharacterCount

Returns the number of characters in this cluster.

Namespace: Euresys::Open_eVision

[C++]

```
int GetCharacterCount()
```

EOCR2CharacterCluster::GetCharacters

Returns all characters from this cluster.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2DatabaseCharacter> GetCharacters()
```

EOCR2CharacterCluster::Clear

Clears the cluster.

Namespace: Euresys::Open_eVision

[C++]

```
void Clear(
 )
```

EOCR2CharacterCluster::GetCode

EOCR2CharacterCluster::SetCode

The ASCII code of the cluster.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 GetCode()
void SetCode(OEV_UINT16& code)
```

EOCR2CharacterCluster::EOCR2CharacterCluster

Constructs an EOCR2CharacterCluster context.

Namespace: Euresys::Open_eVision

```
[C++]
void EOCR2CharacterCluster(
 )
void EOCR2CharacterCluster(
 const EOCR2CharacterCluster& other
 )
```

Parameters

other

EOCR2CharacterCluster object to be copied.

EOCR2CharacterCluster::GetCharacter

Returns a single character from the cluster.

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2DatabaseCharacter GetCharacter(
 const int& index
 )
```

Parameters

index

The index of this character.

EOCR2CharacterCluster::operator=

Copies all the data from another EOCR2CharacterCluster object into the current EOCR2CharacterCluster object

Namespace: Euresys::Open_eVision

[C++]

```
EOCR2CharacterCluster& operator=(
 const EOCR2CharacterCluster& other
 )
```

Parameters

other

EOCR2CharacterCluster object to be copied

EOCR2CharacterCluster::RemoveCharacter

Removes a character from the cluster.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveCharacter(
 int index
 )
```

Parameters

index

The index of the character to be removed.

4.173. EOCR2CharacterDatabase Class

Holds all information related to a character database.

Namespace: Euresys::Open_eVision

Methods

AddCharacter	Adds a single character to the database.
AddCharacters	Add characters to the database.
AddCluster	Adds the characters from an EOCR2CharacterCluster to the database
AddClusters	Adds the characters from a vector of EOCR2CharacterCluster objects to the database
ClearDatabase	Clears the database.
ClusterDatabase	Performs a clustering on the database.
EOCR2CharacterDatabase	Constructs an EOCR2CharacterDatabase context.
GetCharacter	Returns a character from the database.
GetCharacters	Returns all character from the database.

operator=	Copies all the data from another EOCR2CharacterDatabase object into the current EOCR2CharacterDatabase object
RemoveCharacter	Removes a character from the database.
Save	Saves the character database to disk.

EOCR2CharacterDatabase::AddCharacter

Adds a single character to the database.

Namespace: Euresys::Open_eVision

[C++]

```
void AddCharacter(
 const EOCR2DatabaseCharacter& character
 )

void AddCharacter(
 const EOCR2Char& character
 )
```

Parameters

character

The EOCR2DatabaseCharacter or EOCR2Char to be added to the database.

EOCR2CharacterDatabase::AddCharacters

Add characters to the database.

Namespace: Euresys::Open_eVision

[C++]

```
void AddCharacters(
 const std::vector<Euresys::Open_eVision::EOCR2DatabaseCharacter>& characters
 )

void AddCharacters(
 const std::string& path
 )

void AddCharacters(
 const std::string& path,
 Euresys::Open_eVision::EasyOCR2CharacterFilter filter
 )

void AddCharacters(
 const EOCR2Word& word
 )
```

```
void AddCharacters(
 const EOCR2Line& line
 )
void AddCharacters(
 const EOCR2Text& text
 )
```

Parameters

characters

A vector of EOCR2DatabaseCharacter objects to be added to this database.

path

The path on disk of the character database to be added to this database.

filter

An EasyOCR2CharacterFilter that tells the method which subset of the character-set should be loaded.

word

An EOCR2Word object to be added to this database.

line

An EOCR2Line object to be added to this database.

text

An EOCR2Text object to be added to this database.

EOCR2CharacterDatabase::AddCluster

Adds the characters from an EOCR2CharacterCluster to the database

Namespace: Euresys::Open_eVision

[C++]

```
void AddCluster(
 const EOCR2CharacterCluster& cluster
 )
```

Parameters

cluster

The EOCR2CharacterCluster to be added to the database.

EOCR2CharacterDatabase::AddClusters

Adds the characters from a vector of EOCR2CharacterCluster objects to the database

Namespace: Euresys::Open_eVision

[C++]

```
void AddClusters(
 const std::vector<Euresys::Open_eVision::EOCR2CharacterCluster>& clusters
 )
```

Parameters

clusters

The vector of EOCR2CharacterCluster objects to be added to the database.

EOCR2CharacterDatabase::GetCharacters

Returns all character from the database.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2DatabaseCharacter> GetCharacters()
```

EOCR2CharacterDatabase::ClearDatabase

Clears the database.

Namespace: Euresys::Open_eVision

[C++]

```
void ClearDatabase(
 )
```

EOCR2CharacterDatabase::ClusterDatabase

Performs a clustering on the database.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2CharacterCluster> ClusterDatabase(
 const int nClusters
 )
```

Parameters

nClusters

The amount of clusters to be generated.

EOCR2CharacterDatabase::EOCR2CharacterDatabase

Constructs an EOCR2CharacterDatabase context.

Namespace: Euresys::Open_eVision

```
[C++]
void EOCR2CharacterDatabase(
 )
void EOCR2CharacterDatabase(
 const EOCR2CharacterDatabase& other
 )
```

Parameters

other

EOCR2CharacterDatabase object to be copied.

EOCR2CharacterDatabase::GetCharacter

Returns a character from the database.

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2DatabaseCharacter GetCharacter(
 const int index
 )
```

Parameters

index

The index of the character to be returned.

EOCR2CharacterDatabase::operator=

Copies all the data from another EOCR2CharacterDatabase object into the current EOCR2CharacterDatabase object

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2CharacterDatabase& operator=(
 const EOCR2CharacterDatabase& other
 )
```

Parameters

other

EOCR2CharacterDatabase object to be copied

EOCR2CharacterDatabase::RemoveCharacter

Removes a character from the database.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveCharacter(
 const int index
 )
```

Parameters

index

The index of the character to be removed.

EOCR2CharacterDatabase::Save

Saves the character database to disk.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The path of the file.

serializer

The serializer.

4.174. EOCR2DatabaseCharacter Class

Holds all information related to a database character.

Namespace: Euresys::Open_eVision

Methods

EOCR2DatabaseCharacter	Constructs an EOCR2DatabaseCharacter context.
GetBitmap	The bitmap associated to this character.
GetCharacterCode	The ascii code associated to this character.
operator=	Copies all the data from another EOCR2DatabaseCharacter object into the current EOCR2DatabaseCharacter object
SetCharacterCode	The ascii code associated to this character.

EOCR2DatabaseCharacter::GetBitmap

The bitmap associated to this character.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW8& GetBitmap()
```

EOCR2DatabaseCharacter::GetCharacterCode

EOCR2DatabaseCharacter::SetCharacterCode

The ascii code associated to this character.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 GetCharacterCode()
```

```
void SetCharacterCode(OEV_UINT16 code)
```

EOCR2DatabaseCharacter::EOCR2DatabaseCharacter

Constructs an EOCR2DatabaseCharacter context.

Namespace: Euresys::Open_eVision

[C++]

```
void EOCR2DatabaseCharacter(
 )
```

```
void EOCR2DatabaseCharacter(
 const EOCR2DatabaseCharacter& other
 )
```

Parameters

other

EOCR2DatabaseCharacter object to be copied.

EOCR2DatabaseCharacter::operator=

Copies all the data from another EOCR2DatabaseCharacter object into the current EOCR2DatabaseCharacter object

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2DatabaseCharacter& operator=(
 const EOCR2DatabaseCharacter& other
 )
```

Parameters

other

EOCR2DatabaseCharacter object to be copied

4.175. EOCR2Line Class

Holds a vector of EOCR2Word objects representing a line.

Namespace: Euresys::Open_eVision

Methods

EOCR2Line	Constructs an EOCR2Line context.
GetBoundingBox	The bounding box encapsulating all characters in the line.
GetText	The true value of all characters in the line.
GetWords	The vector of EOCR2Word objects representing the line.
operator=	Copies all the data from another EOCR2Line object into the current EOCR2Line object
SetText	The true value of all characters in the line.
SetWords	The vector of EOCR2Word objects representing the line.

EOCR2Line::GetBoundingBox

The bounding box encapsulating all characters in the line.

Namespace: Euresys::Open_eVision

[C++]

```
ERectangle GetBoundingBox()
```

EOCR2Line::EOCR2Line

Constructs an EOCR2Line context.

Namespace: Euresys::Open_eVision

[C++]

```
void EOCR2Line(
 )
void EOCR2Line(
 const EOCR2Line& other
 )
```

Parameters

other

EOCR2Line object to be copied.

EOCR2Line::operator=

Copies all the data from another EOCR2Line object into the current EOCR2Line object

Namespace: Euresys::Open_eVision

[C++]

```
EOCR2Line& operator=(
 const EOCR2Line& other
 )
```

Parameters

other

EOCR2Line object to be copied

EOCR2Line::GetText

EOCR2Line::SetText

The true value of all characters in the line.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetText()
void SetText(const std::string& text)
```

Remarks

Use a space to separate two words.

EOCR2Line::GetWords

EOCR2Line::SetWords

The vector of EOCR2Word objects representing the line.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2Word> GetWords()
void SetWords(std::vector<Euresys::Open_eVision::EOCR2Word> words)
```

4.176. EOCR2Text Class

Holds a vector of EOCR2Line objects representing a text.

Namespace: Euresys::Open_eVision

Methods

EOCR2Text	Constructs an EOCR2Text context.
GetBoundingBox	The bounding box encapsulating all characters in the text.
GetLines	The vector of EOCR2Line objects representing the text.
GetText	The true value of all characters in the text.
operator=	Copies all the data from another EOCR2Text object into the current EOCR2Text object
SetLines	The vector of EOCR2Line objects representing the text.
SetText	The true value of all characters in the text.

EOCR2Text::GetBoundingBox

The bounding box encapsulating all characters in the text.

Namespace: Euresys::Open_eVision

[C++]

```
ERectangle GetBoundingBox()
```

EOCR2Text::EOCR2Text

Constructs an EOCR2Text context.

Namespace: Euresys::Open_eVision

[C++]

```
void EOCR2Text(
 )
void EOCR2Text(
 const EOCR2Text& other
 )
```

Parameters

other

EOCR2Text object to be copied.

Remarks

Default and copy constructors.

EOCR2Text::GetLines

EOCR2Text::SetLines

The vector of EOCR2Line objects representing the text.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2Line> GetLines()
void SetLines(std::vector<Euresys::Open_eVision::EOCR2Line> lines)
```

EOCR2Text::operator=

Copies all the data from another EOCR2Text object into the current EOCR2Text object

Namespace: Euresys::Open_eVision

[C++]

```
EOCR2Text& operator=(
 const EOCR2Text& other
 )
```

Parameters

other

EOCR2Text object to be copied

EOCR2Text::GetText

EOCR2Text::SetText

The true value of all characters in the text.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetText()

void SetText(const std::string& text)
```

Remarks

Use a space (" ") to separate two words and a linebreak ("\n") to separate two lines.

4.177. EOCR2Word Class

Holds a vector of EOCR2Char objects representing a word.

Namespace: Euresys::Open_eVision

Methods

EOCR2Word	Constructs an EOCR2Word context.
GetBoundingBox	The bounding box of the word, encapsulating all characters in the word.
GetCharacters	The vector of EOCR2Char objects representing the word.
GetText	The true value of all characters in the word.
operator=	Copies all the data from another EOCR2Word object into the current EOCR2Word object
SetCharacters	The vector of EOCR2Char objects representing the word.
SetText	The true value of all characters in the word.

EOCR2Word::GetBoundingBox

The bounding box of the word, encapsulating all characters in the word.

Namespace: Euresys::Open_eVision

[C++]

```
ERectangle GetBoundingBox()
```

EOCR2Word::GetCharacters

EOCR2Word::SetCharacters

The vector of EOCR2Char objects representing the word.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EOCR2Char> GetCharacters()
void SetCharacters(std::vector<Euresys::Open_eVision::EOCR2Char> characters)
```

EOCR2Word::EOCR2Word

Constructs an EOCR2Word context.

Namespace: Euresys::Open_eVision

[C++]

```
void EOCR2Word(
 )
void EOCR2Word(
 const EOCR2Word& other
 )
```

Parameters

other

EOCR2Word object to be copied.

EOCR2Word::operator=

Copies all the data from another EOCR2Word object into the current EOCR2Word object

Namespace: Euresys::Open_eVision

```
[C++]
EOCR2Word& operator=(
 const EOCR2Word& other
 )
```

Parameters

other

EOCR2Word object to be copied

EOCR2Word::GetText

EOCR2Word::SetText

The true value of all characters in the word.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetText()
void SetText(const std::string& text)
```

4.178. EPathVector Class

Vector objects are used to store 1-dimensional data.

Remarks

Using vectors is very similar to using 1-dimensional arrays, except that the size can vary at runtime. Memory allocation is handled internally. * To create a vector, use its constructor. * To fill a vector with values, first empty it, using the EPathVector member, and then add elements one at time at the tail by calling the EPathVector::AddElement member. * To access a vector element, either for reading or writing, use the [] operator. * To inquire for the current number of elements, use member EPathVector.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
Draw	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

DrawClosedContour	Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes EContourMode_ClockwiseAlwaysClosed and EContourMode_AnticlockwiseAlwaysClosed.
DrawWithCurrentPen	Draws the path. By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).
EPathVector	Constructs a vector.
GetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EPathVector object into the current EPathVector object
SetClosed	Flag indicating whether the shape built with EasyImage::Contour must be closed or not.
SetElement	Modifies the vector element at the given index by the given value.

EPathVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

```
[C++]
void AddElement(
 EPath element
 )
```

Parameters

element

The element to be added.

EPathVector::GetClosed

EPathVector::SetClosed

Flag indicating whether the shape built with EasyImage::Contour must be closed or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetClosed() const

void SetClosed(bool bClosed)
```

EPathVector::Draw

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPathVectorDrawOption drawOption,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

drawOption

Option for how to draw the path vector

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPathVector::DrawClosedContour

Draws the path vector as the closed contour of an object or a blob. The contour mode used to compute the path must be passed as argument to draw the contour correctly. This method only supports the closed contour modes ClockwiseAlwaysClosed and AnticlockwiseAlwaysClosed.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawClosedContour(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EContourMode contourMode,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

contourMode

Contour mode used to get this path vector used to draw the external boundary of the contour.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

EPathVector::DrawWithCurrentPen

This method is deprecated.

Draws the path.
By default, the path is drawn by connecting the centers of all the pixels in the path. Using the drawOption argument, you can also connect all the top left corners of the pixels (EPathVectorDrawOption_TopLeft) in the path or fill all the pixels in the path (EPathVectorDrawOption_Fill).

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Zooming factor along the X axis (1.0f means no zoom).

zoomY

Zooming factor along the Y axis (1.0f means no zoom).

originX

Abscissa of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

originY

Ordinate of the upper left corner of the plot's bounding rectangle, in pixels. By default, the upper left corner of the window is used.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPathVector::EPathVector

Constructs a vector.

Namespace: Euresys::Open_eVision

```
[C++]
void EPathVector(
 )
void EPathVector(
 OEV_UINT32 maxNumberOfElements
 )
void EPathVector(
 const EPathVector& other
 )
```

Parameters

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

other

EPathVector object to be copied

EPathVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

```
[C++]
EPath GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EPathVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EPathVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EPath& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EPathVector (excluded) of the element to be accessed.

EPathVector::operator=

Copies all the data from another EPathVector object into the current EPathVector object

Namespace: Euresys::Open_eVision

[C++]

```
EPathVector& operator=(
 const EPathVector& other
 )
```

Parameters

other

EPathVector object to be copied

EPathVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EPathVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

```
[C++]
void SetElement(
 int index,
 EPath value
 )
```

Parameters

index

Index, between 0 and EPathVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.179. EPatternFinder Class

Manages a complete finding context in EasyFind.

Namespace: Euresys::Open_eVision

Methods

CopyLearntPattern	Copies the learned pattern in the supplied image. If no pattern has been learned, an exception with code EError_NoPatternLearnt will be thrown.
DrawModel	Draws the model features with an overlay in image coordinates.
DrawModelWithCurrentPen	Draws the model features with an overlay in image coordinates.
EPatternFinder	Constructs a EPatternFinder context.
Find	Locates a set of possible occurrences of the learned pattern in the supplied search field.
GetAngleBias	Angle bias, expressed in the current angle unit.
GetAngleSearchExtent	The angular extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
GetAngleTolerance	Angle tolerance, expressed in the current angle unit.

GetContrastMode	Contrast of the instance, as defined in EFindContrastMode.
GetFeaturePoints	The features points used by the model
GetFindExtension	Extension value, that is the pattern margin size (in pixels) that is allowed to go out of the search field.
GetInterpolate	Whether interpolation is performed when searching for a pattern occurrence.
GetLearningDone	Indicates whether a pattern has already been learned.
GetLightBalance	Light balance, between [-1.0, 1.0].
GetLocalSearchMode	Sets the local search mode.
GetMaxFeaturePoints	Maximum number of feature points at the fine stage.
GetMaxInitialCandidates	Maximum number of initial candidates.
GetMaxInstances	Maximum number of instances to be found.
GetMaxOverlap	Overlapping tolerance
GetMinFeaturePoints	Minimum number of feature points at the coarse stage.
GetMinScore	Minimum score of found instances, between [-1.0, 1.0].
GetPatternType	Pattern type, as defined in EPatternType.
GetPivot	Reference point in the model.
GetPointShape	Get the point shape associated with the EPatternFinder.
GetReductionMode	The reduction mode that is to be used when learning the model (automatic or manual), as defined in EReductionMode.
GetReductionStrength	The reduction strength that is to be used when learning the model, between 0 and 1.
GetScaleBias	Scale bias, expressed in units (not in percent).
GetScaleSearchExtent	The scaling extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
GetScaleTolerance	Scale tolerance, expressed in units (not in percent).
GetThinStructureMode	Mode for EPatternType_ThinStructure, as defined in EThinStructureMode.
GetXSearchExtent	The X-axis extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
GetYSearchExtent	The Y-axis extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
Learn	Learns a given pattern and stores it in the EPatternFinder object.
Load	Loads an EPatternFinder.The given EPatternFinder must have been created for reading.
operator=	-

Save	Loads an EPatternFinder.The given EPatternFinder must have been created for writing.
SetAngleBias	Angle bias, expressed in the current angle unit.
SetAngleSearchExtent	The angular extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
SetAngleTolerance	Angle tolerance, expressed in the current angle unit.
SetContrastMode	Contrast of the instance, as defined in EFindContrastMode.
SetFindExtension	Extension value, that is the pattern margin size (in pixels) that is allowed to go out of the search field.
SetInterpolate	Whether interpolation is performed when searching for a pattern occurrence.
SetLightBalance	Light balance, between [-1.0, 1.0].
SetLocalSearchMode	Sets the local search mode.
SetMaxFeaturePoints	Maximum number of feature points at the fine stage.
SetMaxInitialCandidates	Maximum number of initial candidates.
SetMaxInstances	Maximum number of instances to be found.
SetMaxOverlap	Overlapping tolerance
SetMinFeaturePoints	Minimum number of feature points at the coarse stage.
SetMinScore	Minimum score of found instances, between [-1.0, 1.0].
SetPatternType	Pattern type, as defined in EPatternType.
SetPivot	Reference point in the model.
SetReductionMode	The reduction mode that is to be used when learning the model (automatic or manual), as defined in EReductionMode.
SetReductionStrength	The reduction strength that is to be used when learning the model, between 0 and 1.
SetScaleBias	Scale bias, expressed in units (not in percent).
SetScaleSearchExtent	The scaling extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
SetScaleTolerance	Scale tolerance, expressed in units (not in percent).
SetThinStructureMode	Mode for EPatternType_ThinStructure, as defined in EThinStructureMode.
SetXSearchExtent	The X-axis extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.
SetYSearchExtent	The Y-axis extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.

EPatternFinder::GetAngleBias

EPatternFinder::SetAngleBias

Angle bias, expressed in the current angle unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngleBias() const
void SetAngleBias(float f32AngleBias)
```

Remarks

The AngleBias defines the angle offset between the model and the instances. Finding the pattern is performed in range AngleBias +/- EPatternFinder::AngleTolerance. This range should not exceed a full turn. Default: 0.0.

EPatternFinder::GetAngleSearchExtent

EPatternFinder::SetAngleSearchExtent

The angular extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.

Namespace: Euresys::Open_eVision

[C++]

```
int GetAngleSearchExtent() const
void SetAngleSearchExtent(int n32Extent)
```

EPatternFinder::GetAngleTolerance

EPatternFinder::SetAngleTolerance

Angle tolerance, expressed in the current angle unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngleTolerance() const
void SetAngleTolerance(float f32AngleTolerance)
```

Remarks

The AngleTolerance defines the angle allowance of the instances around the EPatternFinder::AngleBias. Finding the pattern is performed in range EPatternFinder::AngleBias +/- AngleTolerance. This range should not exceed a full turn. A NULL tolerance can be set, in which case the angle bias value is assumed. Default: 0.0.

EPatternFinder::GetContrastMode

EPatternFinder::SetContrastMode

Contrast of the instance, as defined in EFindContrastMode.

Namespace: Euresys::Open_eVision

```
[C++]

Euresys::Open_eVision::EFindContrastMode GetContrastMode() const

void SetContrastMode(Euresys::Open_eVision::EFindContrastMode eContrastMode)
```

Remarks

This is a EPatternType_ConsistentEdges pattern type property. It defines the contrast of regions. Contrast can be normal (as in the model), inverse (inverse contrast of the model), or any (same or inverse contrast of the model). Default: EFindContrastMode_Normal.

EPatternFinder::CopyLearntPattern

Copies the learned pattern in the supplied image. If no pattern has been learned, an exception with code NoPatternLearnt will be thrown.

Namespace: Euresys::Open_eVision

```
[C++]

void CopyLearntPattern(
 EImageBW8& image
 )
```

Parameters

image

Pointer to the image in which the learned pattern will be returned.

EPatternFinder::DrawModel

Draws the model features with an overlay in image coordinates.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawModel(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawModel(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawModel(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination windows.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to 0, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPatternFinder::DrawModelWithCurrentPen

This method is deprecated.

Draws the model features with an overlay in image coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawModelWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination windows.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor. If set to 0, the horizontal zooming factor will be used for isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPatternFinder::EPatternFinder

Constructs a EPatternFinder context.

Namespace: Euresys::Open_eVision

```
[C++]
void EPatternFinder(
 )

void EPatternFinder(
 const EPatternFinder& other
 )
```

Parameters

other

Another EPatternFinder object to be copied in the new EPatternFinder object.

Remarks

All properties are initialized to their respective default values.

EPatternFinder::GetFeaturePoints

The features points used by the model

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EFindFeaturePoint> GetFeaturePoints() const
```

EPatternFinder::Find

Locates a set of possible occurrences of the learned pattern in the supplied search field.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EFoundPattern> Find(
 EROIBW8* source
 )

std::vector<Euresys::Open_eVision::EFoundPattern> Find(
 EROIBW8* source,
 const ERegion& region
 )
```

Parameters

source

Image or part of an image in which the learned model has to be searched for.

region

Region into the ROI where the search is performed.

Remarks

This method will fail if no pattern has been learned previously. The result is a vector of EFoundPattern objects. EFoundPattern instances can be retrieved by using the Vector<>::operator[] method.

EPatternFinder::GetFindExtension

EPatternFinder::SetFindExtension

Extension value, that is the pattern margin size (in pixels) that is allowed to go out of the search field.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFindExtension() const

void SetFindExtension(int n32Extension)
```

Remarks

When a non-NULL value is attributed to the extension, the detection of instances partially out of the ROI is allowed. The extension value defines how much the ROI is extended. Default: 0.

EPatternFinder::GetInterpolate

EPatternFinder::SetInterpolate

Whether interpolation is performed when searching for a pattern occurrence.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetInterpolate() const

void SetInterpolate(bool bInterpolate)
```

Remarks

By default, matching is done with a one-pixel precision for all degrees of freedom (translation, rotation and scaling). You can use an additional interpolation process to achieve sub-pixel accuracy. This generally leads to an improvement of the sub-pixel accuracy by a factor larger than 10. This is possible only when the found instances match closely the model. A score higher than 0.99 indicates that the instances are a close match of the model. In other words, the instance is considered to be more accurate when the score is higher. The added computational cost is low. Default: true.

EPatternFinder::Learn

Learns a given pattern and stores it in the EPatternFinder object.

Namespace: Euresys::Open_eVision

[C++]

```
void Learn(
 const EROIBW8* pattern,
 EROIBW8* dontCare
 )

void Learn(
 const EROIBW8* pattern,
 const ERegion& region
 )
```

```
void Learn(
 EVectorModel& model
 )
```

Parameters

pattern

Model to be learned (ROI).

dontCare

"Don't care" area mask (ROI).

region

Region into the ROI where the learning is performed

model

-

Remarks

Learning another pattern erases the information stored for the first one. A "don't care area" can be set as argument, allowing to mask and not take into account certain parts of the pattern while learning. The "don't care area" mask should have the same size as the pattern or as its eventual parent image. The mask should be a bi-level image with pixel - values of '0' for ignored areas and '255' otherwise.

EPatternFinder::GetLearningDone

Indicates whether a pattern has already been learned.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetLearningDone() const
```

EPatternFinder::GetLightBalance

EPatternFinder::SetLightBalance

Light balance, between [-1.0, 1.0].

Namespace: Euresys::Open_eVision

[C++]

```
float GetLightBalance() const
void SetLightBalance(float f32LightBalance)
```

Remarks

In the EPatternType_ConsistentEdges and EPatternType_ThinStructure modes, the LightBalance property governs the selection of the feature points while learning the model. It drives which edge points are eligible as feature points in the model, by defining a criterion for ignoring those edge points that are not sharp enough. As a consequence, this property will influence the spatial distribution of the feature points. In the aforementioned operating modes, the feature points are the places in the image that exhibit a strong variation in the gray level signal. Mathematically, these places are those at which the magnitude of the gradient is significant. The LightBalance property defines the way the latter threshold on the magnitude of the gradient is computed, through a careful analysis of the dynamics of gradient. The more the LightBalance tends to -1, the more tolerant will be the threshold, and the more edge points will be considered as candidates for becoming feature points. Conversely, as the LightBalance property becomes close to 1, only the points with a high gradient magnitude are taken into consideration. In other words, a small LightBalance defines a loose criterion for defining what an edge point is, whereas a great value implies a conservative criterion. By default, this property is fixed to 0.0. This is an appropriate value for most images which are encountered in industrial machine vision. The LightBalance is automatically set to 0.0 after a learning process. Once the LightBalance is changed, a new learning process has to be done to take the new value into account. An efficient way to see the effect of changing this property is to use the EPatternFinder::DrawModel method.

EPatternFinder::Load

Loads an EPatternFinder.The given EPatternFinder must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path,
 bool daughters
 )

void Load(
 ESerializer* serializer,
 bool daughters
 )
```

Parameters

path

The file path.

daughters

Indicates if the load must be done on the whole hierarchy or just this object.

serializer

Pointer to the ESerializer created for reading.

EPatternFinder::GetLocalSearchMode

EPatternFinder::SetLocalSearchMode

Sets the local search mode.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::ELocalSearchMode GetLocalSearchMode() const
void SetLocalSearchMode(Euresys::Open_eVision::ELocalSearchMode eLocalSearchMode)
```

Remarks

In the multi-stage approach of EasyFind, pattern occurrence candidates are at first found at the coarsest stage. Then, at each of the following stages, their position and score are refined until the last and finest one. This refining is achieved by searching for better candidates in the neighborhood of each of the ones found in the previous stage. The local search mode allows the user to set the extent of this neighborhood. By default, the local search mode is set to ELocalSearchMode_Basic.

EPatternFinder::GetMaxFeaturePoints

EPatternFinder::SetMaxFeaturePoints

Maximum number of feature points at the fine stage.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMaxFeaturePoints() const
void SetMaxFeaturePoints(OEV_UINT32 un32MaxFeaturePoints)
```

Remarks

Default: 1024. Reserved use.

EPatternFinder::GetMaxInitialCandidates

EPatternFinder::SetMaxInitialCandidates

Maximum number of initial candidates.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMaxInitialCandidates() const
void SetMaxInitialCandidates(OEV_UINT32 un32MaxCandidate)
```

Remarks

During the search for matching patterns, EasyFind considers a set of candidates that it progressively refines. The maximum number of initial candidates must be greater or equal than the number of instances to be found (see EPatternFinder::MaxInstances). A large number of initial candidates can help finding difficult or partial match but at the cost of increasing processing time. A small number can help speeding up the find process. By default, the value for maximum number of initial candidates is 0, indicating that the value is chosen internally.

EPatternFinder::GetMaxInstances

EPatternFinder::SetMaxInstances

Maximum number of instances to be found.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMaxInstances() const
void SetMaxInstances(OEV_UINT32 un32MaxInstances)
```

Remarks

Default: 1.

EPatternFinder::GetMaxOverlap

EPatternFinder::SetMaxOverlap

Overlapping tolerance

Namespace: Euresys::Open_eVision

[C++]

```
float GetMaxOverlap() const
void SetMaxOverlap(float f32MaximumOverlapping)
```

Remarks

0.0 means all found patterns must be disconnected, 1.0 means they can fully overlap. Default: 1.0.

EPatternFinder::GetMinFeaturePoints

EPatternFinder::SetMinFeaturePoints

Minimum number of feature points at the coarse stage.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMinFeaturePoints() const
void SetMinFeaturePoints(OEV_UINT32 un32MinFeaturePoints)
```

Remarks

Default: 8. Reserved use.

EPatternFinder::GetMinScore

EPatternFinder::SetMinScore

Minimum score of found instances, between [-1.0, 1.0].

Namespace: Euresys::Open_eVision

```
[C++]
float GetMinScore() const
void SetMinScore(float f32MinScore)
```

Remarks

Instances with a score under the MinScore will not be returned.

EPatternFinder::operator=

-

Namespace: Euresys::Open_eVision

```
[C++]
EPatternFinder& operator=(
 const EPatternFinder& other
 )
```

Parameters

other

-

EPatternFinder::GetPatternType

EPatternFinder::SetPatternType

Pattern type, as defined in EPatternType.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EPatternType GetPatternType() const
void SetPatternType(Euresys::Open_eVision::EPatternType patternType)
```

Remarks

This property informs the EPatternFinder of the general nature of the model to be learned. Default: EPatternType_ConsistentEdges.

EPatternFinder::GetPivot

EPatternFinder::SetPivot

Reference point in the model.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPivot() const
void SetPivot(const EPoint& pivot_)
```

Remarks

The coordinates of the reference point are relative to the upper left corner of the model. The location of an instance (Coordinates (X,Y)) is the location of its reference point defined in the model. By default, the pivot is a EPoint set to the pattern center. EPoint is a structure that contains two x and y float values.

EPatternFinder::GetPointShape

Get the point shape associated with the EPatternFinder.

Namespace: Euresys::Open_eVision

[C++]

```
const EPointShape& GetPointShape() const
```

EPatternFinder::GetReductionMode

EPatternFinder::SetReductionMode

The reduction mode that is to be used when learning the model (automatic or manual), as defined in EReductionMode.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EReductionMode GetReductionMode() const
void SetReductionMode(Euresys::Open_eVision::EReductionMode eReductionMode)
```

Remarks

Specifies whether the best-guess method should be used to assert the level of reduction that will be used when learning the model. If this property is set to EReductionMode_Manual, it is up to the user to provide a suitable reduction strength. This value is only used when learning the model. Default: EReductionMode_Auto, which means that the best-guess algorithm is used by default.

EPatternFinder::GetReductionStrength

EPatternFinder::SetReductionStrength

The reduction strength that is to be used when learning the model, between 0 and 1.

Namespace: Euresys::Open_eVision

```
[C++]
float GetReductionStrength() const
void SetReductionStrength(float f32ReductionStrength)
```

Remarks

Specifies the reduction strength for learning the model (encoded as a percentage). Its precise semantics depends on the reduction mode (see the EPatternFinder::ReductionMode property): * In the automatic reduction mode, its value is undefined until a model is learned. When a model is learned (i.e. after a call to EPatternFinder::Learn), the value of this property can be read, in which case it reflects the reduction strength that has been automatically chosen by the best-guess algorithm. * In the manual reduction mode, this property must be set by the user and is kept constant throughout the entire lifetime of the object. The new value of the property is only used at the following call to EPatternFinder::Learn. This value only has an effect when learning the model. Default: The default value depends on the value of the EPatternFinder::ReductionMode property. Allowed values: Floating-point number in the interval [0..1].

EPatternFinder::Save

Loads an EPatternFinder.The given EPatternFinder must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path,
 bool daughters
 )

void Save(
 ESerializer* serializer,
 bool daughters
 )
```

Parameters

path

The file path.

daughters

Indicates if the save must be done on the whole hierarchy or just this object.

serializer

Pointer to the ESerializer created for writing.

EPatternFinder::GetScaleBias

EPatternFinder::SetScaleBias

Scale bias, expressed in units (not in percent).

Namespace: Euresys::Open_eVision

[C++]

```
float GetScaleBias() const

void SetScaleBias(float f32ScaleBias)
```

Remarks

The ScaleBias defines the scale factor between the model and the instances. Finding the pattern is performed in range ScaleBias +/- ScaleTolerance. This range should not exceed [0.5..2.5] (50 % to 250 % scaling). Default: 1.0 (100 %).

EPatternFinder::GetScaleSearchExtent

EPatternFinder::SetScaleSearchExtent

The scaling extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.

Namespace: Euresys::Open_eVision

[C++]

```
int GetScaleSearchExtent() const

void SetScaleSearchExtent(int n32Extent)
```

EPatternFinder::GetScaleTolerance

EPatternFinder::SetScaleTolerance

Scale tolerance, expressed in units (not in percent).

Namespace: Euresys::Open_eVision

[C++]

```
float GetScaleTolerance() const

void SetScaleTolerance(float f32ScaleTolerance)
```

Remarks

The ScaleTolerance defines the scale allowance of the instances around the EPatternFinder::ScaleBias. Finding the pattern is performed in range EPatternFinder::ScaleBias +/- ScaleTolerance. This range should not exceed [0.5..2] (50 % to 200 % scaling). A NULL tolerance can be set, in which case the scale bias value is assumed. Default: 0.0.

EPatternFinder::GetThinStructureMode

EPatternFinder::SetThinStructureMode

Mode for ThinStructure, as defined in EThinStructureMode.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EThinStructureMode GetThinStructureMode() const

void SetThinStructureMode(Euresys::Open_eVision::EThinStructureMode thinStructureMode)
```

Remarks

EThinStructureMode informs EasyFind if thin elements in the model are darker or brighter than regions. Default: EThinStructureMode_Auto, which detects the best mode between dark or bright.

EPatternFinder::GetXSearchExtent

EPatternFinder::SetXSearchExtent

The X-axis extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.

Namespace: Euresys::Open_eVision

[C++]

```
int GetXSearchExtent() const
void SetXSearchExtent(int n32Extent)
```

EPatternFinder::GetYSearchExtent

EPatternFinder::SetYSearchExtent

The Y-axis extension of the search neighborhood. Only odd values are allowed for this parameter. See the EPatternFinder::LocalSearchMode property description for further details.

Namespace: Euresys::Open_eVision

[C++]

```
int GetYSearchExtent() const
void SetYSearchExtent(int n32Extent)
```

4.180. EPeakVector Class

Represents a vector of profile peaks.

Base Class:EVector

Namespace: Euresys::Open_eVision

Methods

AddElement	Appends (adds at the tail) an element to the vector.
EPeakVector	Constructs a vector.

GetElement	Returns the vector element at the given index.
GetRawDataPtr	Pointer to the vector data.
operator[]	Gives access to the vector element at the given index.
operator=	Copies all the data from another EPeakVector object into the current EPeakVector object
SetElement	Modifies the vector element at the given index by the given value.

EPeakVector::AddElement

Appends (adds at the tail) an element to the vector.

Namespace: Euresys::Open_eVision

[C++]

```
void AddElement(
 EPeak element
 )
```

Parameters

element

The element to be added.

EPeakVector::EPeakVector

Constructs a vector.

Namespace: Euresys::Open_eVision

[C++]

```
void EPeakVector(
 )

void EPeakVector(
 const EPeakVector& other
 )

void EPeakVector(
 OEV_UINT32 maxNumberOfElements
 )
```

Parameters

other

EPeakVector object to be copied

maxNumberOfElements

Optionally, memory can be pre-allocated to accommodate a given number of elements.

EPeakVector::GetElement

Returns the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EPeak GetElement(
 int index
 )
```

Parameters

index

Index, between 0 and EPeakVector (excluded) of the element to be accessed.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

EPeakVector::operator[]

Gives access to the vector element at the given index.

Namespace: Euresys::Open_eVision

[C++]

```
EPeak& operator[](
 OEV_UINT32 index
 )
```

Parameters

index

Index, between 0 and EPeakVector (excluded) of the element to be accessed.

EPeakVector::operator=

Copies all the data from another EPeakVector object into the current EPeakVector object

Namespace: Euresys::Open_eVision

[C++]

```
EPeakVector& operator=(
 const EPeakVector& other
 )
```

Parameters

other

EPeakVector object to be copied

EPeakVector::GetRawDataPtr

Pointer to the vector data.

Namespace: Euresys::Open_eVision

[C++]

```
void* GetRawDataPtr() const
```

EPeakVector::SetElement

Modifies the vector element at the given index by the given value.

Namespace: Euresys::Open_eVision

[C++]

```
void SetElement(
 int index,
 EPeak value
 )
```

Parameters

index

Index, between 0 and EPeakVector (excluded), of the element to be modified.

value

The new value for the element.

Remarks

If the given index is outside the bounds of the vector, the error code EError_Parameter1OutOfRange is set.

4.181. EPhotometricStereoImager Class

Manages photometric stereo reconstruction. This class can build normals, albedos, gradients, mean curvatures and gaussian curvatures images from at least 3 input images by using photometric stereo The algorithm used assumes objects are Lambertian and light sources emit parallel rays

Namespace: Euresys::Open_eVision::Easy3D

Methods

CalibrateFromSphere	Calibrates the light directions on several images (at least 3) of a sphere (or half sphere) and returns a score indicating the reliability of the calibration.
Compute	Compute the different photometric stereo images of an object. To retrieve them, see EPhotometricStereoImager::Normals, EPhotometricStereoImager::GetAlbedos, EPhotometricStereoImager::GradientsX, EPhotometricStereoImager::GradientsY, EPhotometricStereoImager::ComputeGaussianCurvatures and EPhotometricStereoImager::ComputeMeanCurvatures.
ComputeGaussianCurvatures	Computes the gaussian curvatures as an EImageBW8. Gaussian curvatures are inherently floating point images so a conversion must be specified. In all cases, a floating point value of 0 translates to an EBW8 value of 128. See EPhotometricStereoContrast for the different conversions.
ComputeHeightMap	Computes the height at each pixel as an EZMap8 by integrating the surface gradients.
ComputeMeanCurvatures	Computes the mean curvatures as an EImageBW8. Mean curvatures are inherently floating point images so a conversion must be specified. In all cases, a floating point value of 0 translates to an EBW8 value of 128. See EPhotometricStereoContrast for the different conversions.
ConfigureNonUniformLightingCorrection	Configure the non uniform lighting correction of the scene by using flat images. Photometric stereo assumes the lights of each image to be of same direction and intensity. This is however rarely the case in practice, a way to attenuate the problem is to use flat images to correct non-uniform lighting before performing photometric stereo.
EPhotometricStereoImager	Creates an EPhotometricStereoImager object.
GetAlbedos	Gets the albedos as an EImageBW8. Albedos are inherently floating point images so a conversion must be specified. In all cases, a floating point value of 0 translates to an EBW8 value of 0.
GetCalibrationAngles	Gets the calibration angles.
GetCalibrationAzimuthAngles	Gets the calibration azimuth angles.
GetCalibrationElevationAngles	Gets the calibration elevation angles.
GetEnableNonUniformLightingCorrection	Gets/Sets whether the correction of the non uniformity of the lighting is corrected or not (disabled by default, enabled by calling EPhotometricStereoImager::ConfigureNonUniformLightingCorrection). Disabling it increases speed.
GetGradientsX	Gets the gradients on the X axis as an EImageBW8.
GetGradientsY	Gets the gradients on the Y axis as an EImageBW8.
GetNormals	Gets the normals as an EImageC24.

Load	Loads the EPhotometricStereoImager. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the EPhotometricStereoImager. The given ESerializer must have been created for writing.
SetCalibrationAngles	Sets the calibration angles
SetEnableNonUniformLightingCorrection	Gets/Sets whether the correction of the non uniformity of the lighting is corrected or not (disabled by default, enabled by calling EPhotometricStereoImager::ConfigureNonUniformLightingCorrection). Disabling it increases speed.

EPhotometricStereoImager::CalibrateFromSphere

Calibrates the light directions on several images (at least 3) of a sphere (or half sphere) and returns a score indicating the reliability of the calibration.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float CalibrateFromSphere(
 const std::vector<Euresys::Open_eVision::EROIBW8>& sphereImages
 )

float CalibrateFromSphere(
 const std::vector<Euresys::Open_eVision::EROIBW8>& sphereImages,
 const EROIBW8& darkImage
 )

float CalibrateFromSphere(
 const std::vector<Euresys::Open_eVision::EROIBW8>& sphereImages,
 const ECircle& circle
 )

float CalibrateFromSphere(
 const std::vector<Euresys::Open_eVision::EROIBW8>& sphereImages,
 const EROIBW8& darkImage,
 const ECircle& circle
 )

float CalibrateFromSphere(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4
 )
```

```
float CalibrateFromSphere(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4,
 const EROIBW8& darkImage
 )
float CalibrateFromSphere(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4,
 const ECircle& circle
 )
float CalibrateFromSphere(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4,
 const EROIBW8& darkImage,
 const ECircle& circle
 )
```

Parameters

sphereImages

A vector of EROIBW8 of a sphere.

darkImage

An image of the object when there is no specific illumination. This argument can help to improve the calibration especially if the image is not dark when there is no illumination. Otherwise, it is not useful.

circle

An ECircle that represents the position of the sphere. Default: automatically computed.

image1

The first EROIBW8 of a sphere in a 4-image configuration.

image2

The second EROIBW8 of a sphere in a 4-image configuration.

image3

The third EROIBW8 of a sphere in a 4-image configuration.

image4

The fourth EROIBW8 of a sphere in a 4-image configuration.

EPhotometricStereoImager::GetCalibrationAzimuthAngles

Gets the calibration azimuth angles.

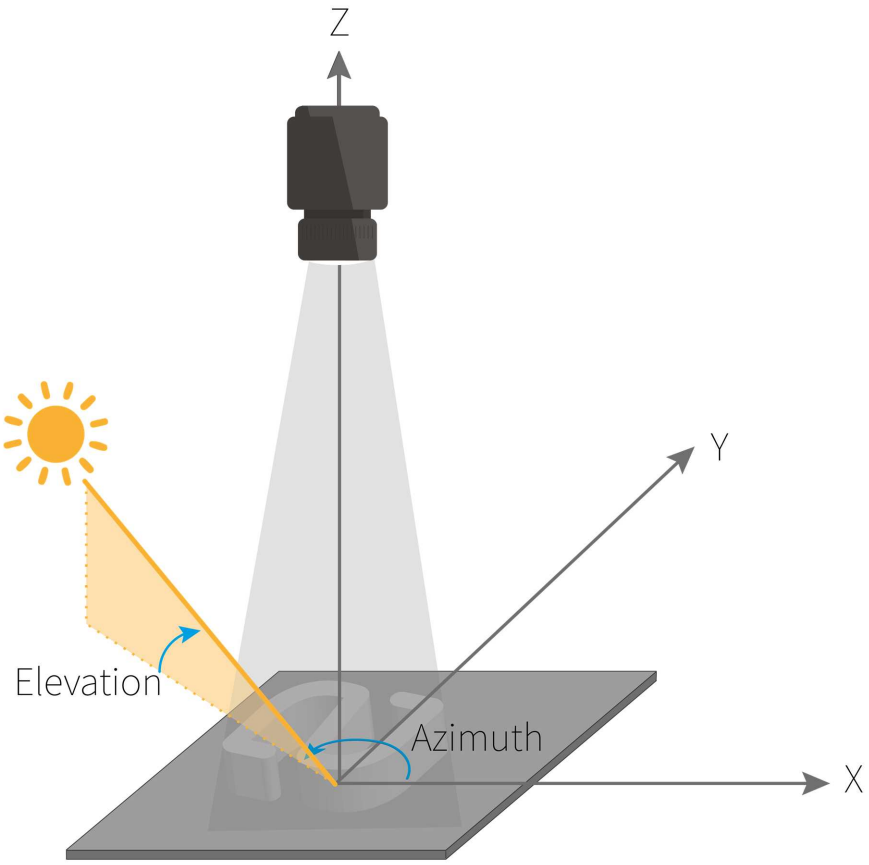
Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<float> GetCalibrationAzimuthAngles() const
```

Remarks

When facing the image, the axis x points right, y points top and z points towards the camera. Azimuth angles are oriented trigonometrically around the z axis. A light source on the right of the image would have an azimuth of 0 degrees. One on top would have an azimuth of 90 degrees.



EPhotometricStereoImager::GetCalibrationElevationAngles

Gets the calibration elevation angles.

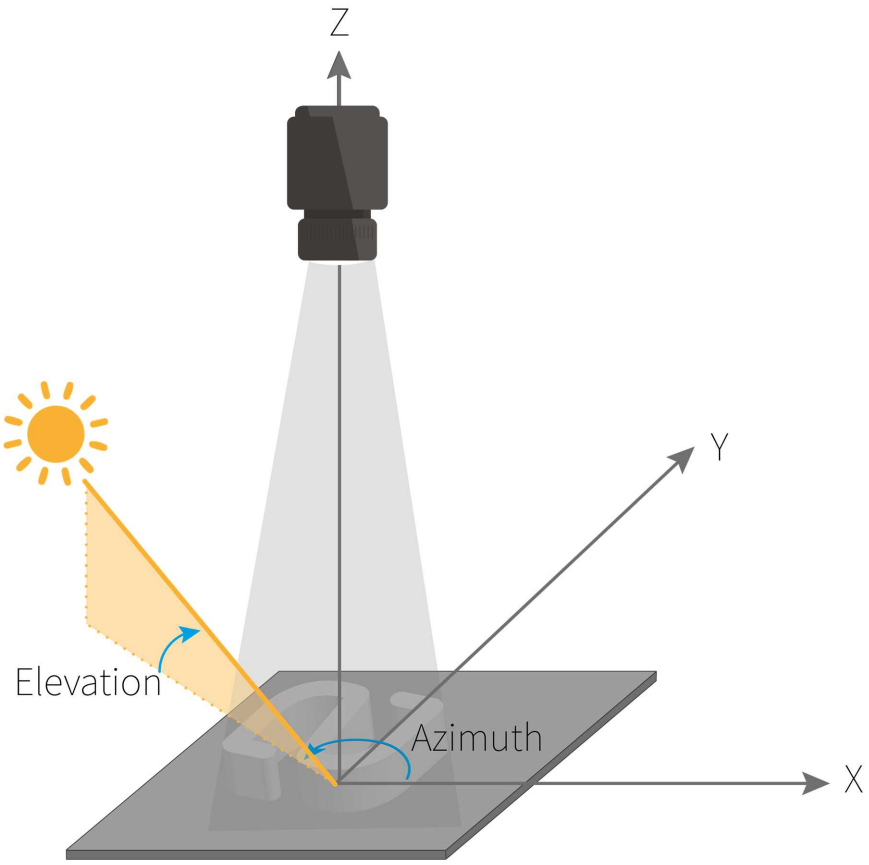
Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<float> GetCalibrationElevationAngles() const
```

Remarks

When facing the image, the axis x points right, y points top and z points towards the camera. Elevation is the angle formed by the base plane and the light source. A light source on the horizon would have an elevation of 0 degrees. One on the camera would have an elevation of 90 degrees.



EPhotometricStereoImager::Compute

Compute the different photometric stereo images of an object. To retrieve them, see EPhotometricStereoImager::Normals, EPhotometricStereoImager::GetAlbedos, EPhotometricStereoImager::GradientsX, EPhotometricStereoImager::GradientsY, EPhotometricStereoImager::ComputeGaussianCurvatures and EPhotometricStereoImager::ComputeMeanCurvatures.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Compute(
 const std::vector<Euresys::Open_eVision::EROIBW8>& objectImages
 )

void Compute(
 const std::vector<Euresys::Open_eVision::EROIBW8>& objectImages,
 const ERegion& region
 )

void Compute(
 const std::vector<Euresys::Open_eVision::EROIBW8>& objectImages,
 const EROIBW8& darkImage
 )

void Compute(
 const std::vector<Euresys::Open_eVision::EROIBW8>& objectImages,
 const EROIBW8& darkImage,
 const ERegion& region
 )

void Compute(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4
 )

void Compute(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4,
 const ERegion& region
 )

void Compute(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4,
 const EROIBW8& darkImage
 )
```

```
void Compute(
 const EROIBW8& image1,
 const EROIBW8& image2,
 const EROIBW8& image3,
 const EROIBW8& image4,
 const EROIBW8& darkImage,
 const ERegion& region
 )
```

Parameters

objectImages

A vector of EROIBW8 of the object to reconstruct. The images must be in the same order as the calibration images/angles with respect to the light direction. If flat images are used, flat and object images must have the same size

region

ERegion within which the computation will be done.

darkImage

An image of the object when there is no specific illumination. This argument can help to improve the results especially if the image is not dark when there is no illumination. Otherwise, it is not useful.

image1

The first EROIBW8 of the object in a 4-image configuration.

image2

The second EROIBW8 of the object in a 4-image configuration.

image3

The third EROIBW8 of the object in a 4-image configuration.

image4

The fourth EROIBW8 of the object in a 4-image configuration.

EPhotometricStereoImager::ComputeGaussianCurvatures

Computes the gaussian curvatures as an EImageBW8. Gaussian curvatures are inherently floating point images so a conversion must be specified. In all cases, a floating point value of 0 translates to an EBW8 value of 128. See EPhotometricStereoContrast for the different conversions.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EImageBW8& ComputeGaussianCurvatures(
 Euresys::Open_eVision::Easy3D::EPhotometricStereoContrast contrast
 )

const EImageBW8& ComputeGaussianCurvatures(
 Euresys::Open_eVision::Easy3D::EPhotometricStereoContrast contrast,
 float maxAbsoluteValue
 )
```

Parameters

contrast

an EPhotometricStereoContrast

maxAbsoluteValue

when contrast is EPhotometricStereoContrast_FixedRange, this parameter can be used to specify the maximal floating point value for the gaussian curvatures, otherwise it is ignored. All values outside of [-maxAbsoluteValue, +maxAbsoluteValue] are clipped. maxAbsoluteValue must be positive and defaults to 2.

Remarks

Both gaussian and mean curvatures are computed in this method and are cached to avoid unnecessary computations.

EPhotometricStereoImager::ComputeHeightMap

Computes the height at each pixel as an EZMap8 by integrating the surface gradients.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EZMap8& ComputeHeightMap(
 )
```

EPhotometricStereoImager::ComputeMeanCurvatures

Computes the mean curvatures as an EImageBW8. Mean curvatures are inherently floating point images so a conversion must be specified. In all cases, a floating point value of 0 translates to an EBW8 value of 128. See EPhotometricStereoContrast for the different conversions.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EImageBW8& ComputeMeanCurvatures(
 Euresys::Open_eVision::Easy3D::EPhotometricStereoContrast contrast
 )

const EImageBW8& ComputeMeanCurvatures(
 Euresys::Open_eVision::Easy3D::EPhotometricStereoContrast contrast,
 float maxAbsoluteValue
 )
```

Parameters

contrast

an EPhotometricStereoContrast

maxAbsoluteValue

when contrast is EPhotometricStereoContrast_FixedRange, this parameter can be used to specify the maximal floating point value for the mean curvatures, otherwise it is ignored. All values outside of [-maxAbsoluteValue, +maxAbsoluteValue] are clipped. maxAbsoluteValue must be positive and defaults to 3.

Remarks

Both mean and gaussian curvatures are computed in this method and are cached to avoid unnecessary computations.

EPhotometricStereoImager::ConfigureNonUniformLightingCorrection

Configure the non uniform lighting correction of the scene by using flat images. Photometric stereo assumes the lights of each image to be of same direction and intensity. This is however rarely the case in practice, a way to attenuate the problem is to use flat images to correct non-uniform lighting before performing photometric stereo.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConfigureNonUniformLightingCorrection(
 const std::vector<Euresys::Open_eVision::EROIBW8>& flatImages
 )

void ConfigureNonUniformLightingCorrection(
 const std::vector<Euresys::Open_eVision::EROIBW8>& flatImages,
 const EROIBW8& darkImage
 )
```

Parameters

flatImages

A vector of EROIBW8 of the flat images. The images must be in the same order as the calibration images/angles with respect to the light direction.

darkImage

An image of the object when there is no specific illumination. This argument can help to improve the results especially if the image is not dark when there is no illumination. Otherwise, it is not useful.

EPhotometricStereoImager::GetEnableNonUniformLightingCorrection

EPhotometricStereoImager::SetEnableNonUniformLightingCorrection

Gets/Sets whether the correction of the non uniformity of the lighting is corrected or not (disabled by default, enabled by calling EPhotometricStereoImager::ConfigureNonUniformLightingCorrection). Disabling it increases speed.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetEnableNonUniformLightingCorrection() const
void SetEnableNonUniformLightingCorrection(bool state)
```

Remarks

It must be configured (EPhotometricStereoImager::ConfigureNonUniformLightingCorrection) before being enabled.

EPhotometricStereoImager::EPhotometricStereoImager

Creates an EPhotometricStereoImager object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EPhotometricStereoImager(
 )
void EPhotometricStereoImager(
 const EPhotometricStereoImager& other
 )
```

Parameters

other

Another EPhotometricStereoImager.

EPhotometricStereoImager::GetAlbedos

Gets the albedos as an EImageBW8. Albedos are inherently floating point images so a conversion must be specified. In all cases, a floating point value of 0 translates to an EBW8 value of 0.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EImageBW8& GetAlbedos(
 Euresys::Open_eVision::Easy3D::EPhotometricStereoContrast contrast
 )
const EImageBW8& GetAlbedos(
 Euresys::Open_eVision::Easy3D::EPhotometricStereoContrast contrast,
 float maxValue
 )
```

Parameters

contrast

an EPhotometricStereoContrast

maxValue

when contrast is EPhotometricStereoContrast_FixedRange, this parameter can be used to specify the maximal floating point value for the albedos, otherwise it is ignored. All values bigger than the parameter are clipped. maxValue must be positive and defaults to 200.

Remarks

Albedos are computed in EPhotometricStereoImager::Compute.

EPhotometricStereoImager::GetCalibrationAngles

Gets the calibration angles.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetCalibrationAngles(
 std::vector<float>& azimuths,
 std::vector<float>& elevations
 )
```

Parameters

azimuths

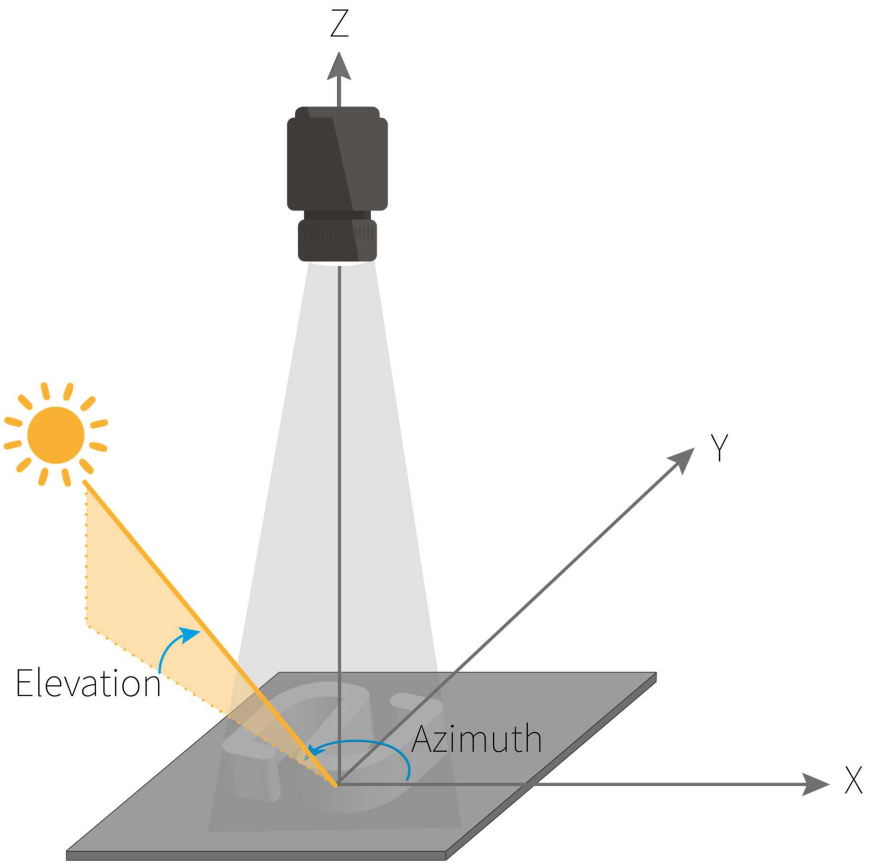
The vector of azimuth angles to retrieve

elevations

The vector of elevation angles to retrieve

Remarks

When facing the image, the axis x points right, y points top and z points towards the camera. Azimuth angles are oriented trigonometrically around the z axis. A light source on the right of the image would have an azimuth of 0 degrees. One on top would have an azimuth of 90 degrees. Elevation is the angle formed by the base plane and the light source. A light source on the horizon would have an elevation of 0 degrees. One on the camera would have an elevation of 90 degrees.



EPhotometricStereoImager::GetGradientsX

Gets the gradients on the X axis as an EImageBW8.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EImageBW8& GetGradientsX()
```

Remarks

Gradients on the X axis are computed in EPhotometricStereoImager::Compute.

EPhotometricStereoImager::GetGradientsY

Gets the gradients on the Y axis as an EImageBW8.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EImageBW8& GetGradientsY()
```

Remarks

Gradients on the Y axis are computed in EPhotometricStereoImager::Compute.

EPhotometricStereoImager::Load

Loads the EPhotometricStereoImager. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

Remarks

Neither photometric stereo results nor intermediary computations are serialized. Thus, results of a call to EPhotometricStereoImager::Compute performed before serialization cannot be retrieved after serialization. On the other hand, the results of the calibration and the NonUniformLightingCorrection are serialized and can be used again.

EPhotometricStereoImager::GetNormals

Gets the normals as an EImageC24.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EImageC24& GetNormals() const
```

Remarks

Normals are computed in EPhotometricStereoImager::Compute.

EPhotometricStereoImager::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EPhotometricStereoImager& operator=(
 const EPhotometricStereoImager& other
 )
```

Parameters

other

Another EPhotometricStereoImager.

EPhotometricStereoImager::Save

Saves the EPhotometricStereoImager. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

Remarks

Neither photometric stereo results nor intermediary computations are serialized. Thus, results of a call to EPhotometricStereoImager::Compute performed before serialization cannot be retrieved after serialization. On the other hand, the results of the calibration and the NonUniformLightingCorrection are serialized and can be used again.

EPhotometricStereoImager::SetCalibrationAngles

Sets the calibration angles

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetCalibrationAngles(
 const std::vector<float>& azimuths,
 const std::vector<float>& elevations
 )
```

Parameters

azimuths

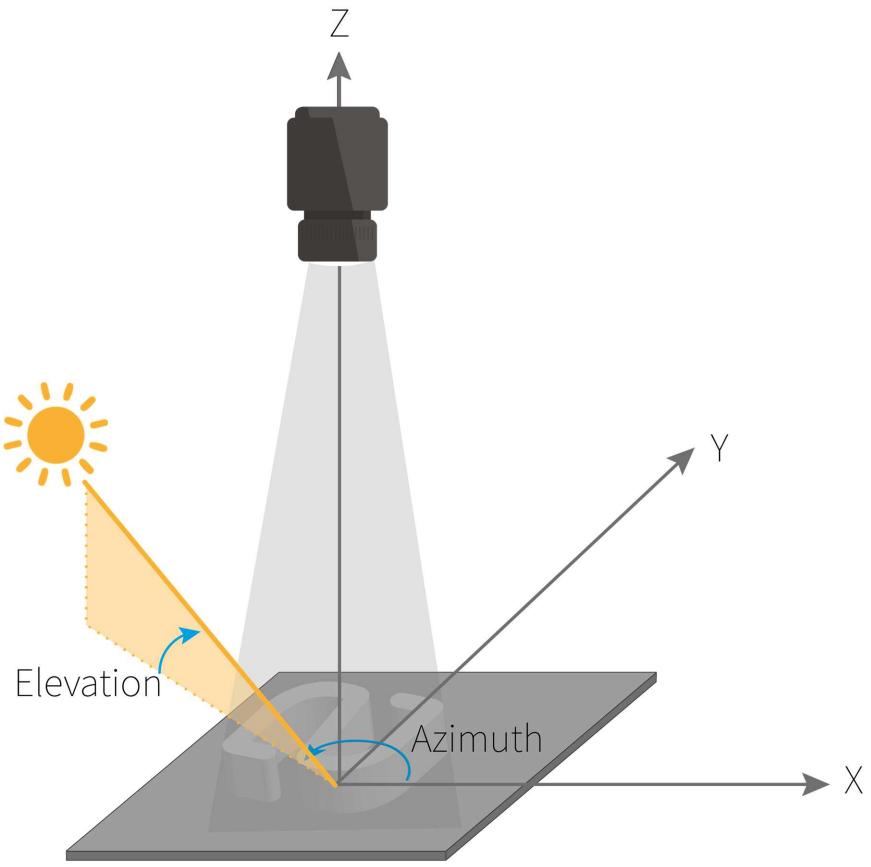
The vector of azimuth angles to set

elevations

The vector of elevation angles to set

Remarks

When facing the image, the axis x points right, y points top and z points towards the camera. Azimuth angles are oriented trigonometrically around the z axis. A light source on the right of the image would have an azimuth of 0 degrees. One on top would have an azimuth of 90 degrees. Elevation is the angle formed by the base plane and the light source. A light source on the horizon would have an elevation of 0 degrees. One on the camera would have an elevation of 90 degrees.



4.182. EPlaneCropper Class

A EPlaneCropper object is used to crop some points of a EPointCloud object.
The points to keep are selected according to their positions with respect to a reference plane.
A EPlaneCropper object is characterized by its reference plane and is used to produce an output EPointCloud object from an input EPointCloud object.
The produced point cloud contains a subset of the input point cloud.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Crop	Crops an EPointCloud. An output point cloud is produced from an input point cloud by only keeping the points that satisfy the specified condition. The condition tests the (signed) distance of the points with respect to the reference plane of the cropper.
EPlaneCropper	Creates an EPlaneCropper object using a horizontal plane as a default reference.
GetPlane	Sets/gets the new reference E3DPlane of the EPlaneCropper object.
Load	Loads the EPlaneCropper configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the EPlaneCropper configuration. The given ESerializer must have been created for writing.
SetPlane	Sets/gets the new reference E3DPlane of the EPlaneCropper object.

EPlaneCropper::Crop

Crops an EPointCloud. An output point cloud is produced from an input point cloud by only keeping the points that satisfy the specified condition.
The condition tests the (signed) distance of the points with respect to the reference plane of the cropper.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Crop(
 const EPointCloud& cloudIn,
 EPointCloud& cloudOut,
 Euresys::Open_eVision::Easy3D::EPlaneCropperType type,
 float maxDistance
 )
```

Parameters

cloudIn

The input point cloud.

cloudOut

The output point cloud.

type

An enum of type EPlaneCropperType that specifies which points of the input point cloud will be copied to the output point cloud.

maxDistance

Specifies the distance from the plane for the types "EPlaneCropperType_KeepClose" and "EPlaneCropperType_KeepFar".
It should be 0 for the types "EPlaneCropperType_KeepAbove" and "EPlaneCropperType_KeepBelow.

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

EPlaneCropper::EPlaneCropper

Creates an EPlaneCropper object using a horizontal plane as a default reference.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EPlaneCropper(
 )

void EPlaneCropper(
 const E3DPlane& plane
 )

void EPlaneCropper(
 const EPlaneCropper& other
 )
```

Parameters

plane

Reference E3DPlane used for the initialization.

other

Reference EPlaneCropper used for the initialization.

EPlaneCropper::Load

Loads the EPlaneCropper configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPlaneCropper::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPlaneCropper& operator=(
 const EPlaneCropper& other
 )
```

Parameters

other

An other EPlaneCropper.

EPlaneCropper::GetPlane

EPlaneCropper::SetPlane

Sets/gets the new reference E3DPlane of the EPlaneCropper object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane GetPlane() const
void SetPlane(const E3DPlane& plane)
```

EPlaneCropper::Save

Saves the EPlaneCropper configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.183. EPlaneFinder Class

A EPlaneFinder object is used to search an E3DPlane in an EPointCloud.
The algorithm searches **the largest plane** in terms of number of "inliers". A point is an "inlier" when its distance to the plane is smaller than a specified threshold (parameter "maximum distance").
Another parameter specifies the expected ratio of inliers over the total number of points in the point cloud (by default, this is set to 0.3).
For more control, you can also set the expected ratio of inliers as a range, when a single number is given the behavior is equivalent to a range (number/2, number).
The method Find throws an error if it cannot achieve a proportion of inliers better than the min of the range. It stops as soon as the max of the range is achieved.
Reducing the size of the range increases speed.
A decimation is applied by default to accelerate the search.
Furthermore, the expected normal to the plane and/or up to two points contained in the plane may be specified.
The method EPlaneFinder::Find processes an EPointCloud object and returns an E3DPlane object when a sufficiently good plane is found in the input point cloud.
The returned plane is the result of fitting an EPlaneFitter on the inliers.

Namespace: Euresys::Open_eVision::Easy3D

Methods

DisableDecimator	Disables the default decimation. The decimation should be disabled when the input point cloud is already decimated.

EnableDecimator	Enables the default decimation which reduces the input point cloud by drawing points randomly. This decimation is enabled by default. The decimation accelerates the search.
EPlaneFinder	Creates an EPlaneFinder object.
Find	Searches the biggest plane in the supplied EPointCloud. If a sufficiently good plane is found, a E3DPlane object is returned. Otherwise an exception is thrown.
GetExpectedCloudInliersRatio	Sets/gets the expected ratio of inliers in the EPointCloud.
GetExpectedCloudInliersRatioRange	Sets/gets the expected ratio of inliers in the EPointCloud.
GetMaxDeviation	Sets/gets the maximum distance of an inlier to the plane.
GetNormal	Returns the expected normal direction (that has been set by EPlaneFinder::SetNormal)
GetNormalTolerance	Returns the angle tolerance around the expected normal (that has been set by EPlaneFinder::SetNormal)
GetNumberOfPointsAfterDecimation	Sets/gets the number of points surviving the decimation. Using the setter enables the decimation if it wasn't already. if numberOfPointsAfterDecimation is bigger than the point cloud size, no decimation occurs.
GetNumberOfPointsSet	Returns the number of points set (this will be either 0, 1 or 2).
GetPoint	Returns an E3DPoint the plane is constrained to contain (that has been set by EPlaneFinder or EPlaneFinder::SetTwoPoints)
GetSeed	Set seed to a custom value, by default, the random seed used is randomly determined.
IsDecimatorEnabled	Returns true if the default decimator is enabled (enabled by default).
IsNormalSet	Returns true if the expected normal direction has been set (not set by default).
IsSeedSet	Returns true if the seed was set to a custom value.
Load	Loads the plane finder configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the plane finder configuration. The given ESerializer must have been created for writing.
SetExpectedCloudInliersRatio	Sets/gets the expected ratio of inliers in the EPointCloud.
SetExpectedCloudInliersRatioRange	Sets/gets the expected ratio of inliers in the EPointCloud.
SetMaxDeviation	Sets/gets the maximum distance of an inlier to the plane.

SetNormal	Sets the expected normal direction and the tolerance around it. These values are used to limit the scope of the plane search. If the angular tolerance is not specified, a default value of 5 degrees is assumed. If the tolerance is specified, the value should be strictly positive and correspond to less than 90 degrees.
SetNumberOfPointsAfterDecimation	Sets/gets the number of points surviving the decimation. Using the setter enables the decimation if it wasn't already. if numberOfPointsAfterDecimation is bigger than the point cloud size, no decimation occurs.
SetOnePoint	Sets one point that the plane will be constrained to contain.
SetSeed	Set seed to a custom value, by default, the random seed used is randomly determined.
SetTwoPoints	Sets two points that the plane will be constrained to contain.
UnsetNormal	Unsets the normal vector definition.
UnsetPoints	Unsets the 1 or 2 points that the plane is constrained to contain.
UnsetSeed	Reset seed to its default state of being randomly initialized.

EPlaneFinder::DisableDecimator

Disables the default decimation.
The decimation should be disabled when the input point cloud is already decimated.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void DisableDecimator(
 )
```

EPlaneFinder::EnableDecimator

Enables the default decimation which reduces the input point cloud by drawing points randomly.
This decimation is enabled by default. The decimation accelerates the search.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EnableDecimator(
 )
```

EPlaneFinder::EPlaneFinder

Creates an EPlaneFinder object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EPlaneFinder(
 float maxDeviation,
 float pcExpectedInCloud
 )
void EPlaneFinder(
 float maxDeviation,
 const EFloatRange& pcExpectedInCloudRange
 )
void EPlaneFinder(
 const EPlaneFinder& other
 )
```

Parameters

maxDeviation

Maximum distance of an inlier to the plane. This value has to be strictly positive.

pcExpectedInCloud

This is an estimation of the ratio of inliers in the point cloud.
This optional parameter has a default value of 0.3.
The algorithm stops as soon as a pointCloud with at least pcExpectedInCloud inliers is found.
It throws an error if the best plane has less than pcExpectedInCloud/2 inliers.

pcExpectedInCloudRange

This is an estimation of the ratio of inliers in the point cloud as a range within]0, 1[.
The algorithm throws an error if the best plane has less than inliersProportion.min inliers.
The algorithm stops as soon as a model with inliersProportion.max is found.
Increasing inliersProportion.min can increase speed but at the risk of missing the plane.
Decreasing the max of the range can increase speed but at the risk of finding a bad plane.

other

The EPlaneFinder object that should be copied.

EPlaneFinder::GetExpectedCloudInliersRatio

EPlaneFinder::SetExpectedCloudInliersRatio

Sets/gets the expected ratio of inliers in the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetExpectedCloudInliersRatio() const

void SetExpectedCloudInliersRatio(float pcExpectedInCloud)
```

Remarks

This setter/getter is used to access the max of the range, its behavior is the same as SetExpectedCloudInliersRatioRange(EFloatRange(pcExpectedInCloud/2, pcExpectedInCloud)) and GetExpectedCloudInliersRatioRange().GetUpperBound()

EPlaneFinder::GetExpectedCloudInliersRatioRange

EPlaneFinder::SetExpectedCloudInliersRatioRange

Sets/gets the expected ratio of inliers in the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EFloatRange GetExpectedCloudInliersRatioRange() const

void SetExpectedCloudInliersRatioRange(const EFloatRange& pcExpectedInCloudRange)
```

Remarks

This is an estimation of the ratio of inliers in the point cloud as a range within]0, 1[.
If pcExpectedInCloudRange.min is too small, we risk missing the plane.
The algorithm throws an error if the best plane has less than pcExpectedInCloudRange.min inliers. The algorithm stops as soon as a model with pcExpectedInCloudRange.max is found.
Increasing pcExpectedInCloudRange.min can increase speed.
Decreasing pcExpectedInCloudRange.max can increase speed.

EPlaneFinder::Find

Searches the biggest plane in the supplied EPointCloud. If a sufficiently good plane is found, a E3DPlane object is returned.
Otherwise an exception is thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane Find(
 const EPointCloud& pointCloud
 )

E3DPlane Find(
 const EPointCloud& pointCloud,
 float& effectiveInliersRatio
 )
```

```
E3DPlane Find(
 const EPointCloud& pointCloud,
 float& effectiveInliersRatio,
 EPointCloud& inliers
 )

E3DPlane Find(
 const EPointCloud& pointCloud,
 float& effectiveInliersRatio,
 EPointCloud& inliers,
 EPointCloud& outliers
 )
```

Parameters

pointCloud

The input point cloud in which the plane should be searched (need at least 3 points)

effectiveInliersRatio

This passed by reference float will contain the effective ratio of inliers

inliers

A pointcloud that will contain the inliers of the plane

outliers

A pointcloud that will contain the outliers of the plane

EPlaneFinder::GetNormal

Returns the expected normal direction (that has been set by EPlaneFinder::SetNormal)

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetNormal(
 )
```

EPlaneFinder::GetPoint

Returns an E3DPoint the plane is constrained to contain (that has been set by EPlaneFinder or EPlaneFinder::SetTwoPoints)

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetPoint(
 bool first
 )
```

Parameters

first

True if you want to get the first point and false if you want the second instead.

EPlaneFinder::IsDecimatorEnabled

Returns true if the default decimator is enabled (enabled by default).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsDecimatorEnabled(
 )
```

EPlaneFinder::IsNormalSet

Returns true if the expected normal direction has been set (not set by default).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsNormalSet(
 )
```

EPlaneFinder::IsSeedSet

Returns true if the seed was set to a custom value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsSeedSet(
 )
```

EPlaneFinder::Load

Loads the plane finder configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPlaneFinder::GetMaxDeviation

EPlaneFinder::SetMaxDeviation

Sets/gets the maximum distance of an inlier to the plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMaxDeviation() const
void SetMaxDeviation(float maxDeviation)
```

EPlaneFinder::GetNormalTolerance

Returns the angle tolerance around the expected normal (that has been set by EPlaneFinder::SetNormal)

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetNormalTolerance() const
```

EPlaneFinder::GetNumberOfPointsAfterDecimation

EPlaneFinder::SetNumberOfPointsAfterDecimation

Sets/gets the number of points surviving the decimation. Using the setter enables the decimation if it wasn't already. if numberOfPointsAfterDecimation is bigger than the point cloud size, no decimation occurs.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumberOfPointsAfterDecimation() const
void SetNumberOfPointsAfterDecimation(int numberOfPointsAfterDecimation)
```

EPlaneFinder::GetNumberOfPointsSet

Returns the number of points set (this will be either 0, 1 or 2).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumberOfPointsSet() const
```

EPlaneFinder::SetOnePoint

Sets one point that the plane will be constrained to contain.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetOnePoint(const E3DPoint& point1)
```

Remarks

To set 2 points, use the function SetTwoPoints.
If two points were previously set, the second one will be removed by this function.

EPlaneFinder::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPlaneFinder& operator=(
 const EPlaneFinder& other
 )
```

Parameters

other

The EPlaneFinder object that should be copied.

EPlaneFinder::Save

Saves the plane finder configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EPlaneFinder::GetSeed

EPlaneFinder::SetSeed

Set seed to a custom value, by default, the random seed used is randomly determined.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
OEV_UINT32 GetSeed() const

void SetSeed(OEV_UINT32 seed)
```

EPlaneFinder::SetNormal

Sets the expected normal direction and the tolerance around it. These values are used to limit the scope of the plane search.
If the angular tolerance is not specified, a default value of 5 degrees is assumed.
If the tolerance is specified, the value should be strictly positive and correspond to less than 90 degrees.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetNormal(
 const E3DPoint& normal
 )

void SetNormal(
 const E3DPoint& normal,
 float angleTolerance
 )

void SetNormal(
 float nx,
 float ny,
 float nz
 )

void SetNormal(
 float nx,
 float ny,
 float nz,
 float angleTolerance
 )

void SetNormal(
 const E3DPlane& referencePlane
 )

void SetNormal(
 const E3DPlane& referencePlane,
 float angleTolerance
 )
```

Parameters

normal

The normal vector specifies the expected perpendicular direction of the plane.

angleTolerance

The angle tolerance is the maximum angular deviation around the expected normal (strictly positive and smaller than 90 degrees). It's set to 5 degrees by default.

nx

The x component of the normal vector.

ny

The y component of the normal vector.

nz

The z component of the normal vector.

referencePlane

The reference plane specifies the expected perpendicular direction.

EPlaneFinder::SetTwoPoints

Sets two points that the plane will be constrained to contain.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetTwoPoints(
 const E3DPoint& point1,
 const E3DPoint& point2
 )
```

Parameters

point1

The first point.

point2

The second point.

Remarks

To set 1 point, use the function SetOnePoint

EPlaneFinder::UnsetNormal

Unsets the normal vector definition.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetNormal(
 )
```

EPlaneFinder::UnsetPoints

Unsets the 1 or 2 points that the plane is constrained to contain.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetPoints(
 )
```

EPlaneFinder::UnsetSeed

Reset seed to its default state of being randomly initialized.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetSeed(
 )
```

4.184. EPlaneFitter Class

A EPlaneFitter object is used to fit an E3DPlane on an EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

Methods

EPlaneFitter	Constructor of an EPlaneFitter object.
Fit	Fits an E3DPlane on a given EPointCloud.
GetMinSampleCount	Sets/Gets the minimum number of samples required for fitting on each side of the shape. By default, a value of 3 is assumed.
Load	Loads the EPlaneFitter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the EPlaneFitter configuration. The given ESerializer must have been created for writing.
SetMinSampleCount	Sets/Gets the minimum number of samples required for fitting on each side of the shape. By default, a value of 3 is assumed.

EPlaneFitter::EPlaneFitter

Constructor of an EPlaneFitter object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EPlaneFitter(
 )
void EPlaneFitter(
 const EPlaneFitter& other
 )
```

Parameters

other

Reference to the EPlaneFitter used for the initialization.

EPlaneFitter::Fit

Fits an E3DPlane on a given EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPlane Fit(
 const EPointCloud& pc
 )
E3DPlane Fit(
 const EPointCloud& pc,
 float& averageDistance
 )
```

Parameters

pc

The reference to the point cloud.

averageDistance

The reference to a float which will store the average distance from this plane to the points that were used for the fit.

EPlaneFitter::Load

Loads the EPlaneFitter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPlaneFitter::GetMinSampleCount

EPlaneFitter::SetMinSampleCount

Sets/Gets the minimum number of samples required for fitting on each side of the shape.
By default, a value of 3 is assumed.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetMinSampleCount() const
void SetMinSampleCount(int minSamples)
```

EPlaneFitter::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPlaneFitter& operator=(
 const EPlaneFitter& other
 )
```

Parameters

other

The EPlaneFitter object that should be copied.

EPlaneFitter::Save

Saves the EPlaneFitter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.185. EPoint Class

An exact (floating-point) location in the 2D space.

Derived Class(es):EFrame

Namespace: Euresys::Open_eVision

Methods

Area	Compute the oriented area of the parallelogram built on two EPoint.
Argument	Compute the polar argument of a EPoint object.
CopyTo	Copies all the data of the current EPoint object into another EPoint object and returns it.
Cross	Compute the cross product of two EPoint object.
Distance	Returns the distance between the addressed point and an EPoint object.
Dot	Compute the dot product of two EPoint object.
EPoint	Constructs a EPoint object.
GetCenter	Center coordinates of a EPoint object.
GetX	Abscissa (X coordinate) of the EPoint object
GetY	Ordinate (Y coordinate) of the EPoint object
Load	Load the EPoint configuration. The given ESerializer must have been created for reading.

MidPoint	Returns the middle coordinate between this EPoint object and another EPoint object.
Modulus	Compute the euclidean modulus of a EPoint.
operator-	Subtracts from the current EPoint center coordinates the center coordinates of another EPoint object.
operator!=	Compares the current EPoint center coordinates with the center coordinates of another EPoint object.
operator*	Multiplies the current EPoint center coordinates by a given multiplier.
operator/	Divides the current EPoint center coordinates by a given divisor.
operator+	Adds to the current EPoint center coordinates the center coordinates of another EPoint object.
operator=	Copies all the data from another EPoint object into the current EPoint object.
operator==	Compares the current EPoint center coordinates with the center coordinates of another EPoint object.
Project	Compute the orthogonal projection of a EPoint on another shape.
Rotate	Returns another EPoint object containing the coordinated of the rotated point.
Save	Save the EPoint configuration. The given ESerializer must have been created for writing.
SetCenter	Center coordinates of a EPoint object.
SetCenterXY	Sets the center coordinates of a EPoint object.
Square	Compute the sum of the squared coordinates of a EPoint.
SquaredDistance	Compute the squared distance between two EPoint.

EPoint::Area

Compute the oriented area of the parallelogram built on two EPoint.

Namespace: Euresys::Open_eVision

[C++]

```
float Area(
 const EPoint& Point
 )
```

Parameters

Point

Second edge of the parallelogram.

Remarks

Compute the oriented area of the parallelogram built on two EPoint. The area is counted as positive if the oriented vector pair ('first edge', 'second edge') is in the same sense that the axis frame. This oriented area can also be viewed as the z-coordinate of a vector product of two 3D vectors obtained in supplementing each edges with a third z-coordinate (set to zero).

EPoint::Argument

Compute the polar argument of a EPoint object.

Namespace: Euresys::Open_eVision

```
[C++]
float Argument(
 )
```

Remarks

Compute the angle (in radians) between the oriented X-axis and the vector going from the axis origin and the EPoint. If the axis frame is orthogonal, this number is also the polar argument of the EPoint.

EPoint::GetCenter

EPoint::SetCenter

Center coordinates of a EPoint object.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

EPoint::CopyTo

Copies all the data of the current EPoint object into another EPoint object and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EPoint& other
 )

EPoint* CopyTo(
 EPoint* other
 )
```

Parameters

other

Pointer to the EPoint object in which the current EPoint object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EPoint object will be created and returned.

EPoint::Cross

Compute the cross product of two EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
float Cross(
 const EPoint& Point
 )
```

Parameters

Point

Second factor of the cross product.

EPoint::Distance

Returns the distance between the addressed point and an EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
float Distance(
 const EPoint& point
 )

float Distance(
 const ELine& line,
 bool segmentOnly
 )
```

```
float Distance(
 const ECircle& circle,
 bool arcOnly
 )
```

Parameters

point

EPoint object with which to calculate the distance.

line

ELine object with which to calculate the distance.

segmentOnly

By default (false), the line is not restricted to a segment.

circle

ECircle object with which to calculate the distance.

arcOnly

By default (false), the circle is not restricted to an arc.

Remarks

Many EasyGauge members provide measurement result as a EPoint object (see EPointGauge::Center, EPointGauge::GetMeasuredPoint,...). The EPoint class has its own members to retrieve all the information pertaining to a point. Among them, the Distance method returns the distance between a point pair or between a point and a line segment, a circle arc or a rectangle.

EPoint::Dot

Compute the dot product of two EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
float Dot(
 const EPoint& Point
 )
```

Parameters

Point

Second factor of the dot product.

EPoint::EPoint

Constructs a EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
void EPoint(
 )
void EPoint(
 float centerX,
 float centerY
 )
void EPoint(
 const EPoint& other
 )
```

Parameters

centerX

Center coordinates of the EPoint object.

centerY

Center coordinates of the EPoint object.

other

Another EPoint object to be copied in the new EPoint object.

EPoint::Load

Load the EPoint configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPoint::MidPoint

Returns the middle coordinate between this EPoint object and another EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint MidPoint(
 const EPoint& Point
 )
```

Parameters

Point

The other EPoint object.

EPoint::Modulus

Compute the euclidean modulus of a EPoint.

Namespace: Euresys::Open_eVision

[C++]

```
float Modulus(
 )
```

Remarks

Compute the squared root of the sum of the squared coordinates of an EPoint. If the axis frame is orthogonal, this number is also the euclidean norm of the EPoint.

EPoint::operator-

Subtracts from the current EPoint center coordinates the center coordinates of another EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint operator-(
 const EPoint& point
 )
```

Parameters

point

The other EPoint object.

EPoint::operator!=

Compares the current EPoint center coordinates with the center coordinates of another EPoint object.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const EPoint& point
 )
```

Parameters

point

The other EPoint object.

Remarks

Returns true if EPoint::X or EPoint::Y are respectively different.

EPoint::operator*

Multiplies the current EPoint center coordinates by a given multiplier.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint operator*(
 float scalar
 )
```

Parameters

scalar

The multiplier.

EPoint::operator/

Divides the current EPoint center coordinates by a given divisor.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint operator/(
 float scalar
 )
```

Parameters

scalar

The divisor.

EPoint::operator+

Adds to the current EPoint center coordinates the center coordinates of another EPoint object.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint operator+(
 const EPoint& point
 )
```

Parameters

point

The other EPoint object.

EPoint::operator=

Copies all the data from another EPoint object into the current EPoint object.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint& operator=(
 const EPoint& other
 )
```

Parameters

other

EPoint object to be copied.

EPoint::operator==

Compares the current EPoint center coordinates with the center coordinates of another EPoint object.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EPoint& point
 )
```

Parameters

point

The other EPoint object.

Remarks

Returns true if both EPoint::X and EPoint::Y are respectively the same.

EPoint::Project

Compute the orthogonal projection of a EPoint on another shape.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint Project(
 const ELine& shape
 )
EPoint Project(
 const ECircle& shape
 )
```

Parameters

shape

Shape object to which point is projected

Remarks

Compute the orthogonal projection of a EPoint on another shape. This computation is only valid when the axis frame is orthogonal.

EPoint::Rotate

Returns another EPoint object containing the coordinated of the rotated point.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint Rotate(
 float angle
 )
```

Parameters

angle

Rotation angle (in radians)

Remarks

Rotates a EPoint around the origin (0, 0) by an angle of angle radians. By definition, the smallest (in absolute value) rotation of the oriented X-Axis toward the oriented Y-Axis is chosen as the positive sense of rotation. In a direct frame, this is also the trigonometric sense (counter clockwise).

EPoint::Save

Save the EPoint configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPoint::SetCenterXY

Sets the center coordinates of a EPoint object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the EPoint object.

centerY

Center coordinates of the EPoint object.

EPoint::Square

Compute the sum of the squared coordinates of a EPoint.

Namespace: Euresys::Open_eVision

[C++]

```
float Square(
 )
```

Remarks

Compute the sum of the squared coordinates of a EPoint. If the axis frame is orthogonal, this sum of squares is also the squared euclidean norm of the EPoint object.

EPoint::SquaredDistance

Compute the squared distance between two EPoint.

Namespace: Euresys::Open_eVision

```
[C++]
float SquaredDistance(
 const EPoint& Point
 )
```

Parameters

Point

Second EPoint.

Remarks

Compute the sum of squared coordinates differences of two EPoint. If the axis frame is orthogonal, this number is also the squared euclidean distance between two EPoint.

EPoint::GetX

Abscissa (X coordinate) of the EPoint object

Namespace: Euresys::Open_eVision

```
[C++]
float GetX() const
```

EPoint::GetY

Ordinate (Y coordinate) of the EPoint object

Namespace: Euresys::Open_eVision

```
[C++]
float GetY() const
```

4.186. EPointCloud Class

Represents a 3D point cloud.

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddCustomAttributeBuffer	Allocates and copies the data to the new custom attribute buffer.
AddPoint	Adds a point to the EPointCloud.
AddPointAndAttributesTo	Adds a point and its attributes to another EPointCloud.
AddPointCloud	Adds the points of another point cloud to EPointCloud.
AddPoints	Adds a vector of points to the EPointCloud.
AllocateAttributeBuffer	Allocates an attribute buffer and fills it with defaultValue if the cloud is not empty.
AllocateCustomAttributeBuffer	Allocates the data to the new custom attribute buffer and fills it with default values.
Clear	Empties the EPointCloud.
ClearAttributeBuffer	Empties the attribute buffer.
ComputeNormalsAndCurvatures	Compute the normals of the cloud's points by fitting a plane on the 10 neighbors points. This method does not allow to compute the sign of the normals (we can't determine if a normal is x,y,z or -x,-y,-z). The curvatures of the points will be computed when the computeCurvatures argument is set to true.
ComputePlaneBehind	Returns the E3DPlane with given normal on which the EPointCloud lays. This is useful when projecting an EPointCloud on an EZMap.
CopyAllAttributesTo	Copies all attributes at a given index from one EPointCloud to another one.
DistanceToLine	Returns the shortest distance between an E3DPoint of the EPointCloud and a line represented by 2 E3DPoint.
DistanceToSegment	Returns the shortest distance between an E3DPoint of the EPointCloud and a segment represented by 2 E3DPoint.
EPointCloud	Creates an EPointCloud object.
FillAttributeBuffer	Allocates and copies the data to the attribute buffer.
FillPointsBuffer	Copies an external points buffer into the internal points buffer.
GetAttribute	Retrieve the value of an attribute.
GetAttributeBuffer	Retrieves a pointer to the internal attribute buffer.
GetAttributeBufferType	Returns the EAttributeType corresponding to the E3DAttribute.
GetEnableSpacePartition	Sets/Gets the status of space partitioning when computing distance to Segments and Lines. Space partitioning is a way to speed up the geometric queries on EPointCloud. The affected methods are EPointCloud::DistanceToSegment, EPointCloud::DistanceToLine. By default, space partition is disabled for these two methods.
GetInitializedAttributes	Fills the vector with the ids (in growing order) of all the attributes that have been initialized.

GetNumPoints	Number of points in the EPointCloud.
GetPoint	Retrieves a point from the EPointCloud.
GetPointsBuffer	Retrieves a pointer to the internal points buffer.
HasAttributeBuffer	Returns true if a buffer of the given E3DAttribute exists in the point cloud
Load	Loads the EPointCloud. Supported formats are Open eVision proprietary, CSV, OBJ, PCD, PLY and XYZ. The given ESerializer must have been created for reading.
LoadCSV	Loads an EPointCloud stored in the CSV (Comma-Separated Values) file format.
LoadOBJ	Loads an EPointCloud stored in the OBJ file format.
LoadPCD	Loads an EPointCloud stored in the PCD (Point Cloud Library) file format. ASCII and binary formats are compatible.
LoadPLY	Loads an EPointCloud stored in the PLY file format.
LoadXYZ	Loads an EPointCloud stored in the XYZ file format.
operator=	Assignment operator.
PrepareSpacePartition	Enables the build of the space partition (like EPointCloud) and builds it now instead of when it is first needed. The computation time of the space partition depends on the number of points in the point cloud.
RemovePoint	Removes a point and its corresponding attributes from the EPointCloud.
Save	Saves the EPointCloud. Supported formats are Open eVision proprietary, CSV, OBJ, PCD, PLY and XYZ. The given ESerializer must have been created for writing.
SaveCSV	Saves the EPointCloud in the CSV (Comma-Separated Values) file format.
SaveOBJ	Saves the EPointCloud in the OBJ file format.
SavePCD	Saves the EPointCloud in the PCD (Point Cloud Library) file format. ASCII and binary formats are available.
SavePLY	Saves the EPointCloud in the PLY file format.
SaveXYZ	Saves the EPointCloud in the XYZ file format.
SetAttribute	Sets the corresponding attribute at the given index.
SetEnableSpacePartition	Sets/Gets the status of space partitioning when computing distance to Segments and Lines. Space partitioning is a way to speed up the geometric queries on EPointCloud. The affected methods are EPointCloud::DistanceToSegment, EPointCloud::DistanceToLine. By default, space partition is disabled for these two methods.

EPointCloud::AddCustomAttributeBuffer

Allocates and copies the data to the new custom attribute buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int AddCustomAttributeBuffer(
 const std::vector<OEV_UINT8>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<OEV_UINT8>& data,
 OEV_UINT8 defaultValue
 )

int AddCustomAttributeBuffer(
 const std::vector<OEV_UINT16>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<OEV_UINT16>& data,
 OEV_UINT16 defaultValue
 )

int AddCustomAttributeBuffer(
 const std::vector<OEV_UINT32>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<OEV_UINT32>& data,
 OEV_UINT32 defaultValue
 )

int AddCustomAttributeBuffer(
 const std::vector<int>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<int>& data,
 int defaultValue
 )

int AddCustomAttributeBuffer(
 const std::vector<float>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<float>& data,
 float defaultValue
 )

int AddCustomAttributeBuffer(
 const std::vector<double>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<double>& data,
 double defaultValue
 )
```

```
int AddCustomAttributeBuffer(
 const std::vector<Euresys::Open_eVision::EC24A>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<Euresys::Open_eVision::EC24A>& data,
 const EC24A& defaultValue
 )

int AddCustomAttributeBuffer(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& data
 )

int AddCustomAttributeBuffer(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& data,
 const E3DPoint& defaultValue
 )
```

Parameters

data

The address of the external attribute buffer. The buffer should be of the same length as the EPointCloud (see EPointCloud::NumPoints).

defaultValue

The value used to set the attribute when the EPointCloud grows (see EPointCloud::AddPoint, EPointCloud::AddPoints and EPointCloud::AddPointCloud).

Remarks

An exception is also thrown if data is nullptr, if you don't want to specify the data, use EPointCloud::AllocateCustomAttributeBuffer.

EPointCloud::AddPoint

Adds a point to the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddPoint(
 const E3DPoint& point
 )
```

Parameters

point

The E3DPoint to add to the point cloud.

EPointCloud::AddPointAndAttributesTo

Adds a point and its attributes to another EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddPointAndAttributesTo(
 EPointCloud& dstCloud,
 OEV_UINT32 srcIndex,
 bool addAttributeIfNotPresent
 )
```

Parameters

dstCloud

The EPointCloud where the attributes are copied.

srcIndex

The index of the element to copy from the source EPointCloud.

addAttributeIfNotPresent

Whether to initialize a buffer if it is present in EPointCloud but not in dstCloud. this could result in the creation of an attribute filled with default values.

EPointCloud::AddPointCloud

Adds the points of another point cloud to EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddPointCloud(
 const EPointCloud& cloud,
 bool addAttributeIfNotPresent
 )
```

Parameters

cloud

Point cloud whose points will be added to the point cloud.

addAttributeIfNotPresent

Whether to initialize a buffer if it is present cloud but not in EPointCloud. this could result in the creation of an attribute filled with default values.

EPointCloud::AddPoints

Adds a vector of points to the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddPoints(
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& points
 )
```

Parameters

points

Vector of E3DPoint to add to the point cloud.

EPointCloud::AllocateAttributeBuffer

Allocates an attribute buffer and fills it with defaultValue if the cloud is not empty.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AllocateAttributeBuffer(
 int attribute,
 OEV_UINT8 defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 OEV_UINT16 defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 OEV_UINT32 defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 int defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 float defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 double defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 const EC24A& defaultValue
 )

void AllocateAttributeBuffer(
 int attribute,
 const E3DPoint& defaultValue
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

defaultValue

The value used to set the attribute when the EPointCloud grows (see EPointCloud::AddPoint, EPointCloud::AddPoints and EPointCloud::AddPointCloud).

Remarks

If the data type is not correct depending on the E3DAttribute, it throws an EException with an EError.

EPointCloud::AllocateCustomAttributeBuffer

Allocates the data to the new custom attribute buffer and fills it with default values.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int AllocateCustomAttributeBuffer(
 OEV_UINT8 defaultValue
 )

int AllocateCustomAttributeBuffer(
 OEV_UINT16 defaultValue
 )

int AllocateCustomAttributeBuffer(
 OEV_UINT32 defaultValue
 )

int AllocateCustomAttributeBuffer(
 int defaultValue
 )

int AllocateCustomAttributeBuffer(
 float defaultValue
 )

int AllocateCustomAttributeBuffer(
 double defaultValue
 )

int AllocateCustomAttributeBuffer(
 const EC24A& defaultValue
 )

int AllocateCustomAttributeBuffer(
 const E3DPoint& defaultValue
 )
```

Parameters

defaultValue

The value used to set the attribute initially and when the EPointCloud grows (see EPointCloud::AddPoint, EPointCloud::AddPoints and EPointCloud::AddPointCloud).

EPointCloud::Clear

Empties the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Clear(
 )
```

EPointCloud::ClearAttributeBuffer

Empties the attribute buffer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearAttributeBuffer(
 int attribute
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

Remarks

If the attribute buffer was a custom attribute, the id is not valid after the call to the method as the buffer will have been deallocated.

EPointCloud::ComputeNormalsAndCurvatures

Compute the normals of the cloud's points by fitting a plane on the 10 neighbors points. This method does not allow to compute the sign of the normals (we can't determine if a normal is x,y,z or -x,-y,-z).
The curvatures of the points will be computed when the computeCurvatures argument is set to true.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ComputeNormalsAndCurvatures(
 bool computeCurvatures,
 OEV_UINT32 nbAdditionalNeighborsForCurvatures
 )
```

Parameters

computeCurvatures

Set to true to compute the local curvatures (numbers between 0 for a plane and 1/3 for random noise) of each point as well.

nbAdditionalNeighborsForCurvatures

Each curvature is computed using 10 + nbAdditionalNeighborsForCurvatures points.

Remarks

The normals are stored in the E3DAttribute_Normal attribute buffer and the curvatures in the E3DAttribute_Curvature attribute buffer (if requested).

EPointCloud::ComputePlaneBehind

Returns the E3DPlane with given normal on which the EPointCloud lays. This is useful when projecting an EPointCloud on an EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane ComputePlaneBehind(
 E3DPoint normal,
 float margin
 )
```

Parameters

normal

The normal of the plane.

margin

Let P be the plane, A point of the EPointCloud closest to P and B the of the EPointCloud furthest to B. margin = dist(P, A) / dist(P, B). Must be positive. Default: 0.02.

EPointCloud::CopyAllAttributesTo

Copies all attributes at a given index from one EPointCloud to another one.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void CopyAllAttributesTo(
 EPointCloud& dstCloud,
 OEV_UINT32 dstIndex,
 OEV_UINT32 srcIndex
 )
```

Parameters

dstCloud

The EPointCloud where the attributes are copied.

dstIndex

The index of the destination EPointCloud.

srcIndex

The index of the source EPointCloud.

Remarks

An attribute is copied only if both attribute buffers are initialized. If both attribute types are not the same, then a conversion is done (if possible). This function does not perform an allocation. To add a new point with all its attributes to another EPointCloud, use function EPointCloud::AddPointAndAttributesTo.

EPointCloud::DistanceToLine

Returns the shortest distance between an E3DPoint of the EPointCloud and a line represented by 2 E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float DistanceToLine(
 const E3DPoint& origin,
 const E3DPoint& end
 )

float DistanceToLine(
 const E3DPoint& origin,
 const E3DPoint& end,
 E3DPoint& closest,
 int& closestIndex
 )
```

Parameters

origin
One point of the line.

end
The other point of the line.

closest
Where the closest point will be stored.

closestIndex
Where the index of the closest point will be stored.

EPointCloud::DistanceToSegment

Returns the shortest distance between an E3DPoint of the EPointCloud and a segment represented by 2 E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float DistanceToSegment(
 const E3DPoint& origin,
 const E3DPoint& end
 )
float DistanceToSegment(
 const E3DPoint& origin,
 const E3DPoint& end,
 E3DPoint& closest,
 int& closestIndex
 )
```

Parameters

origin
One endpoint of the segment.

end
The other endpoint of the segment.

closest
Where the closest point will be stored.

closestIndex
Where the index of the closest point will be stored.

EPointCloud::GetEnableSpacePartition

EPointCloud::SetEnableSpacePartition

Sets/Gets the status of space partitioning when computing distance to Segments and Lines. Space partitioning is a way to speed up the geometric queries on EPointCloud. The affected methods are EPointCloud::DistanceToSegment, EPointCloud::DistanceToLine. By default, space partition is disabled for these two methods.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetEnableSpacePartition() const
void SetEnableSpacePartition(bool state)
```

Remarks

See also EPointCloud::PrepareSpacePartition. For EPointCloud, the same space partition is used but it cannot be disabled as using it is always faster.

EPointCloud::EPointCloud

Creates an EPointCloud object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EPointCloud(
 )
void EPointCloud(
 const EPointCloud& other
 )
```

Parameters

other

Reference to the EPointCloud used for the initialization.

EPointCloud::FillAttributeBuffer

Allocates and copies the data to the attribute buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillAttributeBuffer(
 int attribute,
 const std::vector<OEV_UINT8>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<OEV_UINT8>& data,
 OEV_UINT8 defaultValue
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<OEV_UINT16>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<OEV_UINT16>& data,
 OEV_UINT16 defaultValue
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<OEV_UINT32>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<OEV_UINT32>& data,
 OEV_UINT32 defaultValue
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<int>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<int>& data,
 int defaultValue
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<float>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<float>& data,
 float defaultValue
 )
```

```
void FillAttributeBuffer(
 int attribute,
 const std::vector<double>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<double>& data,
 double defaultValue
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<Euresys::Open_eVision::EC24A>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<Euresys::Open_eVision::EC24A>& data,
 const EC24A& defaultValue
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& data
 )

void FillAttributeBuffer(
 int attribute,
 const std::vector<Euresys::Open_eVision::Easy3D::E3DPoint>& data,
 const E3DPoint& defaultValue
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

data

The address of the external attribute buffer. The buffer should be of the same length as the EPointCloud (see EPointCloud::NumPoints).

defaultValue

The value used to set the attribute when the EPointCloud grows (see EPointCloud::AddPoint, EPointCloud::AddPoints and EPointCloud::AddPointCloud).

Remarks

If the data type is not correct depending on the E3DAttribute, it throws an EException with an EError. An exception is also thrown if data is nullptr, if you don't want to specify the data, use EPointCloud::AllocateAttributeBuffer.

EPointCloud::FillPointsBuffer

Copies an external points buffer into the internal points buffer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillPointsBuffer(
 void* pointsBuffer,
 int numPoints
 )
```

Parameters

pointsBuffer

Address of the external points buffer.

numPoints

Number of points in the external points buffer.

Remarks

The buffer must contain points in the form of triplets of 32bits floats stored in the (X,Y,Z) order.

EPointCloud::GetAttribute

Retrieve the value of an attribute.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 OEV_UINT8& value
 )

void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 OEV_UINT16& value
 )

void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 OEV_UINT32& value
 )

void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 int& value
 )
```

```
void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 float& value
 )

void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 double& value
 )

void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 EC24A& value
 )

void GetAttribute(
 int attribute,
 OEV_UINT32 index,
 E3DPoint& value
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

index

The index of the element.

value

Where the attribute value will be stored.

Remarks

If the attribute has not been initialized or if the index is out of bound, it throws an EException.

EPointCloud::GetAttributeBuffer

Retrieves a pointer to the internal attribute buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const void* GetAttributeBuffer(
 int attribute
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

Remarks

If the attribute has not been initialized, it throws an EException.

EPointCloud::GetAttributeBufferType

Returns the EAttributeType corresponding to the E3DAttribute.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EAttributeType GetAttributeBufferType(
 int attribute
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

EPointCloud::GetInitializedAttributes

Fills the vector with the ids (in growing order) of all the attributes that have been initialized.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetInitializedAttributes(
 std::vector<int>& attributes
 )
```

Parameters

attributes

The vector that will contain the initialized attributes id.

EPointCloud::GetPoint

Retrieves a point from the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetPoint(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the point to be retrieved.

EPointCloud::HasAttributeBuffer

Returns true if a buffer of the given E3DAttribute exists in the point cloud

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool HasAttributeBuffer(
 int attribute
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

EPointCloud::Load

Loads the EPointCloud. Supported formats are Open eVision proprietary, CSV, OBJ, PCD, PLY and XYZ. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is read from.

EPointCloud::LoadCSV

Loads an EPointCloud stored in the CSV (Comma-Separated Values) file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadCSV(
 const std::string& path
 )

void LoadCSV(
 const std::string& path,
 const EFloatRange& undefinedZValues
 )
```

Parameters

path

The full path of the input file.

undefinedZValues

Optional parameter, discard the points with Z value in the given range.

Remarks

Only the x,y,z coordinates of the points of the pointcloud are loaded. Use pcd or ply if you need more.
There is no standard for pointcloud in CSV file format. To be correctly read, the file needs to have at least the components (x, y, z) and each value must be separated by a comma.
An acceptable file could be:
x, y, z
x1, y1, z1
x2, y2, z2
x3, y3, z3
Other components can be in the file and the order of the elements may differs from the example as long as it is defined in the header. Note that files containing series of profile frames (i.e. only containing z-values) are not supported.
Use EPointCloud::SaveCSV to save to CSV file.

EPointCloud::LoadOBJ

Loads an EPointCloud stored in the OBJ file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadOBJ(
 const std::string& path
 )
```

```
void LoadOBJ(
 const std::string& path,
 const EFloatRange& undefinedZValues
 )
```

Parameters

path

The full path of the input file.

undefinedZValues

Optional parameter, discard the points with Z value in the given range.

Remarks

The OBJ file format is documented here:
https://www.fileformat.info/format/wavefrontobj/egff.htm.
The only attribute loaded in the obj file format is the normals. Use pcd or ply if you need more.
Use EPointCloud::SaveOBJ to save to OBJ file.

EPointCloud::LoadPCD

Loads an EPointCloud stored in the PCD (Point Cloud Library) file format.
ASCII and binary formats are compatible.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadPCD(
 const std::string& path
 )

void LoadPCD(
 const std::string& path,
 const EFloatRange& undefinedZValues
 )
```

Parameters

path

The full path of the input file.

undefinedZValues

Optional parameter, discard the points with Z value in the given range.

Remarks

The PCD file format is documented here:
http://pointclouds.org/documentation/tutorials/pcd_file_format.html.
When loading PCD files:
- columns where the field COUNT is not set to 1 are ignored
- in ascii files, all numbers are assumed to be written in base 10
- file is assumed to contain at least the floating point fields x, y and z. They mustn't be the first but have to be consecutive. If one of these points is nan, inf or too big to be represented on a float, the line is ignored. These points can be double but are stored internally as float.
- the other columns are loaded as user-custom attributes unless they match the specifications of our particular attributes
- a column representing a color must have as suffix "_C" and be of type uint32 or float, they are bitwise encoded as a combination of uint8 in the form argb (uint32) or _rgb (float). Otherwise it is loaded as an uint32/float column.
- columns representing an E3DPoint must have as suffixes "_x", "_y", "_z", be consecutive and of float/double type. They are stored as float in both cases. Otherwise they are loaded as 3 float/double columns.
- we do not have internal int16 or int8 types so those are converted to int32 types.
The particular attributes we define and the conditions to be parsed as one are:
- E3DAttribute_Color: name must be rgb (in which case alpha is not read and set to 255) or rgba, encoding must correspond to a color.
- E3DAttribute_Normal: names must be (normal_x, normal_y and normal_z) or (nx, ny and nz), encoding must correspond to a point.
- E3DAttribute_Intensity: name must be intensity, intensity must be a numeric type.
- E3DAttribute_Texture: name must be texture (with corresponding suffixes if type is color or point).
- E3DAttribute_Index: name must be index and type uint32 or int32.
- E3DAttribute_Confidence: name must be confidence and type should be float or double (but is converted internally to float).
- E3DAttribute_Distance: name must be distance and type should be float or double (but is converted internally to float).
Use EPointCloud::SavePCD to save to PCD file.

EPointCloud::LoadPLY

Loads an EPointCloud stored in the PLY file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadPLY(
 const std::string& path
 )
```

```
void LoadPLY(
 const std::string& path,
 const EFloatRange& undefinedZValues
 )
```

Parameters

path

The full path of the input file.

undefinedZValues

Optional parameter, discard the points with Z value in the given range.

Remarks

The PLY file format is documented here: http://paulbourke.net/dataformats/ply/.
When loading PLY files:
- in ascii files, all numbers are assumed to be written in base 10
- file is assumed to contain at least the floating point fields x, y and z. They mustn't be the first but have to be consecutive. If one of these points is nan, inf or too big to be represented on a float, the line is ignored. These points can be double but are stored internally as float.
- the other columns are loaded as user-custom attributes unless they match the specifications of our particular attributes
- the 3(4) columns representing a color must be of type uchar and end with _red, _green, _blue (, _alpha). They must be consecutive and in the red, green, blue (alpha) order.
- columns representing an E3DPoint must have as suffixes "_x", "_y", "_z", be consecutive and of float type. They are stored as float in both cases. Otherwise they are loaded as 3 float columns.
- we do not have internal int16 or int8 types so those are converted to int32 types.
The particular attributes we define and the conditions to be parsed as one are:
- E3DAttribute_Color: names must be red, green, blue (alpha), encoding must correspond to a color.
- E3DAttribute_Normal: names must be (nx, ny and nz) or (normal_x, normal_y and normal_z), encoding must correspond to a point.
- E3DAttribute_Intensity: name must be intensity, intensity must be a numeric type.
- E3DAttribute_Texture: name must be texture (with corresponding suffixes if type is color or point).
- E3DAttribute_Index: name must be index and type uint32 or int32.
- E3DAttribute_Confidence: name must be confidence and type should be float or double (but is converted internally to float).
- E3DAttribute_Distance: name must be distance and type should be float or double (but is converted internally to float).
Use EPointCloud::SavePLY to save to PLY file.

EPointCloud::LoadXYZ

Loads an EPointCloud stored in the XYZ file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadXYZ(
 const std::string& path
 )
void LoadXYZ(
 const std::string& path,
 const EFloatRange& undefinedZValues
 )
```

Parameters

path

The full path of the input file.

undefinedZValues

Optional parameter, discard the points with Z value in the given range.

Remarks

Only the x,y,z coordinates of the points of the pointcloud are loaded. Use pcd or ply if you need more. Use EPointCloud::SaveXYZ to save to XYZ file.

EPointCloud::GetNumPoints

Number of points in the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumPoints() const
```

EPointCloud::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloud& operator=(
 const EPointCloud& other
 )
```

Parameters

other

The EPointCloud object that should be copied.

EPointCloud::GetPointsBuffer

Retrieves a pointer to the internal points buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetPointsBuffer()
```

EPointCloud::PrepareSpacePartition

Enables the build of the space partition (like EPointCloud) and builds it now instead of when it is first needed. The computation time of the space partition depends on the number of points in the point cloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void PrepareSpacePartition(
 )
```

Remarks

See also EPointCloud::EnableSpacePartition.

EPointCloud::RemovePoint

Removes a point and its corresponding attributes from the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void RemovePoint(
 OEV_UINT32 index
 )
```

Parameters

index

The index of the point to remove.

EPointCloud::Save

Saves the EPointCloud. Supported formats are Open eVision proprietary, CSV, OBJ, PCD, PLY and XYZ. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EPointCloud::SaveCSV

Saves the EPointCloud in the CSV (Comma-Separated Values) file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveCSV(
 const std::string& path
 )
```

Parameters

path

The full path of the destination file.

Remarks

Only the x,y,z coordinates of the points of the pointcloud are saved. Use pcd or ply if you need more. Use EPointCloud::LoadCSV to load from CSV file.

EPointCloud::SaveOBJ

Saves the EPointCloud in the OBJ file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveOBJ(
 const std::string& path
 )
```

Parameters

path

The full path of the destination file.

Remarks

The OBJ file format is documented here:
https://www.fileformat.info/format/wavefrontobj/egff.htm.
The only attribute saved in the obj file format is the normals. Use pcd or ply if you need more.
Use EPointCloud::LoadOBJ to load from OBJ file.

EPointCloud::SavePCD

Saves the EPointCloud in the PCD (Point Cloud Library) file format.
ASCII and binary formats are available.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SavePCD(
 const std::string& path,
 bool binary
 )
```

Parameters

path

The full path of the destination file.

binary

Whether to store the file in binary instead of ascii, defaults to true.

Remarks

The PCD file format is documented here:
http://pointclouds.org/documentation/tutorials/pcd_file_format.html.
Custom user fields are named custom0, custom1,...
Colors are saved as an uint32_t (alpha, red, green, blue) and custom attributes' names are suffixed with _C. Points are saved as 3 consecutive floats and custom attributes' names are suffixed with _x, _y, _z.
Use EPointCloud::LoadPCD to load from PCD file.

EPointCloud::SavePLY

Saves the EPointCloud in the PLY file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SavePLY(
 const std::string& path,
 bool binary
 )
```

Parameters

path

The full path of the destination file.

binary

Whether to store the file in binary instead of ascii, defaults to true.

Remarks

The PLY file format is documented here: http://paulbourke.net/dataformats/ply/.
Custom user fields are named custom0, custom1,...
Colors are saved as 4 uchar fields suffixed with (_red, _green, _blue, _alpha) for custom attributes or named (red, green, blue, alpha) for the pointcloud's color attribute.
Points are saved as 3 consecutive floats whose names are suffixed with _x, _y, _z.
Use EPointCloud::LoadPLY to load from PLY file.

EPointCloud::SaveXYZ

Saves the EPointCloud in the XYZ file format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveXYZ(
 const std::string& path
 )
```

Parameters

path

The full path of the destination file.

Remarks

Only the x,y,z coordinates of the points of the pointcloud are saved. Use pcd or ply if you need more. Use EPointCloud::LoadXYZ to load from XYZ file.

EPointCloud::SetAttribute

Sets the corresponding attribute at the given index.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 OEV_UINT8 value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 OEV_UINT16 value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 OEV_UINT32 value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 int value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 float value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 double value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 const EC24A& value
 )

void SetAttribute(
 int attribute,
 OEV_UINT32 index,
 const E3DPoint& value
 )
```

Parameters

attribute

The attribute buffer id. Either an E3DAttribute or an id returned by EPointCloud::AddCustomAttributeBuffer.

index

The index of the element.

value

The new value of the element.

Remarks

This function does not perform an allocation. To add a new point with some attributes in the EPointCloud, use functions EPointCloud::AddPoint or EPointCloud::AddPoints which will add default value(s) to the attribute buffers that are initialized. Then, they can be set with this function. If the attribute has not been initialized or if the index is out of bound, it throws an EException.

4.187. EPointCloudFactory Class

Manages a context for creating point clouds of specific shapes.

Namespace: Euresys::Open_eVision::Easy3D

Methods

CreateCubicPointCloud	Creates a point cloud in the shape of a cube.
CreateRectangularPointCloud	Creates a point cloud in the shape of a rectangular parallelepiped.
CreateSphericPointCloud	Creates a point cloud in the shape of a sphere.

EPointCloudFactory::CreateCubicPointCloud

Creates a point cloud in the shape of a cube.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloud CreateCubicPointCloud(
 E3DPoint center,
 float size,
 float roll,
 float pitch,
 float yaw,
 OEV_UINT32 numSamples
 )
```

Parameters

center
Center of the cube.
size
Edge size of the cube.
roll
Roll (rotation along the X axis) of the cube.
pitch
Pitch (rotation along the Y axis) of the cube.
yaw
Yaw (rotation along the Z axis) of the cube.
numSamples
Number of points along each edge of the cube.

EPointCloudFactory::CreateRectangularPointCloud

Creates a point cloud in the shape of a rectangular parallelepiped.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloud CreateRectangularPointCloud(
 E3DPoint center,
 float width,
 float height,
 float depth,
 float roll,
 float pitch,
 float yaw,
 OEV_UINT32 numSamples
 )
```

Parameters

center
Center of the rectangular parallelepiped.
width
Width (size along the X axis before rotation) of the rectangular parallelepiped.
height
Height (size along the Y axis before rotation) of the rectangular parallelepiped.
depth
Depth (size along the Z axis before rotation) of the rectangular parallelepiped.
roll
Roll (rotation along the X axis) of the rectangular parallelepiped.
pitch
Pitch (rotation along the Y axis) of the rectangular parallelepiped.

yaw
Yaw (rotation along the Z axis) of the rectangular parallelepiped.
numSamples
Number of points along each edge of the rectangular parallelepiped.

EPointCloudFactory::CreateSphericPointCloud

Creates a point cloud in the shape of a sphere.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloud CreateSphericPointCloud(
 const E3DPoint& center,
 float radius,
 int numCircles,
 int numSamples
 )
```

Parameters

center
Center of the sphere.
radius
Radius of the sphere.
numCircles
Number of parallels and meridians to be rendered.
numSamples
Number of points along each meridian and parallel.

4.188. EPointCloudFilter Class

An EPointCloudFilter is used to filter some points out of an EPointCloud.
The points to filter out are selected according to a criteria (see EPointCloudFilteringMethod) based on their neighborhood.

Namespace: Euresys::Open_eVision::Easy3D

Methods

EPointCloudFilter	Creates an EPointCloudFilter object to filter out some points in an EPointCloud.
Filter	Filters the input point cloud and puts the result of the operation in the output point cloud.
GetFilteringMethod	Sets/Gets the filtering method.

GetNumberOfNeighbors	Sets/Gets the number of neighbors.
GetReplaceByAverage	Sets/Gets whether to replace filtered points by their average.
GetThresholdMultiplier	Sets/Gets the filtering threshold multiplier.
Load	Loads the EPointCloudFilter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the EPointCloudFilter configuration. The given ESerializer must have been created for writing.
SetFilteringMethod	Sets/Gets the filtering method.
SetNumberOfNeighbors	Sets/Gets the number of neighbors.
SetReplaceByAverage	Sets/Gets whether to replace filtered points by their average.
SetThresholdMultiplier	Sets/Gets the filtering threshold multiplier.

EPointCloudFilter::EPointCloudFilter

Creates an EPointCloudFilter object to filter out some points in an EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EPointCloudFilter(
 float thresholdMultiplier,
 OEV_UINT32 numberOfNeighbors,
 Euresys::Open_eVision::Easy3D::EPointCloudFilteringMethod method,
 bool replaceByAverage
 )

void EPointCloudFilter(
 const EPointCloudFilter& other
 )
```

Parameters

thresholdMultiplier

Points whose criterion is greater than mean + thresholdMultiplier * stddev are removed, mean and stddev being the mean and standard derivation of the criterion across all points of the cloud. Default: 1.

numberOfNeighbors

Number of neighbors in the neighborhood used to determine if a point is filtered or not. Default: 10

method

EPointCloudFilteringMethod used to select the points to filter. Default: EPointCloudFilteringMethod_HighStandardDerivation.

replaceByAverage

Set to true to replace points to filter out by the mean of their neighborhood instead of removing them. Default: false

other

The point cloud filterer we copy from.

EPointCloudFilter::Filter

Filters the input point cloud and puts the result of the operation in the output point cloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Filter(
 const EPointCloud& cloudIn,
 EPointCloud& cloudOut
 )
```

Parameters

cloudIn

Input point cloud that will be filtered

cloudOut

Output point cloud that has been filtered

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

EPointCloudFilter::GetFilteringMethod

EPointCloudFilter::SetFilteringMethod

Sets/Gets the filtering method.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EPointCloudFilteringMethod GetFilteringMethod() const

void SetFilteringMethod(Euresys::Open_eVision::Easy3D::EPointCloudFilteringMethod
method)
```

EPointCloudFilter::Load

Loads the EPointCloudFilter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is read from

EPointCloudFilter::GetNumberOfNeighbors

EPointCloudFilter::SetNumberOfNeighbors

Sets/Gets the number of neighbors.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
OEV_UINT32 GetNumberOfNeighbors() const

void SetNumberOfNeighbors(OEV_UINT32 numberOfNeighbors)
```

EPointCloudFilter::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EPointCloudFilter& operator=(
 const EPointCloudFilter& other
 )
```

Parameters

other

An other EPointCloudFilter.

EPointCloudFilter::GetReplaceByAverage

EPointCloudFilter::SetReplaceByAverage

Sets/Gets whether to replace filtered points by their average.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetReplaceByAverage() const

void SetReplaceByAverage(bool replaceByAverage)
```

EPointCloudFilter::Save

Saves the EPointCloudFilter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to

EPointCloudFilter::GetThresholdMultiplier

EPointCloudFilter::SetThresholdMultiplier

Sets/Gets the filtering threshold multiplier.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetThresholdMultiplier() const
void SetThresholdMultiplier(float thresholdMultiplier)
```

4.189. EPointCloudMerger Class

Merges several EPointCloud of the same scene from different sensors after having performed a calibration.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Calibrate	Calibrates the transformations to apply to the point clouds from clouds of the calibration object.
EPointCloudMerger	Constructs an EPointCloudMerger.
GetEnableDecimation	Sets/Gets whether the filtering of the merged cloud is enabled. If set to true, an EGridDecimator is used to perform a grid decimation so that a point present in several clouds is not duplicated.
GetMergedCloudResolution	Sets/Gets the resolution of the merged point clouds.
GetTransformations	Sets/Gets the transformations to apply to all of the clouds to merge.
Load	Loads the EPointCloudMerger. The given ESerializer must have been created for reading.
Merge	Merges the given clouds by applying the transformation matrices and optionally filter the result to remove duplicate points.
operator=	Assignment operator.
Save	Saves the EPointCloudMerger. The given ESerializer must have been created for writing.
SetEnableDecimation	Sets/Gets whether the filtering of the merged cloud is enabled. If set to true, an EGridDecimator is used to perform a grid decimation so that a point present in several clouds is not duplicated.
SetMergedCloudResolution	Sets/Gets the resolution of the merged point clouds.

SetTransformations Sets/Gets the transformations to apply to all of the clouds to merge.

EPointCloudMerger::Calibrate

Calibrates the transformations to apply to the point clouds from clouds of the calibration object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float Calibrate(
 const std::vector<Euresys::Open_eVision::Easy3D::EPointCloud>& calibrationClouds,
 float calibrationObjectSize,
 bool computeMergedCloudResolution
 )
```

Parameters

calibrationClouds

A vector of EPointCloud representing different points of view of the calibration object.

calibrationObjectSize

The size of an edge of the calibration cube in the point clouds.

computeMergedCloudResolution

Whether to compute the resolution of the merged cloud from the given clouds or not. If set to true, the resolution will be set to half the average distance between a point on the calibration object of the first cloud and its nearest neighbor.

Remarks

The transformations to apply and possibly the mergedCloudResolution can also be set by EPointCloudMerger::Transformations and EPointCloudMerger::MergedCloudResolution respectively.
The biggest face of the calibration cube should at least contain 2% of the points in each calibration cloud for the object to be detected.

EPointCloudMerger::GetEnableDecimation

EPointCloudMerger::SetEnableDecimation

Sets/Gets whether the filtering of the merged cloud is enabled. If set to true, an EGridDecimator is used to perform a grid decimation so that a point present in several clouds is not duplicated.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetEnableDecimation() const

void SetEnableDecimation(bool enableDecimation)
```

EPointCloudMerger::EPointCloudMerger

Constructs an EPointCloudMerger.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EPointCloudMerger(
 )

void EPointCloudMerger(
 const EPointCloudMerger& other
 )

void EPointCloudMerger(
 bool enableDecimation,
 float mergedCloudResolution,
 const std::vector<Euresys::Open_eVision::Easy3D::E3DTransformMatrix>& transforms
 )
```

Parameters

other

Another EPointCloudMerger object to be copied in the new EPointCloudMerger object.

enableDecimation

Whether we want to filter the merged cloud to remove point duplicates coming from different clouds by using an EGridDecimator or not. Set to true by default.

mergedCloudResolution

The size of a cubic cell of EGridDecimator applied to the merged EPointCloud. Set to 1 by default.

transforms

A vector of E3DTransformMatrix that will be applied to each EPointCloud before merging them together.

EPointCloudMerger::Load

Loads the EPointCloudMerger. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPointCloudMerger::Merge

Merges the given clouds by applying the transformation matrices and optionally filter the result to remove duplicate points.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Merge(
 const std::vector<Euresys::Open_eVision::Easy3D::EPointCloud>& clouds,
 EPointCloud& mergedCloud
 )
```

Parameters

clouds

-

mergedCloud

An empty cloud that will be filled with the merging of the calibrationClouds. Useful to manually assess the results.

Remarks

The EPointCloudMerger must have been calibrated or the transformations to apply set, see EPointCloudMerger::Calibrate, EPointCloudMerger or constructor.

EPointCloudMerger::GetMergedCloudResolution

EPointCloudMerger::SetMergedCloudResolution

Sets/Gets the resolution of the merged point clouds.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetMergedCloudResolution() const
void SetMergedCloudResolution(float mergedCloudResolution)
```

EPointCloudMerger::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EPointCloudMerger& operator=(
 const EPointCloudMerger& other
 )
```

Parameters

other

The EPointCloudMerger object that should be assigned.

EPointCloudMerger::Save

Saves the EPointCloudMerger. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EPointCloudMerger::GetTransformations

EPointCloudMerger::SetTransformations

Sets/Gets the transformations to apply to all of the clouds to merge.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::vector<Euresys::Open_eVision::Easy3D::E3DTransformMatrix> GetTransformations()
const

void SetTransformations(const std::vector<Euresys::Open_
eVision::Easy3D::E3DTransformMatrix>& transforms)
```

Remarks

The transformations to apply can also be produced by EPointCloudMerger::Calibrate.

4.190. EPointCloudStatistics Class

Manages a context for retrieving statistics on an EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

Methods

GetPointCloudBounds	Retrieves the bounds of an EPointCloud.
GetPointCloudCentroid	Retrieves the centroid (arithmetic mean position of all the points, also known as center of gravity or barycenter) of an EPointCloud. It is possible to get the centroid of a sphere or a rectangle (rectangular parallelepiped) shape inside the point cloud. The sphere is defined by it's center and radius, in the EPointCloud coordinate system. The 3D rectangle is defined by 3 ranges, in X, Y and Z axis, in the EPointCloud coordinate system. An exception will be thrown if no point is present in the shape or the point cloud.

EPointCloudStatistics::GetPointCloudBounds

Retrieves the bounds of an EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetPointCloudBounds(
 const EPointCloud& cloud,
 EFloatRange& rangeX,
 EFloatRange& rangeY,
 EFloatRange& rangeZ
 )
```

Parameters

cloud
Point cloud.

rangeX
Bounds of the point cloud along the X direction.

rangeY
Bounds of the point cloud along the Y direction.

rangeZ
Bounds of the point cloud along the Z direction.

EPointCloudStatistics::GetPointCloudCentroid

Retrieves the centroid (arithmetic mean position of all the points, also known as center of gravity or barycenter) of an EPointCloud.
It is possible to get the centroid of a sphere or a rectangle (rectangular parallelepiped) shape inside the point cloud.
The sphere is defined by it's center and radius, in the EPointCloud coordinate system.
The 3D rectangle is defined by 3 ranges, in X, Y and Z axis, in the EPointCloud coordinate system.
An exception will be thrown if no point is present in the shape or the point cloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetPointCloudCentroid(
 const EPointCloud& cloud
 )

E3DPoint GetPointCloudCentroid(
 const EPointCloud& cloud,
 const E3DPoint& sphereCenter,
 float sphereRadius
 )

E3DPoint GetPointCloudCentroid(
 const EPointCloud& cloud,
 const EFloatRange& rangeX,
 const EFloatRange& rangeY,
 const EFloatRange& rangeZ
 )
```

Parameters

cloud
Point cloud.
sphereCenter
The position of the center of the sphere.
sphereRadius
The radius of the sphere.
rangeX
The bounds along the X direction.
rangeY
The bounds along the Y direction.
rangeZ
The bounds along the Z direction.

4.191. EPointCloudToMeshConverter Class

Performs the conversion from an EPointCloud to an EMesh.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	The method 'Convert' performs the conversion from an EPointCloud to an EMesh. The cloud is first projected on a Plane from which the mesh is generated. This method is thus expecting 2.5D point clouds.
EPointCloudToMeshConverter	Creates an EPointCloudToMeshConverter object.
GetMaxEdgeLength	Sets/Gets the maximal length of the longest edge of a triangle of the mesh. Larger triangles will be filtered out. A value of 0 does not perform any filtering. Set to 0 by default.
GetProjectionPlane	The plane on which the cloud is projected before generating a mesh. The parts of the object behind the plane are ignored. Set to the "z = 0" plane by default.
GetResolution	Approximate size of a side of a triangle composing the mesh, set to 0 to have the value automatically computed.
Load	Loads the converter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator
Save	Saves the converter configuration. The given ESerializer must have been created for writing.
SetMaxEdgeLength	Sets/Gets the maximal length of the longest edge of a triangle of the mesh. Larger triangles will be filtered out. A value of 0 does not perform any filtering. Set to 0 by default.

SetProjectionPlane	The plane on which the cloud is projected before generating a mesh. The parts of the object behind the plane are ignored. Set to the "z = 0" plane by default.
SetResolution	Approximate size of a side of a triangle composing the mesh, set to 0 to have the value automatically computed.

EPointCloudToMeshConverter::Convert

The method 'Convert' performs the conversion from an EPointCloud to an EMesh. The cloud is first projected on a Plane from which the mesh is generated. This method is thus expecting 2.5D point clouds.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Convert(
 const EPointCloud& cloud,
 EMesh& mesh
 )
```

Parameters

cloud

The cloud to convert.

mesh

The destination mesh.

EPointCloudToMeshConverter::EPointCloudToMeshConverter

Creates an EPointCloudToMeshConverter object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EPointCloudToMeshConverter(
 )

void EPointCloudToMeshConverter(
 const EPointCloudToMeshConverter& other
 )
```

Parameters

other

Reference to the EPointCloudToMeshConverter object used for the initialization.

EPointCloudToMeshConverter::Load

Loads the converter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPointCloudToMeshConverter::GetMaxEdgeLength

EPointCloudToMeshConverter::SetMaxEdgeLength

Sets/Gets the maximal length of the longest edge of a triangle of the mesh. Larger triangles will be filtered out. A value of 0 does not perform any filtering. Set to 0 by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMaxEdgeLength() const

void SetMaxEdgeLength(float maxEdgeLength)
```

EPointCloudToMeshConverter::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloudToMeshConverter& operator=(
 const EPointCloudToMeshConverter& other
 )
```

Parameters

other

Reference to the EPointCloudToMeshConverter object used for the assignment.

EPointCloudToMeshConverter::GetProjectionPlane

EPointCloudToMeshConverter::SetProjectionPlane

The plane on which the cloud is projected before generating a mesh. The parts of the object behind the plane are ignored. Set to the "z = 0" plane by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane GetProjectionPlane() const
void SetProjectionPlane(const E3DPlane& projectionPlane)
```

EPointCloudToMeshConverter::GetResolution

EPointCloudToMeshConverter::SetResolution

Approximate size of a side of a triangle composing the mesh, set to 0 to have the value automatically computed.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetResolution() const
void SetResolution(float resolution)
```

EPointCloudToMeshConverter::Save

Saves the converter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

Pointer to the ESerializer created for writing.

4.192. EPointCloudToZMapConverter Class

Computes an EZMap from an EPointCloud. The value of the pixels of the ZMap are the distance between the 3D points and the reference plane.
All 3D points under the reference plane are discarded.
Various options can be set with methods EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::SetFillMode, EPointCloudToZMapConverter::SetMapXYResolution, EPointCloudToZMapConverter::MapZResolution, EPointCloudToZMapConverter::OrientationVector...
When the conversion is called without defining specific parameters, the algorithm uses the following options:
- The reference plane is the horizontal plane.
- The orientation vector is selected automatically.
- The origin is set as the lowest left position of the projected point cloud on the reference plane.
- The resolution (the dimensions of the Z map) is estimated to have approximately one Point Cloud point per ZMap pixels.
- The scale is calculated from the point cloud ranges and the estimated resolution.
- The fill mode is enabled and the method is set to 'EFillUndefinedPixelsDirection_Local' (see method EDepthMap8::FillUndefinedPixels).

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	Computes an EZMap from a world space EPointCloud. The value of the pixels of the ZMap are the distance between the 3D points and the reference plane. Various options can be set with methods EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector, EPointCloudToZMapConverter::SetMapSize, ...
EnableFillMode	Enables or disables fill mode. Fill mode parameters are defined by method EPointCloudToZMapConverter::SetFillMode or EPointCloudToZMapConverter::SetFillModeMedian. Fill mode is enabled by default. If fill mode is disable, undefined pixels may remain in the EZMap.
EPointCloudToZMapConverter	Creates an EPointCloudToZMapConverter object.
GetExtension	Sets a metric value used to enlarge the point cloud 3D domain. That value affects X,Y and Z directions and can be used to generate an EZMap with borders of undefined pixels. Default value is 0, which means a ZMap without border.

GetFillUndefinedPixelsDirection	Gets the undefined pixel fill direction (see EFillUndefinedPixelsDirection).
GetFillUndefinedPixelsMethod	Gets the undefined pixel fill method (see EFillUndefinedPixelsMethod).
GetMapHeight	Gets the required height (number of pixels) of the generated EZMap. By default, the required size is not set.
GetMapWidth	Gets the required width (number of pixels) of the generated EZMap. By default, the required size is not set.
GetMapXResolution	Gets the resolution of the EZMap pixels along the X axis.
GetMapYResolution	Gets the resolution of the EZMap pixels along the Y axis.
GetMapZResolution	Gets/sets the EZMap Z resolution, in world space units per gray value. The resolution is used to compute the transformation of the world Z position to an integer 8, 16 or 32 bits pixel value.
GetOrientationVector	Sets an explicit orientation for the EZMap. Overrides the orientation mode given by the method EPointCloudToZMapConverter::OrientationVectorMode.
GetOrientationVectorMode	Chooses the EZMap orientation from a list of predefined axis, automatic mode or user defined vector. Use EPointCloudToZMapConverter::OrientationVector to set an explicit orientation vector for the ZMap.
GetOrigin	Chooses the EZMap origin. It is the 3D world position used as origin of the ZMap upper left pixel (0,0).
GetReferencePlane	Sets the E3DPlane reference plane. The resulting EZMap is the distance of the 3D points above that plane. 3D points below the reference plane are discarded.
GetReferencePlaneMode	Sets an axis aligned reference plane. Overrides the explicit reference plane given by the method EPointCloudToZMapConverter::ReferencePlane.
GetWorldToZMapTransform	Explicitly sets the world to ZMap transformation. EPointCloudToZMapConverter::WorldToZMapTransform overrides the settings done by EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector and EPointCloudToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the world positions are transformed to the EZMap space. The matrix must be a rigid transformation (translation and rotation only). The resolutions of the ZMap are defined by the EPointCloudToZMapConverter::SetMapXYResolution and EPointCloudToZMapConverter::MapZResolution methods.

GetZMaptoWorldTransform	Explicitly sets the ZMap to World transformation. "SetZMaptoWorldTransform" overrides the settings done by EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector and EPointCloudToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the EZMap positions are transformed to the World space. The matrix must be a rigid transformation (translation and rotation only). The resolutions of the World are defined by the EPointCloudToZMapConverter::SetMapXYResolution and EPointCloudToZMapConverter::MapZResolution methods.
IsFillModeEnabled	Tells if the fill mode is enabled or not. Use EPointCloudToZMapConverter::EnableFillMode to toggle the fill mode and EPointCloudToZMapConverter::SetFillMode or EPointCloudToZMapConverter::SetFillModeMedian to set the filling parameters.
Load	Loads the converter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator
operator==	Comparison operator
Save	Saves the converter configuration. The given ESerializer must have been created for writing.
SetExtension	Sets a metric value used to enlarge the point cloud 3D domain. That value affects X,Y and Z directions and can be used to generate an EZMap with borders of undefined pixels. Default value is 0, which means a ZMap without border.
SetFillMode	Inpainting options used to fill the "holes" in the EZMap. A hole exists when no 3D point is projected at that pixel position in the ZMap.
SetFillModeMedian	Inpainting options used to fill the "holes" in the EZMap using a median rectangular kernel of odd size. A hole exists when no 3D point is projected at that pixel position in the ZMap.
SetMapSize	Sets the required size of the generated EZMap; expressed in number of pixels for width and height dimensions. By default, the required size is not set.
SetMapXYResolution	Sets the resolution (possibly anisotropic) of the EZMap pixels along the X and Y axes, in world space units per pixel (e.g mm/pixel). The resolution is used to compute the ZMap size (width and height), depending on the projected point cloud on the reference plane.
SetMapZResolution	Gets/sets the EZMap Z resolution, in world space units per gray value. The resolution is used to compute the transformation of the world Z position to an integer 8, 16 or 32 bits pixel value.
SetOrientationVector	Sets an explicit orientation for the EZMap. Overrides the orientation mode given by the method EPointCloudToZMapConverter::OrientationVectorMode.

SetOrientationVectorMode	Chooses the EZMap orientation from a list of predefined axis, automatic mode or user defined vector. Use EPointCloudToZMapConverter::OrientationVector to set an explicit orientation vector for the ZMap.
SetOrigin	Chooses the EZMap origin. It is the 3D world position used as origin of the ZMap upper left pixel (0,0).
SetReferencePlane	Sets the E3DPlane reference plane. The resulting EZMap is the distance of the 3D points above that plane. 3D points below the reference plane are discarded.
SetReferencePlaneMode	Sets an axis aligned reference plane. Overrides the explicit reference plane given by the method EPointCloudToZMapConverter::ReferencePlane.
SetWorldToZMapTransform	Explicitly sets the world to ZMap transformation. EPointCloudToZMapConverter::WorldToZMapTransform overrides the settings done by EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector and EPointCloudToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the world positions are transformed to the EZMap space. The matrix must be a rigid transformation (translation and rotation only). The resolutions of the ZMap are defined by the EPointCloudToZMapConverter::SetMapXYResolution and EPointCloudToZMapConverter::MapZResolution methods.
SetZMaptoWorldTransform	Explicitly sets the ZMap to World transformation. "SetZMaptoWorldTransform" overrides the settings done by EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector and EPointCloudToZMapConverter::Origin. That E3DTransformMatrix transform expresses how the EZMap positions are transformed to the World space. The matrix must be a rigid transformation (translation and rotation only). The resolutions of the World are defined by the EPointCloudToZMapConverter::SetMapXYResolution and EPointCloudToZMapConverter::MapZResolution methods.
UnsetMapSize	Unsets the resolution. Lets the conversion decides for the optimum resolution, depending on the projected point cloud and pixel scale.
UnsetMapXYResolution	Unsets the X and Y resolutions. Lets the conversion decide the optimal resolution, depending on the projected point cloud and EZMap size.
UnsetMapZResolution	Unsets the ZMap Z resolution. Lets the conversion decide the optimal Z resolution.
UnsetOrigin	Lets the conversion process decides for the EZMap origin position (based on projected point cloud on the reference plane). Use EPointCloudToZMapConverter::Origin to enable and choose the ZMap origin.
UnsetWorldToZMapTransform	Disables the explicit world to ZMap transformation, set with EPointCloudToZMapConverter::WorldToZMapTransform.

EPointCloudToZMapConverter::Convert

Computes an EZMap from a world space EPointCloud. The value of the pixels of the ZMap are the distance between the 3D points and the reference plane. Various options can be set with methods EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector, EPointCloudToZMapConverter::SetMapSize, ...

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Convert(
 const EPointCloud& cloud,
 EZMap8& zmap
 )
void Convert(
 const EPointCloud& cloud,
 EZMap16& zmap
 )
void Convert(
 const EPointCloud& cloud,
 EZMap32f& zmap
 )
void Convert(
 const EPointCloud& cloud,
 EZMap* zmap
 )
```

Parameters

cloud

The input 3D point cloud.

zmap

The generated ZMap in 8, 16 or 32 bits format.

EPointCloudToZMapConverter::EnableFillMode

Enables or disables fill mode. Fill mode parameters are defined by method EPointCloudToZMapConverter::SetFillMode or EPointCloudToZMapConverter::SetFillModeMedian.
Fill mode is enabled by default. If fill mode is disable, undefined pixels may remain in the EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EnableFillMode(
 bool state
 )
```

Parameters

state

Set to true to enable fill mode.

EPointCloudToZMapConverter::EPointCloudToZMapConverter

Creates an EPointCloudToZMapConverter object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void EPointCloudToZMapConverter(
 )
void EPointCloudToZMapConverter(
 const EPointCloudToZMapConverter& other
 )
```

Parameters

other

Reference to the EPointCloudToZMapConverter object used for the initialization.

EPointCloudToZMapConverter::GetExtension

EPointCloudToZMapConverter::SetExtension

Sets a metric value used to enlarge the point cloud 3D domain.
That value affects X,Y and Z directions and can be used to generate an EZMap with borders of undefined pixels.
Default value is 0, which means a ZMap without border.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetExtension() const
void SetExtension(float size)
```

EPointCloudToZMapConverter::GetFillUndefinedPixelsDirection

Gets the undefined pixel fill direction (see EFillUndefinedPixelsDirection).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection
GetFillUndefinedPixelsDirection() const
```

EPointCloudToZMapConverter::GetFillUndefinedPixelsMethod

Gets the undefined pixel fill method (see EFillUndefinedPixelsMethod).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod GetFillUndefinedPixelsMethod
() const
```

EPointCloudToZMapConverter::IsFillModeEnabled

Tells if the fill mode is enabled or not.
Use EPointCloudToZMapConverter::EnableFillMode to toggle the fill mode and EPointCloudToZMapConverter::SetFillMode or EPointCloudToZMapConverter::SetFillModeMedian to set the filling parameters.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsFillModeEnabled(
 )
```

EPointCloudToZMapConverter::Load

Loads the converter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EPointCloudToZMapConverter::GetMapHeight

Gets the required height (number of pixels) of the generated EZMap.
By default, the required size is not set.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetMapHeight() const
```

EPointCloudToZMapConverter::GetMapWidth

Gets the required width (number of pixels) of the generated EZMap.
By default, the required size is not set.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetMapWidth() const
```

EPointCloudToZMapConverter::GetMapXResolution

Gets the resolution of the EZMap pixels along the X axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetMapXResolution() const
```

EPointCloudToZMapConverter::GetMapYResolution

Gets the resolution of the EZMap pixels along the Y axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetMapYResolution() const
```

EPointCloudToZMapConverter::GetMapZResolution

EPointCloudToZMapConverter::SetMapZResolution

Gets/sets the EZMap Z resolution, in world space units per gray value.
The resolution is used to compute the transformation of the world Z position to an integer 8, 16 or 32 bits pixel value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMapZResolution() const

void SetMapZResolution(float scale)
```

EPointCloudToZMapConverter::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPointCloudToZMapConverter& operator=(
 const EPointCloudToZMapConverter& other
 )
```

Parameters

other

Reference to the EPointCloudToZMapConverter object used for the assignment.

EPointCloudToZMapConverter::operator==

Comparison operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator==(
 const EPointCloudToZMapConverter& other
 )
```

Parameters

other

Reference to the EPointCloudToZMapConverter object used for the comparison.

EPointCloudToZMapConverter::GetOrientationVector

EPointCloudToZMapConverter::SetOrientationVector

Sets an explicit orientation for the EZMap.
Overrides the orientation mode given by the method EPointCloudToZMapConverter::OrientationVectorMode.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetOrientationVector() const

void SetOrientationVector(const E3DPoint& direction)
```

Remarks

The direction should be an E3DPoint representing the expected direction of the X (width) axis of the ZMap.
That direction will be used after projection on the reference plane normal.
That direction must NOT be aligned with the reference plane normal.

EPointCloudToZMapConverter::GetOrientationVectorMode

EPointCloudToZMapConverter::SetOrientationVectorMode

Chooses the EZMap orientation from a list of predefined axis, automatic mode or user defined vector.
Use EPointCloudToZMapConverter::OrientationVector to set an explicit orientation vector for the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EZMapOrientationVectorMode GetOrientationVectorMode() const

void SetOrientationVectorMode(Euresys::Open_eVision::Easy3D::EZMapOrientationVectorMode mode)
```

Remarks

Choose between Automatic mode (default), world space axis or explicit user defined vector (see EZMapOrientationVectorMode).

EPointCloudToZMapConverter::GetOrigin

EPointCloudToZMapConverter::SetOrigin

Chooses the EZMap origin. It is the 3D world position used as origin of the ZMap upper left pixel (0,0).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetOrigin() const

void SetOrigin(const E3DPoint& position)
```

Remarks

That position will be projected on the reference plane.
To let the conversion chooses for the origin, call EPointCloudToZMapConverter::UnsetOrigin.

EPointCloudToZMapConverter::GetReferencePlane

EPointCloudToZMapConverter::SetReferencePlane

Sets the E3DPlane reference plane.
The resulting EZMap is the distance of the 3D points above that plane.
3D points below the reference plane are discarded.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPlane GetReferencePlane() const

void SetReferencePlane(const E3DPlane& plane)
```

EPointCloudToZMapConverter::GetReferencePlaneMode

EPointCloudToZMapConverter::SetReferencePlaneMode

Sets an axis aligned reference plane.
Overrides the explicit reference plane given by the method EPointCloudToZMapConverter::ReferencePlane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::EZMapReferencePlaneMode GetReferencePlaneMode() const
```

```
void SetReferencePlaneMode(Euresys::Open_eVision::Easy3D::EZMapReferencePlaneMode mode)
```

Remarks

Choose between X, Y or Z reference plane (see EZMapReferencePlaneMode).
The plane offset is set automatically on the point cloud lowest 3D point.

EPointCloudToZMapConverter::Save

Saves the converter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

Pointer to the ESerializer created for writing.

EPointCloudToZMapConverter::SetFillMode

Inpainting options used to fill the "holes" in the EZMap. A hole exists when no 3D point is projected at that pixel position in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetFillMode(
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

direction

Direction in which the undefined pixels are filled in a depthmap from EFillUndefinedPixelsDirection

method

Which values used to fill the undefined pixels in a depthmap from EFillUndefinedPixelsMethod

EPointCloudToZMapConverter::SetFillModeMedian

Inpainting options used to fill the "holes" in the EZMap using a median rectangular kernel of odd size. A hole exists when no 3D point is projected at that pixel position in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetFillModeMedian(
 OEV_UINT32 halfKernelX,
 OEV_UINT32 halfKernelY
 )
```

Parameters

halfKernelX

-

halfKernelY

-

EPointCloudToZMapConverter::SetMapSize

Sets the required size of the generated EZMap; expressed in number of pixels for width and height dimensions.
By default, the required size is not set.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetMapSize(
 int width,
 int height
 )
```

Parameters

width

The required width for the Generated ZMap.

height

The required height for the Generated ZMap.

EPointCloudToZMapConverter::SetMapXYResolution

Sets the resolution (possibly anisotropic) of the EZMap pixels along the X and Y axes, in world space units per pixel (e.g mm/pixel).
The resolution is used to compute the ZMap size (width and height), depending on the projected point cloud on the reference plane.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetMapXYResolution(
 float resolution
 )
void SetMapXYResolution(
 float resolutionX,
 float resolutionY
 )
```

Parameters

resolution

The resolution for the isotropic case.

resolutionX

The resolution for the X axis.

resolutionY

The resolution for the Y axis.

Remarks

The isotropic scale, for X and Y axis is in metric world units.

EPointCloudToZMapConverter::UnsetMapSize

Unsets the resolution. Lets the conversion decides for the optimum resolution, depending on the projected point cloud and pixel scale.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetMapSize(
 )
```

EPointCloudToZMapConverter::UnsetMapXYResolution

Unsets the X and Y resolutions. Lets the conversion decide the optimal resolution, depending on the projected point cloud and EZMap size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetMapXYResolution(
 )
```

EPointCloudToZMapConverter::UnsetMapZResolution

Unsets the ZMap Z resolution. Lets the conversion decide the optimal Z resolution.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetMapZResolution(
 )
```

EPointCloudToZMapConverter::UnsetOrigin

Lets the conversion process decides for the EZMap origin position (based on projected point cloud on the reference plane).
Use EPointCloudToZMapConverter::Origin to enable and choose the ZMap origin.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetOrigin(
 )
```

EPointCloudToZMapConverter::UnsetWorldToZMapTransform

Disables the explicit world to ZMap transformation, set with EPointCloudToZMapConverter::WorldToZMapTransform.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void UnsetWorldToZMapTransform(
 )
```

EPointCloudToZMapConverter::GetWorldToZMapTransform

EPointCloudToZMapConverter::SetWorldToZMapTransform

Explicitly sets the world to ZMap transformation.
EPointCloudToZMapConverter::WorldToZMapTransform overrides the settings done by EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector and EPointCloudToZMapConverter::Origin.
That E3DTransformMatrix transform expresses how the world positions are transformed to the EZMap space.
The matrix must be a rigid transformation (translation and rotation only).
The resolutions of the ZMap are defined by the EPointCloudToZMapConverter::SetMapXYResolution and EPointCloudToZMapConverter::MapZResolution methods.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetWorldToZMapTransform() const

void SetWorldToZMapTransform(const E3DTransformMatrix& matrix)
```

EPointCloudToZMapConverter::GetZMaptoWorldTransform

EPointCloudToZMapConverter::SetZMaptoWorldTransform

Explicitly sets the ZMap to World transformation.
"SetZMaptoWorldTransform" overrides the settings done by EPointCloudToZMapConverter::ReferencePlane, EPointCloudToZMapConverter::OrientationVector and EPointCloudToZMapConverter::Origin.
That E3DTransformMatrix transform expresses how the EZMap positions are transformed to the World space.
The matrix must be a rigid transformation (translation and rotation only).
The resolutions of the World are defined by the EPointCloudToZMapConverter::SetMapXYResolution and EPointCloudToZMapConverter::MapZResolution methods.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetZMaptoWorldTransform() const

void SetZMaptoWorldTransform(const E3DTransformMatrix& matrix)
```

4.193. EPointGauge Class

Manages a point location gauge.

Base Class:EPointShape

Namespace: Euresys::Open_eVision

Methods

CopyTo	Copies all the data of the current EPointGauge object into another EPointGauge object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
EPointGauge	Constructs a point measurement context.
GetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
GetMeasuredPeak	Returns information pertaining to the derivative peak associated with the specified edge-crossing point, such as its area, amplitude, start, length and center.
GetMeasuredPoint	Returns the coordinates of an edge-crossing point, measured along the unique sample path of the point location gauge.
GetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
GetMinArea	Minimum area value.
GetNumMeasuredPoints	Number of edge-crossing points along the point location gauge.
GetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
GetSmoothing	Number of pixels used for the low-pass filtering operation.
GetThickness	Number of parallel segments used to extract the data profile.
GetThreshold	Threshold level used to delimit significant peaks in the data profile.
GetTolerance	Half length of the point location gauge.
GetToleranceAngle	Rotation angle of the point location gauge.
GetTransitionChoice	Transition choice.
GetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.

GetTransitionType	Transition type.
GetType	Shape type.
GetValid	Flag indicating if at least one valid transition has been found.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Measure	Triggers the point location or the model fitting operation.
operator=	Copies all the data from another EPointGauge object into the current EPointGauge object
Plot	Draws the profile that is crossed by a point location gauge, as defined by EPlotItem.
PlotWithCurrentPen	Draws the profile that is crossed by a point location gauge, as defined by EPlotItem.
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.
SetActive	Sets the flag indicating whether the gauge is active or not.
SetCenter	Center coordinates of a EPointGauge object.
SetCenterXY	Sets the center coordinates of a EPointGauge object.
SetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
SetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
SetMinArea	Minimum area value.
SetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
SetSmoothing	Number of pixels used for the low-pass filtering operation.
SetThickness	Number of parallel segments used to extract the data profile.
SetThreshold	Threshold level used to delimit significant peaks in the data profile.
SetTolerance	Half length of the point location gauge.
SetToleranceAngle	Rotation angle of the point location gauge.
SetTolerances	Sets the half length and the rotation angle of the point location gauge.
SetTransitionChoice	Transition choice.
SetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
SetTransitionType	Transition type.

EPointGauge::SetActive

Sets the flag indicating whether the gauge is active or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetActive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying EPointGauge::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (true).

EPointGauge::SetCenter

Center coordinates of a EPointGauge object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenter(const EPoint& center)
```

EPointGauge::CopyTo

Copies all the data of the current EPointGauge object into another EPointGauge object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EPointGauge& other,
 bool recursive
 )

EPointGauge* CopyTo(
 EPointGauge* other,
 bool recursive
 )
```

Parameters

other

Pointer to the EPointGauge object in which the current EPointGauge object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EPointGauge object will be created and returned.

EPointGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y
 )
```

Parameters

x

Cursor current X coordinate.

y

Cursor current Y coordinate.

EPointGauge::Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPointGauge::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPointGauge::EPointGauge

Constructs a point measurement context.

Namespace: Euresys::Open_eVision

[C++]

```
void EPointGauge(
 )

void EPointGauge(
 float centerX,
 float centerY
 )

void EPointGauge(
 const EPointGauge& other
 )
```

Parameters

centerX

Point X coordinate.

centerY

Point Y coordinate.

other

Another EPointGauge object to be copied in the new EPointGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed point measurement context is based on a pre-existing EPointGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the EPointGauge::CopyTo method.

EPointGauge::GetMeasuredPeak

Returns information pertaining to the derivative peak associated with the specified edge-crossing point, such as its area, amplitude, start, length and center.

Namespace: Euresys::Open_eVision

[C++]

```
EPeak GetMeasuredPeak(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the edge-crossing point along the probed line segment, between 0 and EPointGauge::NumMeasuredPoints (excluded).

Remarks

If index is left unchanged from its default value (i.e. ~0 = 0xFFFFFFFF), the peak associated to the default edge-crossing point is inspected. This default point is chosen according to the transition choice parameter that is managed by the EPointGauge::TransitionChoice property.

Note. For this method to succeed, it is necessary to previously call EPointGauge::Measure.

EPointGauge::GetMeasuredPoint

Returns the coordinates of an edge-crossing point, measured along the unique sample path of the point location gauge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetMeasuredPoint(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the edge-crossing point along the probed line segment, between 0 and EPointGauge::NumMeasuredPoints (excluded).

Remarks

These coordinates pertain to the World space; they are expressed in the reference frame to which the current EPointGauge object belongs. An EPointGauge object features only one sample path, which contrasts with the other kinds of gauges. The argument index specifies the index of the edge-crossing point that is considered along this unique sample path. If index is left unchanged from its default value (i.e. ~0= 0xFFFFFFFF), the default edge-crossing point is inspected. This default point is chosen according to the transition choice parameter that is managed by the EPointGauge::TransitionChoice property.

Note. For this method to succeed, it is necessary to previously call EPointGauge::Measure.

EPointGauge::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool daughters
 )
```

Parameters

daughters

true if the daughters gauges handles have to be considered as well.

EPointGauge::GetHVConstraint

EPointGauge::SetHVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetHVConstraint()
void SetHVConstraint(bool bHVConstraint)
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

EPointGauge::Measure

Triggers the point location or the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
void Measure(
 EROIBW8* sourceImage
 )
void Measure(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Pointer to the source image.

region

Region to use with the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EPointGauge::GetMinAmplitude

EPointGauge::SetMinAmplitude

Offset added to the Threshold when a peak is to be detected.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinAmplitude()
void SetMinAmplitude(OEV_UINT32 un32MinAmplitude)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value. To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected. When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

EPointGauge::GetMinArea

EPointGauge::SetMinArea

Minimum area value.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinArea()
void SetMinArea(OEV_UINT32 un32MinArea)
```

Remarks

A transition is detected if its derivative peak reaches Threshold + MinAmplitude value, and then declared valid if the area between the peak curve and the horizontal at level Threshold reaches the MinArea value.

EPointGauge::GetNumMeasuredPoints

Number of edge-crossing points along the point location gauge.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumMeasuredPoints()
```

EPointGauge::operator=

Copies all the data from another EPointGauge object into the current EPointGauge object

Namespace: Euresys::Open_eVision

[C++]

```
EPointGauge& operator=(
 const EPointGauge& other
 )
```

Parameters

other

EPointGauge object to be copied

EPointGauge::Plot

Draws the profile that is crossed by a point location gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void Plot(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

```
void Plot(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
void Plot(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.

height
Height of the plot.

originX
Origin point coordinates of the plot along the X axis.

originY
Origin point coordinates of the plot along the Y axis.

color
The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPointGauge::PlotWithCurrentPen

This method is deprecated.

Draws the profile that is crossed by a point location gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void PlotWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of EPlotItem members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPointGauge::Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

Namespace: Euresys::Open_eVision

[C++]

```
void Process(
 EROIBW8* sourceImage,
 bool daughters
 )

void Process(
 EROIBW8* sourceImage,
 const ERegion& region,
 bool daughters
 )
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

region

Region to use with the source image.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EPointGauge::GetRectangularSamplingArea

EPointGauge::SetRectangularSamplingArea

Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRectangularSamplingArea()

void SetRectangularSamplingArea(bool bRectangularSamplingArea)
```

Remarks

By default, this flag is set to true: the sampling area always remains a rectangle. Setting this property is only useful when the thickness transition parameter is greater than 1. In fact, when thickness transition parameter is equal to 1, rectangle and parallelogram reduce to the same segment.

EPointGauge::SetCenterXY

Sets the center coordinates of a EPointGauge object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the EPointGauge object.

centerY

Center coordinates of the EPointGauge object.

EPointGauge::SetTolerances

Sets the half length and the rotation angle of the point location gauge.

Namespace: Euresys::Open_eVision

```
[C++]
void SetTolerances(
 float tolerance,
 float angle
 )
```

Parameters

tolerance

Half length of the point location gauge. The default value is 10.

angle

Rotation angle of the point location gauge. The default value is 0.

Remarks

By default, the point location gauge length value is 20 (2x10), which means 20 pixels when the field of view is not calibrated and 20 "units" in case of a calibrated field of view. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EPointGauge::GetSmoothing

EPointGauge::SetSmoothing

Number of pixels used for the low-pass filtering operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetSmoothing()
void SetSmoothing(OEV_UINT32 un32Smoothing)
```

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

EPointGauge::GetThickness

EPointGauge::SetThickness

Number of parallel segments used to extract the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThickness()
void SetThickness(OEV_UINT32 un32Thickness)
```

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

EPointGauge::GetThreshold

EPointGauge::SetThreshold

Threshold level used to delimit significant peaks in the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 un32Threshold)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value. To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected. When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

EPointGauge::GetTolerance

EPointGauge::SetTolerance

Half length of the point location gauge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetTolerance()
void SetTolerance(float tolerance)
```

Remarks

By default, the length of the point location gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

EPointGauge::GetToleranceAngle

EPointGauge::SetToleranceAngle

Rotation angle of the point location gauge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetToleranceAngle()
void SetToleranceAngle(float toleranceAngle)
```

Remarks

By default, the rotation angle of the point location gauge is 0. The sign of the rotation angle depends whether the field of view is calibrated or not. * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value. * When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EPointGauge::GetTransitionChoice

EPointGauge::SetTransitionChoice

Transition choice.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionChoice GetTransitionChoice()
void SetTransitionChoice(Euresys::Open_eVision::ETransitionChoice eTransitionChoice)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. In case of ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd transition choice, set EPointGauge::TransitionIndex to specify the desired transition. By default, the selected transition corresponds to the one with the largest amplitude (ETransitionChoice_LargestAmplitude).

EPointGauge::GetTransitionIndex

EPointGauge::SetTransitionIndex

Index (from 0 on) of the transition to be retained when the transition choice parameter is set to NthFromBegin or NthFromEnd.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTransitionIndex()
void SetTransitionIndex(OEV_UINT32 un32TransitionIndex)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition. By default, the first transition is retained (the index value is 0).

EPointGauge::GetTransitionType

EPointGauge::SetTransitionType

Transition type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionType GetTransitionType()
void SetTransitionType(Euresys::Open_eVision::ETransitionType eTransitionType)
```

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object. By default, the searched transition type is indifferently a black to white or a white to black transition (ETransitionType_BwOrWb).

EPointGauge::GetType

Shape type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EShapeType GetType()
```

EPointGauge::GetValid

Flag indicating if at least one valid transition has been found.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetValid()
```

Remarks

A false value means that no measurement has been performed. A true value means that a transition was found along the sample path defined by the EPointGauge, and thus a point was measured.

4.194. EPointShape Class

Manages a point shape context.

Base Class:EShape
Derived Class(es):EPointGauge
Namespace: Euresys::Open_eVision

Methods

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copies all the data of the current EPointShape object into another EPointShape object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
GetCenter	Center coordinates of a EPointShape object.

GetCenterX	Abscissa of the origin point of the frame.
GetCenterY	Ordinate of the origin point of the frame.
GetType	Shape type.
HitTest	Checks if there is a handle under the cursor.
operator!=	Compares the instance another EPointShape object and returns true if they are not identical.
operator=	Copies all the data from another EPointShape object into the current EPointShape object
operator==	Compares the instance another EPointShape object and returns true if they are identical.
SetCenter	Center coordinates of a EPointShape object.
SetCenterXY	Sets the center coordinates of a EPointShape object.

EPointShape::GetCenter

EPointShape::SetCenter

Center coordinates of a EPointShape object.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& point)
```

EPointShape::GetCenterX

Abscissa of the origin point of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

EPointShape::GetCenterY

Ordinate of the origin point of the frame.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

EPointShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

EPointShape::CopyTo

Copies all the data of the current EPointShape object into another EPointShape object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EPointShape& other,
 bool recursive
 )

EPointShape* CopyTo(
 EPointShape* other,
 bool recursive
 )
```

Parameters

other

Pointer to the EPointShape object in which the current EPointShape object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EPointShape object will be created and returned.

EPointShape::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

-

n32CursorY

-

EPointShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPointShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPointShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

EPointShape::operator!=

Compares the instance another EPointShape object and returns true if they are not identical.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const EPointShape& other
 )
```

Parameters

other

EPointShape object to be compared

EPointShape::operator=

Copies all the data from another EPointShape object into the current EPointShape object

Namespace: Euresys::Open_eVision

```
[C++]
EPointShape& operator=(
 const EPointShape& other
 )
```

Parameters

other

EPointShape object to be copied

EPointShape::operator==

Compares the instance another EPointShape object and returns true if they are identical.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const EPointShape& other
 )
```

Parameters

other

EPointShape object to be compared

EPointShape::SetCenterXY

Sets the center coordinates of a EPointShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the EPointShape object.

centerY

Center coordinates of the EPointShape object.

EPointShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

4.195. EPolygon Class

Represents a polygon. The number of vertices is arbitrary. A EPolygon could be closed (last and first point connected) or open (like a polyline shape).

Base Class:EFrame

Namespace: Euresys::Open_eVision

Methods

AppendVertex	Appends a polygon vertex after the last.
CopyTo	Copies all the data of the current EPolygon object into another EPolygon object, and returns it.
EPolygon	Constructs a EPolygon
GetArea	Returns the area of the polygon.
GetIsClosed	The polygon close attribute.
GetLength	Returns the length (perimeter) of the polygon.
GetNumEdges	Returns the number of polygon's edges (or sides). Depends on the EPolygon::IsClosed status.
GetNumVertices	Returns the number of polygon's vertices.
GetVertex	Gets a polygon vertex.
GetVertices	Returns the list of polygon vertices
InsertVertex	Inserts a vertex before the given index.
IsValid	Returns true if the polygon is valid. A valid polygon has at least 2 (when open) or 3 (when closed) distinct and valid vertices and no duplicate points.
Load	Loads the polygon.
operator!=	Compares the current EPolygon with another EPolygon object.
operator=	Copies all the data from another EPolygon object into the current EPolygon object
operator==	Compares the current EPolygon with another EPolygon object.
RemoveVertex	Removes vertex at the given index.
Save	Saves the polygon.
SetIsClosed	The polygon close attribute.
SetVertex	Sets a polygon vertex.

EPolygon::AppendVertex

Appends a polygon vertex after the last.

Namespace: Euresys::Open_eVision

[C++]

```
void AppendVertex(
 const EPoint& pt
 )
```

Parameters

pt

The point to append.

EPolygon::GetArea

Returns the area of the polygon.

Namespace: Euresys::Open_eVision

[C++]

```
float GetArea() const
```

EPolygon::CopyTo

Copies all the data of the current EPolygon object into another EPolygon object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EPolygon& other
 )

EPolygon* CopyTo(
 EPolygon* other
 )
```

Parameters

other

Pointer to the EPolygon object in which the current EPolygon object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EPolygon object will be created and returned.

EPolygon::EPolygon

Constructs a EPolygon

Namespace: Euresys::Open_eVision

[C++]

```
void EPolygon(
 )
```

```
void EPolygon(
 const EPolygon& other
 )
void EPolygon(
 const std::vector<Euresys::Open_eVision::EPoint>& vertices,
 bool closed
 )
```

Parameters

other

Reference to the EPolygon used for the initialization.

vertices

A vector of at least 3 points.

closed

An optional parameter to set the closed properties. When closed, an edge exists between the last and first points of the polygon. By default, created polygons are closed.

EPolygon::GetVertex

Gets a polygon vertex.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetVertex(
 int index
 )
```

Parameters

index

The index of the vertex.

EPolygon::InsertVertex

Inserts a vertex before the given index.

Namespace: Euresys::Open_eVision

[C++]

```
void InsertVertex(
 int index,
 const EPoint& pt
 )
```

Parameters

index

The index after which the vertex will be inserted.

pt

The point to insert.

EPolygon::GetIsClosed

EPolygon::SetIsClosed

The polygon close attribute.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetIsClosed() const
void SetIsClosed(bool state)
```

EPolygon::IsValid

Returns true if the polygon is valid. A valid polygon has at least 2 (when open) or 3 (when closed) distinct and valid vertices and no duplicate points.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsValid(
 )
```

EPolygon::GetLength

Returns the length (perimeter) of the polygon.

Namespace: Euresys::Open_eVision

[C++]

```
float GetLength() const
```

EPolygon::Load

Loads the polygon.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& file
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

file

File path.

serializer

Serializer. Must be in read mode.

EPolygon::GetNumEdges

Returns the number of polygon's edges (or sides). Depends on the EPolygon::IsClosed status.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumEdges() const
```

EPolygon::GetNumVertices

Returns the number of polygon's vertices.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumVertices() const
```

EPolygon::operator!=

Compares the current EPolygon with another EPolygon object.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EPolygon& polygon
 )
```

Parameters

polygon

The other EPolygon object.

EPolygon::operator=

Copies all the data from another EPolygon object into the current EPolygon object

Namespace: Euresys::Open_eVision

```
[C++]

EPolygon& operator=(
 const EPolygon& other
 )
```

Parameters

other

EPolygon object to be copied

EPolygon::operator==

Compares the current EPolygon with another EPolygon object.

Namespace: Euresys::Open_eVision

```
[C++]

bool operator==(
 const EPolygon& polygon
 )
```

Parameters

polygon

The other EPolygon object.

EPolygon::RemoveVertex

Removes vertex at the given index.

Namespace: Euresys::Open_eVision

```
[C++]

void RemoveVertex(
 int index
 )
```

Parameters

index

The index of the vertex to remove.

EPolygon::Save

Saves the polygon.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& file
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

file

File path.

serializer

Serializer. Must be in write mode.

EPolygon::SetVertex

Sets a polygon vertex.

Namespace: Euresys::Open_eVision

[C++]

```
void SetVertex(
 int index,
 const EPoint& pt
 )
```

Parameters

index

The index of the vertex.

pt

The vertex position.

EPolygon::GetVertices

Returns the list of polygon vertices

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::EPoint> GetVertices() const
```

4.196. EPolygonGauge Class

Manages a polygon fitting gauge.

Base Class:EPolygonShape

Namespace: Euresys::Open_eVision

Methods

AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.
CopyTo	Copies all the data of the current EPolygonGauge object into another EPolygonGauge object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
EPolygonGauge	Constructs a polygon measurement context.
GetAverageDistance	Average distance between the sampled points and the fitted model.
GetEnableScaling	When using the EPolygonMeasurementMode_Global mode, if EPolygonGauge::EnableScaling is enable then the best scale is evaluated.
GetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
GetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
GetMeasuredPolygon	Returns the measured polygon (if found) or the nominal polygon otherwise.
GetMeasurementMode	The polygon gauge measurement mode, in EPolygonMeasurementMode. The default mode is EPolygonMeasurementMode_Global.
GetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
GetMinArea	Minimum area value.
GetMinNumFitSamples	Sets the minimum number of samples required for fitting on each edges of the polygon.

GetNumFilteringPasses	Number of filtering passes for a model fitting operation.
GetNumSamples	Number of sampled points during the model fitting operation. The samples are distributed evenly on the polygon.
GetNumSkipRanges	Number of skip ranges in the gauge after a call to EPolygonGauge::AddSkipRange.
GetNumValidSamples	Number of valid sample points remaining after a model fitting operation.
GetSamplingStep	Approximate distance between sampled points during a model fitting operation.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the EPolygonGauge::AddSkipRange method.
GetSmoothing	Number of pixels used for the low-pass filtering operation.
GetThickness	Number of parallel segments used to extract the data profile.
GetThreshold	Threshold level used to delimit significant peaks in the data profile.
GetTolerance	Searching area half thickness of the polygon fitting gauge.
GetTransitionChoice	Transition choice.
GetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
GetTransitionType	Transition type.
GetType	Shape type.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
operator=	Copies all the data from another EPolygonGauge object into the current EPolygonGauge object
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.
RemoveAllSkipRanges	Removes all the skip ranges previously created by a call to EPolygonGauge::AddSkipRange.
RemoveSkipRange	After a call to EPolygonGauge::AddSkipRange, removes the skip range with the given index.
SetActive	Sets the flag indicating whether the gauge is active or not.
SetEnableScaling	When using the EPolygonMeasurementMode_Global mode, if EPolygonGauge::EnableScaling is enable then the best scale is evaluated.
SetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

SetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
SetMeasurementMode	The polygon gauge measurement mode, in EPolygonMeasurementMode. The default mode is EPolygonMeasurementMode_Global.
SetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
SetMinArea	Minimum area value.
SetMinNumFitSamples	Sets the minimum number of samples required for fitting on each edges of the polygon.
SetNumFilteringPasses	Number of filtering passes for a model fitting operation.
SetSamplingStep	Approximate distance between sampled points during a model fitting operation.
SetSmoothing	Number of pixels used for the low-pass filtering operation.
SetThickness	Number of parallel segments used to extract the data profile.
SetThreshold	Threshold level used to delimit significant peaks in the data profile.
SetTolerance	Searching area half thickness of the polygon fitting gauge.
SetTransitionChoice	Transition choice.
SetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
SetTransitionType	Transition type.

EPolygonGauge::SetActive

Sets the flag indicating whether the gauge is active or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetActive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying EPolygonGauge::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (true).

EPolygonGauge::AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 AddSkipRange(
 const OEV_UINT32 start,
 const OEV_UINT32 end
 )
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process.
The EPolygonGauge::AddSkipRange method allows to define skip ranges in an EPolygonGauge. This means that, at measure time, samples belonging to these ranges will not be taken into account.
A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another.
The range is allowed to be reversed (i.e. end is not required to be greater than start).
Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for EPolygonGauge::NumSamples).

EPolygonGauge::GetAverageDistance

Average distance between the sampled points and the fitted model.

Namespace: Euresys::Open_eVision

```
[C++]
float GetAverageDistance()
```

Remarks

Irrelevant in case of a point location operation.

EPolygonGauge::CopyTo

Copies all the data of the current EPolygonGauge object into another EPolygonGauge object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EPolygonGauge& other,
 bool recursive
 )

EPolygonGauge* CopyTo(
 EPolygonGauge* other,
 bool recursive
 )
```

Parameters

other

Pointer to the EPolygonGauge object in which the current EPolygonGauge object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EPolygonGauge object will be created and returned.

EPolygonGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y
 )
```

Parameters

x

Cursor current X coordinate.

y

Cursor current Y coordinate.

EPolygonGauge::Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC hdc,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC hdc,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

hdc

-

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPolygonGauge::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC hdc,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

hdc

-

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPolygonGauge::GetEnableScaling

EPolygonGauge::SetEnableScaling

When using the Global mode, if EPolygonGauge::EnableScaling is enable then the best scale is evaluated.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableScaling() const

void SetEnableScaling(bool state)
```

EPolygonGauge::EPolygonGauge

Constructs a polygon measurement context.

Namespace: Euresys::Open_eVision

[C++]

```
void EPolygonGauge(
 )

void EPolygonGauge(
 const EPolygonGauge& other
 )
```

Parameters

other

Another EPolygonGauge object to be copied in the new EPolygonGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values.
With the copy constructor, the constructed polygon measurement context is based on a pre-existing EPolygonGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the EPolygonGauge::CopyTo method.

EPolygonGauge::GetFilteringThreshold

EPolygonGauge::SetFilteringThreshold

Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFilteringThreshold()

void SetFilteringThreshold(float f32FilteringThreshold)
```

Remarks

Irrelevant in case of a point location operation.
During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).

EPolygonGauge::GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the EPolygonGauge::AddSkipRange method.

Namespace: Euresys::Open_eVision

[C++]

```
void GetSkipRange(
 const OEV_UINT32 index,
 OEV_UINT32& start,
 OEV_UINT32& end
 )
```

Parameters

index
Index of the skip range.

start
Beginning of the skip range.

end
End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

EPolygonGauge::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 bool daughters
 )
```

Parameters

daughters
true if the daughters gauges handles have to be considered as well.

EPolygonGauge::GetHVConstraint

EPolygonGauge::SetHVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetHVConstraint()

void SetHVConstraint(bool bHVConstraint)
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

EPolygonGauge::Measure

Triggers the point location or the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
void Measure(
 EROIBW8* sourceImage
 )
void Measure(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Pointer to the source image.

region

Region to use with the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EPolygonGauge::GetMeasuredPolygon

Returns the measured polygon (if found) or the nominal polygon otherwise.

Namespace: Euresys::Open_eVision

```
[C++]
EPolygon GetMeasuredPolygon()
```

Remarks

Use method EShape::GetFound to get the status of the measurement.
EPolygonGauge::MeasuredPolygon returns a successful fitted polygon if EShape::GetFound is true, otherwise it returns the orginal (nominal) polygon.

EPolygonGauge::GetMeasurementMode

EPolygonGauge::SetMeasurementMode

The polygon gauge measurement mode, in EPolygonMeasurementMode. The default mode is Global.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EPolygonMeasurementMode GetMeasurementMode() const
```

```
void SetMeasurementMode(Euresys::Open_eVision::EPolygonMeasurementMode mode)
```

EPolygonGauge::MeasureSample

Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureSample(
 EROIBW8* image,
 int edgeIndex,
 int sampleIndex
 )

void MeasureSample(
 EROIBW8* image,
 const ERegion& region,
 int edgeIndex,
 int sampleIndex
 )
```

Parameters

image

-

edgeIndex

-

sampleIndex

-

region

Region on which to measure.

Remarks

This method stores its results into a temporary variable inside the EPolygonGauge object.

EPolygonGauge::GetMinAmplitude

EPolygonGauge::SetMinAmplitude

Offset added to the Threshold when a peak is to be detected.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinAmplitude()
```

```
void SetMinAmplitude(OEV_UINT32 un32MinAmplitude)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value.
To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected.
When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

EPolygonGauge::GetMinArea

EPolygonGauge::SetMinArea

Minimum area value.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinArea()
void SetMinArea(OEV_UINT32 un32MinArea)
```

Remarks

A transition is detected if its derivative peak reaches Threshold + MinAmplitude value, and then declared valid if the area between the peak curve and the horizontal at level Threshold reaches the MinArea value.

EPolygonGauge::GetMinNumFitSamples

EPolygonGauge::SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each edges of the polygon.

Namespace: Euresys::Open_eVision

[C++]

```
int GetMinNumFitSamples()
void SetMinNumFitSamples(int side)
```

Remarks

When the EPolygonGauge::Measure method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EPolygonGauge::GetNumFilteringPasses

EPolygonGauge::SetNumFilteringPasses

Number of filtering passes for a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumFilteringPasses()

void SetNumFilteringPasses(OEV_UINT32 un32NumFilteringPasses)
```

Remarks

Irrelevant in case of a point location operation.
During a filtering pass, the points that are too distant from the model are discarded.
During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).
By default (the number of filtering passes is 0), the outliers rejection process is disabled.

EPolygonGauge::GetNumSamples

Number of sampled points during the model fitting operation. The samples are distributed evenly on the polygon.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamples() const
```

Remarks

After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

EPolygonGauge::GetNumSkipRanges

Number of skip ranges in the gauge after a call to EPolygonGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSkipRanges() const
```

EPolygonGauge::GetNumValidSamples

Number of valid sample points remaining after a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumValidSamples()
```

Remarks

Irrelevant in case of a point location operation.
After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

EPolygonGauge::operator=

Copies all the data from another EPolygonGauge object into the current EPolygonGauge object

Namespace: Euresys::Open_eVision

```
[C++]
EPolygonGauge& operator=(
 const EPolygonGauge& other
 )
```

Parameters

other

EPolygonGauge object to be copied

EPolygonGauge::Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

Namespace: Euresys::Open_eVision

```
[C++]
void Process(
 EROIBW8* sourceImage,
 bool daughters
 )

void Process(
 EROIBW8* sourceImage,
 const ERegion& region,
 bool daughters
 )
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

region

Region to use with the source image.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EPolygonGauge::RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to EPolygonGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveAllSkipRanges(
 )
```

EPolygonGauge::RemoveSkipRange

After a call to EPolygonGauge::AddSkipRange, removes the skip range with the given index.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveSkipRange(
 const OEV_UINT32 index
 )
```

Parameters

index

Index of the skip range to remove, as returned by EPolygonGauge::AddSkipRange.

EPolygonGauge::GetSamplingStep

EPolygonGauge::SetSamplingStep

Approximate distance between sampled points during a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSamplingStep() const
void SetSamplingStep(float f32SamplingStep)
```

Remarks

Irrelevant in case of a point location operation.
To fit a model, a series of point location operations are performed along the model. The point location gauges spacing is given by the sampling step.
By default, the sampling step is set to 5, which means 5 pixels when the field of view is not calibrated, and 5 physical units in case of a calibrated field of view.
Be aware that if the sampling step is too large, the number of sampled points along the model will not be sufficient enough to accurately locate it.

EPolygonGauge::GetSmoothing

EPolygonGauge::SetSmoothing

Number of pixels used for the low-pass filtering operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetSmoothing()
void SetSmoothing(OEV_UINT32 un32Smoothing)
```

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

EPolygonGauge::GetThickness

EPolygonGauge::SetThickness

Number of parallel segments used to extract the data profile.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetThickness()
void SetThickness(OEV_UINT32 un32Thickness)
```

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

EPolygonGauge::GetThreshold

EPolygonGauge::SetThreshold

Threshold level used to delimit significant peaks in the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 un32Threshold)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value.
To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected.
When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

EPolygonGauge::GetTolerance

EPolygonGauge::SetTolerance

Searching area half thickness of the polygon fitting gauge.

Namespace: Euresys::Open_eVision

```
[C++]
float GetTolerance()
void SetTolerance(float f32Tolerance)
```

Remarks

A polygon fitting gauge is fully defined knowing its nominal positions (vertices), its nominal origin, angle and scale, and its outline tolerance.
By default, the searching area thickness of the polygon fitting gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

EPolygonGauge::GetTransitionChoice

EPolygonGauge::SetTransitionChoice

Transition choice.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionChoice GetTransitionChoice()
void SetTransitionChoice(Euresys::Open_eVision::ETransitionChoice eTransitionChoice)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition.
In case of ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd transition choice, set EPolygonGauge::TransitionIndex to specify the desired transition.
By default, the selected transition corresponds to the one with the largest amplitude (ETransitionChoice_LargestAmplitude).

EPolygonGauge::GetTransitionIndex

EPolygonGauge::SetTransitionIndex

Index (from 0 on) of the transition to be retained when the transition choice parameter is set to NthFromBegin or NthFromEnd.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTransitionIndex()
void SetTransitionIndex(OEV_UINT32 un32TransitionIndex)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition.
By default, the first transition is retained (the index value is 0).

EPolygonGauge::GetTransitionType

EPolygonGauge::SetTransitionType

Transition type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionType GetTransitionType()
void SetTransitionType(Euresys::Open_eVision::ETransitionType eTransitionType)
```

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object.
By default, the searched transition type is indifferently a black to white or a white to black transition (ETransitionType_BwOrWb).

EPolygonGauge::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

4.197. EPolygonRegion Class

Manages a complete context for an ERegion shaped like a polygon.

Base Class:ERegion
Namespace: Euresys::Open_eVision

Methods

AddPoint	Add a new vertex at the end of the region
Drag	Moves the specified handle to a new position and updates all placement parameters of the region.
EPolygonRegion	Constructs an EPolygonRegion context.
GetNumPoints	Number of vertices of the region.
GetPoints	List of vertices of the region
HitTest	Detects if the cursor is placed over one of the dragging handles.
InsertPoint	Insert a vertice between two existing ones
Load	Loads the EPolygonRegion. The given ESerializer must have been created for reading.
operator!=	Checks if this EPolygonRegion instance is not strictly equal to another
operator=	Assignment operator.
operator==	Checks if this EPolygonRegion instance is strictly equal to another

RemovePoint	Remove a vertex
Rotate	Creates a new EPolygonRegion by rotating the EPolygonRegion.
Save	Saves the EPolygonRegion. The given ESerializer must have been created for writing.
Scale	Creates a new EPolygonRegion by scaling the region.
SetPoints	List of vertices of the region
Translate	Creates a new EPolygonRegion by translating the EPolygonRegion.

EPolygonRegion::AddPoint

Add a new vertex at the end of the region

Namespace: Euresys::Open_eVision

[C++]

```
void AddPoint(
 const EPoint& point
 )
```

Parameters

point

-

EPolygonRegion::Drag

Moves the specified handle to a new position and updates all placement parameters of the region.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
x-coordinate of the mouse cursor.

y
y-coordinate of the mouse cursor.

zoomX
Horizontal zoom factor. By default, true scale is used.

zoomY
Vertical zoom factor. By default, true scale is used.

panX
Horizontal pan offset. By default, no pan is added.

panY
Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with EPolygonRegion::HitTest and EPolygonRegion::Drag.

EPolygonRegion::EPolygonRegion

Constructs an EPolygonRegion context.

Namespace: Euresys::Open_eVision

[C++]

```
void EPolygonRegion(
 )

void EPolygonRegion(
 const std::vector<Euresys::Open_eVision::EPoint>& points
 )

void EPolygonRegion(
 const EPolygonRegion& other
 )
```

Parameters

points
The list of vertices of the EPolygonRegion.

other
EPolygonRegion context to copy.

EPolygonRegion::HitTest

Detects if the cursor is placed over one of the dragging handles.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EEditionMode HitTest(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal pan offset. By default, no pan is added.

panY

Vertical pan offset. By default, no pan is added.

Remarks

Returns a handle identifier, as defined by EEditionMode.
If zooming and/or panning were used when drawing the region, the same values must be used with EPolygonRegion::HitTest and EPolygonRegion::Drag.

EPolygonRegion::InsertPoint

Insert a vertice between two existing ones

Namespace: Euresys::Open_eVision

[C++]

```
bool InsertPoint(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
x-coordinate of the mouse cursor.

y
y-coordinate of the mouse cursor.

zoomX
Horizontal zoom factor. By default, true scale is used.

zoomY
Vertical zoom factor. By default, true scale is used.

panX
Horizontal pan offset. By default, no pan is added.

panY
Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with EPolygonRegion::HitTest and EPolygonRegion::Drag.

EPolygonRegion::Load

Loads the EPolygonRegion. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EPolygonRegion::GetNumPoints

Number of vertices of the region.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumPoints() const
```

EPolygonRegion::operator!=

Checks if this EPolygonRegion instance is not strictly equal to another

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EPolygonRegion& other
 )
```

Parameters

other

Reference to the other EPolygonRegion instance

EPolygonRegion::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EPolygonRegion& operator=(
 const EPolygonRegion& other
 )
```

Parameters

other

Reference to the EPolygonRegion used for the assignment

EPolygonRegion::operator==

Checks if this EPolygonRegion instance is strictly equal to another

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const EPolygonRegion& other
 )
```

Parameters

other

Reference to the other EPolygonRegion instance

EPolygonRegion::GetPoints

EPolygonRegion::SetPoints

List of vertices of the region

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EPoint> GetPoints() const
void SetPoints(const std::vector<Euresys::Open_eVision::EPoint>& points)
```

EPolygonRegion::RemovePoint

Remove a vertex

Namespace: Euresys::Open_eVision

[C++]

```
bool RemovePoint(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal pan offset. By default, no pan is added.

panY

Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with EPolygonRegion::HitTest and EPolygonRegion::Drag.

EPolygonRegion::Rotate

Creates a new EPolygonRegion by rotating the EPolygonRegion.

Namespace: Euresys::Open_eVision

[C++]

```
EPolygonRegion Rotate(
 float angle
 )
```

Parameters

angle

rotation angle

EPolygonRegion::Save

Saves the EPolygonRegion. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EPolygonRegion::Scale

Creates a new EPolygonRegion by scaling the region.

Namespace: Euresys::Open_eVision

[C++]

```
EPolygonRegion Scale(
 float scale
 )
EPolygonRegion Scale(
 float scaleX,
 float scaleY
 )
```

Parameters

scale
Isotropic scale
scaleX
Horizontal scale
scaleY
Vertical scale

EPolygonRegion::Translate

Creates a new EPolygonRegion by translating the EPolygonRegion.

Namespace: Euresys::Open_eVision

[C++]

```
EPolygonRegion Translate(
 float dx,
 float dy
 )
```

Parameters

dx
Horizontal translation in pixel value
dy
Vertical translation in pixel value

4.198. EPolygonShape Class

Manages a polygonal shape.

Base Class:EShape
Derived Class(es):EPolygonGauge
Namespace: Euresys::Open_eVision

Methods

AddVertexAtDisplayPosition	Add a vertex to the polygon at a position depending on a display coordinate. The current display parameters (zoom and pan) are used to find the best insert position.
AppendVertex	Append a polygon vertex after the last.
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copies all the data of the current EPolygonShape object into another EPolygonShape object and returns it.
DisplayToLocalPosition	-
Drag	Moves a handle to a new position and updates the position parameters of the shape.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
EPolygonShape	Constructs a EPolygonShape
GetAngle	Orientation of the shape.
GetCenter	Center point of the shape.
GetCenterX	Abscissa of the origin point of the shape.
GetCenterY	Ordinate of the origin point of the shape.
GetFormat	-
GetIsClosed	The polygon close attribute.
GetLength	Returns the length (perimeter) of the polygon
GetNumEdges	Returns the number of polygon's edges (or sides). Depends on the EPolygon::IsClosed status.
GetNumVertices	Returns the number of polygon's vertices.
GetPolygon	Sets/Gets the polygon according to a known EPolygon object.
GetScale	Scale of the polygon.
GetType	Shape type.
GetVertex	A polygon vertex.
HitTest	Checks if there is a handle under the cursor.
LocalToDisplayPosition	-
operator=	Copies all the data from another EPolygonShape object into the current EPolygonShape object
RemoveVertexAtDisplayPosition	Remove the closest vertex to the display position (x,y)
SerializeData	-

SetAngle	Orientation of the shape.
SetCenter	Center point of the shape.
SetCenterXY	Sets the center coordinates of a EPolygonShape object.
SetIsClosed	The polygon close attribute.
SetPolygon	Sets/Gets the polygon according to a known EPolygon object.
SetScale	Scale of the polygon.
SetVertex	-

EPolygonShape::AddVertexAtDisplayPosition

Add a vertex to the polygon at a position depending on a display coordinate. The current display parameters (zoom and pan) are used to find the best insert position.

Namespace: Euresys::Open_eVision

[C++]

```
void AddVertexAtDisplayPosition(
 int x,
 int y
 )
```

Parameters

x

The X coordinate in display space

y

The Y coordinate in display space

EPolygonShape::GetAngle

EPolygonShape::SetAngle

Orientation of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const

void SetAngle(float f32Angle)
```

EPolygonShape::AppendVertex

Append a polygon vertex after the last.

Namespace: Euresys::Open_eVision

```
[C++]
void AppendVertex(
 const EPoint& pt
 )
```

Parameters

pt

The point to append.

EPolygonShape::GetCenter

EPolygonShape::SetCenter

Center point of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

EPolygonShape::GetCenterX

Abscissa of the origin point of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCenterX() const
```

EPolygonShape::GetCenterY

Ordinate of the origin point of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCenterY() const
```

EPolygonShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

```
[C++]
void Closest(
 )
```

EPolygonShape::CopyTo

Copies all the data of the current EPolygonShape object into another EPolygonShape object and returns it.

Namespace: Euresys::Open_eVision

```
[C++]
void CopyTo(
 EPolygonShape& other,
 bool bRecursive
 )
EPolygonShape* CopyTo(
 EPolygonShape* dest,
 bool bRecursive
 )
```

Parameters

other

-

bRecursive

true if the children shapes have to be copied as well, false otherwise.

dest

Pointer to the EPolygonShape object in which the current EPolygonShape object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EPolygonShape object will be created and returned.

EPolygonShape::DisplayToLocalPosition

-

Namespace: Euresys::Open_eVision

[C++]

```
EPoint DisplayToLocalPosition(
 int x,
 int y
 )
```

Parameters

x

-

y

-

EPolygonShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

EPolygonShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

```
void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPolygonShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EPolygonShape::EPolygonShape

Constructs a EPolygonShape

Namespace: Euresys::Open_eVision

[C++]

```
void EPolygonShape(
 )

void EPolygonShape(
 const EPolygonShape& other
 )

void EPolygonShape(
 const std::vector<Euresys::Open_eVision::EPoint>& vertices,
 bool closed
 )
```

Parameters

other

Reference to the EPolygonShape used for the initialization.

vertices

A vector of at least 3 points.

closed

An optional parameter to set the closed properties. When closed, an edge exists between the last and first points of the polygon. By default, created polygons are closed.

EPolygonShape::GetFormat

-

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetFormat()
```

EPolygonShape::GetVertex

A polygon vertex.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetVertex(
 int index
 )
```

Parameters

index

The index of the vertex.

EPolygonShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

EPolygonShape::GetIsClosed

EPolygonShape::SetIsClosed

The polygon close attribute.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetIsClosed() const

void SetIsClosed(bool state)
```

EPolygonShape::GetLength

Returns the length (perimeter) of the polygon

Namespace: Euresys::Open_eVision

[C++]

```
float GetLength() const
```

EPolygonShape::LocalToDisplayPosition

-

Namespace: Euresys::Open_eVision

[C++]

```
EPoint LocalToDisplayPosition(
 const EPoint& p
 )
```

Parameters

p

-

EPolygonShape::GetNumEdges

Returns the number of polygon's edges (or sides). Depends on the EPolygon::IsClosed status.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumEdges() const
```

EPolygonShape::GetNumVertices

Returns the number of polygon's vertices.

Namespace: Euresys::Open_eVision

[C++]

```
int GetNumVertices() const
```

EPolygonShape::operator=

Copies all the data from another EPolygonShape object into the current EPolygonShape object

Namespace: Euresys::Open_eVision

```
[C++]
EPolygonShape& operator=(
 const EPolygonShape& other
 )
```

Parameters

other

EPolygonShape object to be copied

EPolygonShape::GetPolygon

EPolygonShape::SetPolygon

Sets/Gets the polygon according to a known EPolygon object.

Namespace: Euresys::Open_eVision

```
[C++]
const EPolygon& GetPolygon() const
void SetPolygon(const EPolygon& polygon)
```

EPolygonShape::RemoveVertexAtDisplayPosition

Remove the closest vertex to the display position (x,y)

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveVertexAtDisplayPosition(
 int x,
 int y
 )
```

Parameters

x

The X coordinate in display space

y

The Y coordinate in display space

EPolygonShape::GetScale

EPolygonShape::SetScale

Scale of the polygon.

Namespace: Euresys::Open_eVision

```
[C++]
float GetScale() const
void SetScale(float f32Scale)
```

EPolygonShape::SerializeData

-

Namespace: Euresys::Open_eVision

```
[C++]
void SerializeData(
 ESerializer* serializer,
 OEV_UINT32 un32FileVersion,
 bool bDaughters
 )
void SerializeData(
 ESerializer* serializer,
 OEV_UINT32 un32FileVersion
 )
```

Parameters

serializer

-

un32FileVersion

-

bDaughters

-

EPolygonShape::SetCenterXY

Sets the center coordinates of a EPolygonShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the EPolygonShape object.

centerY

Center coordinates of the EPolygonShape object.

EPolygonShape::SetVertex

-

Namespace: Euresys::Open_eVision

[C++]

```
void SetVertex(
 int index,
 const EPoint& pt
 )
```

Parameters

index

-

pt

-

EPolygonShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

4.199. EPrincipalAxisExtractor Class

A EPrincipalAxisExtractor object computes the principal axis analysis (PCA) on an EPointCloud and produces a E3DTransformMatrix as a result.

Namespace: Euresys::Open_eVision::Easy3D

Methods

EPrincipalAxisExtractor	Creates an EPrincipalAxisExtractor object.
Extract	Computes the E3DTransformMatrix for a given EPointCloud. This will compute the PCA, then select the direction of each axis of the basis so that this basis is as close as possible as the reference transform. Optionally returns the standard deviation along the 3 axis.
GetReferenceTransform	Sets/Gets the reference transform (E3DTransformMatrix). If not set, it will use the main basis as reference to select the direction of each axis of the new basis.
HasReferenceTransformSet	Returns 'true' if a reference transform (E3DTransformMatrix) has been set explicitly.
Load	Loads the EPrincipalAxisExtractor object configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the EPrincipalAxisExtractor object configuration. The given ESerializer must have been created for writing.
SetReferenceTransform	Sets/Gets the reference transform (E3DTransformMatrix). If not set, it will use the main basis as reference to select the direction of each axis of the new basis.
UnsetReferenceTransform	Unset the reference transform (E3DTransformMatrix).

EPrincipalAxisExtractor::EPrincipalAxisExtractor

Creates an EPrincipalAxisExtractor object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EPrincipalAxisExtractor(
 )

void EPrincipalAxisExtractor(
 const EPrincipalAxisExtractor& other
 )
```

Parameters

other

The object used for the initialization

EPrincipalAxisExtractor::Extract

Computes the E3DTransformMatrix for a given EPointCloud.
This will compute the PCA, then select the direction of each axis of the basis so that this basis is as close as possible as the reference transform.
Optionally returns the standard deviation along the 3 axis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix Extract(
 const EPointCloud& pc
 )

E3DTransformMatrix Extract(
 const EPointCloud& pc,
 float& stdDevX,
 float& stdDevY,
 float& stdDevZ
 )
```

Parameters

pc
Input point cloud.

stdDevX
Variable to store the X component of the standard deviation.

stdDevY
Variable to store the Y component of the standard deviation.

stdDevZ
Variable to store the Z component of the standard deviation.

EPrincipalAxisExtractor::HasReferenceTransformSet

Returns 'true' if a reference transform (E3DTransformMatrix) has been set explicitly.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool HasReferenceTransformSet(
 )
```

EPrincipalAxisExtractor::Load

Loads the EPrincipalAxisExtractor object configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EPrincipalAxisExtractor::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EPrincipalAxisExtractor& operator=(
 const EPrincipalAxisExtractor& other
 )
```

Parameters

other

The EPrincipalAxisExtractor object that should be copied.

EPrincipalAxisExtractor::GetReferenceTransform

EPrincipalAxisExtractor::SetReferenceTransform

Sets/Gets the reference transform (E3DTransformMatrix). If not set, it will use the main basis as reference to select the direction of each axis of the new basis.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DTransformMatrix GetReferenceTransform() const
void SetReferenceTransform(const E3DTransformMatrix& refTransform)
```

EPrincipalAxisExtractor::Save

Saves the EPrincipalAxisExtractor object configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EPrincipalAxisExtractor::UnsetReferenceTransform

Unset the reference transform (E3DTransformMatrix).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void UnsetReferenceTransform(
 )
```

4.200. EPseudoColorLookup Class

Describes a lookup table, that is used to for pseudo-coloring (i.e. for assigning colors to gray-level images).

Namespace: Euresys::Open_eVision

Methods

EPseudoColorLookup	Default constructor of EPseudoColorLookup objects.
SetShading	Sets up a pseudo-color mapping such that gray level 0 corresponds to color c24Black, gray level 255 corresponds to color c24White, and intermediate values are interpolated linearly between these two extremes.

EPseudoColorLookup::EPseudoColorLookup

Default constructor of EPseudoColorLookup objects.

Namespace: Euresys::Open_eVision

```
[C++]
void EPseudoColorLookup(
 )
```

EPseudoColorLookup::SetShading

Sets up a pseudo-color mapping such that gray level 0 corresponds to color c24Black, gray level 255 corresponds to color c24White, and intermediate values are interpolated linearly between these two extremes.

Namespace: Euresys::Open_eVision

```
[C++]
void SetShading(
 EC24 black,
 EC24 white,
 Euresys::Open_eVision::EColorSystem colorSystem,
 bool wrap
 )
```

Parameters

black

Color to be mapped on a black (value 0) pixel.

white

Color to be mapped on a white (value 255) pixel.

colorSystem

Color system in which interpolation takes place.

wrap

If the color system supports a hue component, indicates whether hue wrap around must be applied.

Remarks

Furthermore, interpolation is performed in the designated color system. Even though interpolation is performed in an arbitrary color system, the extreme colors are specified in the RGB space. To obtain interesting shades of colors, it is recommended to interpolate on the hue component alone.

4.201. EQRCode Class

Represents a QR code found in the search field.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the QR code using a pre-defined pen.
DrawErrors	Draws the detected errors.
DrawErrorsWithCurrentPen	Draws the detected errors using the pen currently set in the graphical context.
DrawWithCurrentPen	Draws the QR code using the pen currently set in the graphical context.
EQRCode	Creates an EQRCode object.
GetCellPosition	Position of a cell of the QR code in the Image/ROI.
GetDecodedStream	Decoded stream extracted from the QR Code, returned as an EQRCodeDecodedStream object.
GetDecodedString	String containing the concatenated decoded data of the decoded stream.
GetErrors	Retrieves the position of the errors detected in the QR code symbol.
GetGeometry	Geometry of the QR code retruned as an EQRCodeGeometry object.
GetIsDecodingReliable	Decoding reliabililty.
GetIso15415GradingParameters	ISO/IEC 15415 grading parameters
GetIso29158GradingParameters	ISO/IEC 29158 grading parameters
GetLevel	Error correction level of the QR code.
GetModel	Model of the QR code.
GetUnusedErrorCorrection	Unused error correction.
GetVersion	Version of the QR code.
IsGS1	Return true if the decoded string uses the GS1 standard.
operator=	Assignment operator.

EQRCode::GetDecodedStream

Decoded stream extracted from the QR Code, returned as an EQRCodeDecodedStream object.

Namespace: Euresys::Open_eVision

[C++]

```
const EQRCodeDecodedStream& GetDecodedStream() const
```

EQRCode::Draw

Draws the QR code using a pre-defined pen.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* drawAdapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

Draw adapter.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

hDC

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQRCode::DrawErrors

Draws the detected errors.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawErrors(
 EDrawAdapter* drawAdapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawErrors(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

Draw adapter.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

hDC

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQRCode::DrawErrorsWithCurrentPen

This method is deprecated.

Draws the detected errors using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawErrorsWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQRCode::DrawWithCurrentPen

This method is deprecated.

Draws the QR code using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQRCode::EQRCode

Creates an EQRCode object.

Namespace: Euresys::Open_eVision

[C++]

```
void EQRCode(
 )
void EQRCode(
 const EQRCode& other
 )
```

Parameters

other

Another EQRCode.

EQRCode::GetErrors

Retrieves the position of the errors detected in the QR code symbol.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EMatrixPosition> GetErrors() const
```

EQRCode::GetGeometry

Geometry of the QR code retruned as an EQRCodeGeometry object.

Namespace: Euresys::Open_eVision

[C++]

```
const EQRCodeGeometry& GetGeometry() const
```

Remarks

The geometry of an EQRCode objects is described by its position and by the position of its finder pattern centers.

EQRCode::GetCellPosition

Position of a cell of the QR code in the Image/ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EQuadrangle GetCellPosition(
 int x,
 int y
 )

EQuadrangle GetCellPosition(
 EMatrixPosition position
 )
```

Parameters

x

The horizontal index of the cell in the QR code symbol.

y

The vertical index of the cell in the QR code symbol.

position

The position of the cell in the QR code symbol.

EQRCode::GetDecodedString

String containing the concatenated decoded data of the decoded stream.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetDecodedString(
 )

std::string GetDecodedString(
 Euresys::Open_eVision::EByteInterpretationMode byteInterpretationMode
 )
```

Parameters

byteInterpretationMode

The EByteInterpretationMode that should be used to interpret bytes.

Remarks

No parameter is required if no bytes are encoded or if the ECI byte encoding mode is supported.
Exception will be thrown if a parameter is required or if a wrong one is used.

The EByteInterpretationMode_Hexadecimal parameter throws no exception and will return the bytes hexadecimal values wrapped between '0xEFBFBD'.
This mode overrides the ECI mode.
Sample : RC -> EFBFBD + RC + EFBFBD -> EFBFBD5243EFBFBD
Note: This method has currently limitations in .NET for byte encoded parts. A workaround is the conversion of the string to bytes then to UTF8.
See EByteInterpretationMode for more options.

EQRCode::GetIsDecodingReliable

Decoding reliabililty.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetIsDecodingReliable() const
```

EQRCode::IsGS1

Return true if the decoded string uses the GS1 standard.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsGS1(
 )
```

EQRCode::GetIso15415GradingParameters

ISO/IEC 15415 grading parameters

Namespace: Euresys::Open_eVision

[C++]

```
EQRCodeIso15415GradingParameters GetIso15415GradingParameters() const
```

Remarks

Grading will only be computed if EQRCodeReader::ComputeGrading is set to true.

EQRCode::GetIso29158GradingParameters

ISO/IEC 29158 grading parameters

Namespace: Euresys::Open_eVision

```
[C++]
EQRCodeIso29158GradingParameters GetIso29158GradingParameters() const
```

Remarks

Grading will only be computed if EQRCodeReader::ComputeGrading is set to true.

EQRCode::GetLevel

Error correction level of the QR code.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EQRCodeLevel GetLevel() const
```

Remarks

The EQRCodeLevel enum contains the four possible values, L, M, Q, H.

EQRCode::GetModel

Model of the QR code.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EQRCodeModel GetModel() const
```

Remarks

Possible values are part of the EQRCodeModel enum.

EQRCode::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EQRCode& operator=(
 const EQRCode& other
 )
```

Parameters

other

The EQRCode object that should be copied

EQRCode::GetUnusedErrorCorrection

Unused error correction.

Namespace: Euresys::Open_eVision

```
[C++]
float GetUnusedErrorCorrection() const
```

Remarks

Returns the amount of unused error correction as a percentage.
This parameter ranges from 0 to 1. Returns -1 if error correction failed (too many errors).

EQRCode::GetVersion

Version of the QR code.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetVersion() const
```

Remarks

The version of a QR code indicates its size in terms of module number per line (number of module per line = 17+4*version).

4.202. EQRCodeDecodedStream Class

Represents the complete decoded stream extracted from a QR code, EQRCode.

Namespace: Euresys::Open_eVision

Methods

EQRCodeDecodedStrea m	Creates an EQRCodeDecodedStream object.

GetApplicationIndicator	Application indicator.
GetCodingMode	Coding mode used in the decoded QR code. Returned as one of the EQRCodeCodingMode enum value.
GetDecodedStreamParts	Decoded stream parts, EQRCodeDecodedStreamPart.
GetRawBitstream	Raw bit stream as a vector of bytes.
operator=	Assignment operator

EQRCodeDecodedStream::GetApplicationIndicator

Application indicator.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetApplicationIndicator() const
```

Remarks

The application indicator is relevant if the coding mode of the QR code is EQRCodeCodingMode_Fnc1_Aim only.

EQRCodeDecodedStream::GetCodingMode

Coding mode used in the decoded QR code. Returned as one of the EQRCodeCodingMode enum value.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EQRCodeCodingMode GetCodingMode() const
```

EQRCodeDecodedStream::GetDecodedStreamParts

Decoded stream parts, EQRCodeDecodedStreamPart.

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::EQRCodeDecodedStreamPart> GetDecodedStreamParts()
const
```

EQRCodeDecodedStream::EQRCodeDecodedStream

Creates an EQRCodeDecodedStream object.

Namespace: Euresys::Open_eVision

```
[C++]
void EQRCodeDecodedStream(
 )
void EQRCodeDecodedStream(
 const EQRCodeDecodedStream& other
 )
```

Parameters

other

Another EQRCodeDecodedStream.

EQRCodeDecodedStream::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
EQRCodeDecodedStream& operator=(
 const EQRCodeDecodedStream& other
 )
```

Parameters

other

The EQRCodeDecodedStream object that should be copied

EQRCodeDecodedStream::GetRawBitstream

Raw bit stream as a vector of bytes.

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<OEV_UINT8> GetRawBitstream() const
```

Remarks

The raw bit stream is the bit stream of the QR code after unmasking and error correction, but before decoding.

4.203. EQRCodeDecodedStreamPart Class

Represents part of a decoded stream, EQRCodeDecodedStream, extracted from a QR code (EQRCode).

Namespace: Euresys::Open_eVision

Methods

EQRCodeDecodedStreamPart	Creates an EQRCodeDecodedStreamPart object.
GetDecodedData	Decoded data of this part of the decoded stream represented as a vector of bytes.
GetDecodedString	String containing the concatenated decoded data of this part of the decoded stream.
GetECITableIndicator	Extended Channel Interpretation (ECI) table indicator.
GetEncoding	Encoding scheme used for this part of the decoded stream. Available values are contained in the EQRCodeEncoding enum.
operator=	Assignment operator

EQRCodeDecodedStreamPart::GetDecodedData

Decoded data of this part of the decoded stream represented as a vector of bytes.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<OEV_UINT8> GetDecodedData()
```

EQRCodeDecodedStreamPart::GetECITableIndicator

Extended Channel Interpretation (ECI) table indicator.

Namespace: Euresys::Open_eVision

[C++]

```
int GetECITableIndicator()
```

Remarks

The ECI table indicator is relevant if the coding mode of the QR code is EQRCodeCodingMode_ECI only. Value is otherwise set to -1.

EQRCodeDecodedStreamPart::GetEncoding

Encoding scheme used for this part of the decoded stream. Available values are contained in the EQRCodeEncoding enum.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EQRCodeEncoding GetEncoding()
```

EQRCodeDecodedStreamPart::EQRCodeDecodedStreamPart

Creates an EQRCodeDecodedStreamPart object.

Namespace: Euresys::Open_eVision

```
[C++]
void EQRCodeDecodedStreamPart(
 )

void EQRCodeDecodedStreamPart(
 const EQRCodeDecodedStreamPart& other
 )
```

Parameters

other

Another EQRCodeDecodedStreamPart.

EQRCodeDecodedStreamPart::GetDecodedString

String containing the concatenated decoded data of this part of the decoded stream.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetDecodedString(
 )

std::string GetDecodedString(
 Euresys::Open_eVision::EByteInterpretationMode byteInterpretationMode
 )
```

Parameters

byteInterpretationMode

The EByteInterpretationMode that should be used to interpret bytes.

Remarks

No parameter is required if no bytes are encoded or if the ECI byte encoding mode is supported.
Exception will be thrown if a parameter is required or if a wrong one is used.

The EByteInterpretationMode_Hexadecimal parameter throws no exception and will return the bytes hexadecimal values wrapped between '0xEFBFBD'.
This mode overrides the ECI mode.
Sample : RC -> EFBFBD + RC + EFBFBD -> EFBFBD5243EFBFBD

Note: This method has currently limitations in .NET for byte encoded parts. A workaround is the conversion of the string to bytes then to UTF8.
See EByteInterpretationMode for more options.

EQRCodeDecodedStreamPart::operator=

Assignment operator

Namespace: Euresys::Open_eVision

[C++]

```
EQRCodeDecodedStreamPart& operator=(
 const EQRCodeDecodedStreamPart& other
 )
```

Parameters

other

The EQRCodeDecodedStreamPart object that should be copied.

4.204. EQRCodeGeometry Class

Represents the geometry of a QR code.
This geometry is composed of the position of the QR code and the finder pattern centers.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the QR code geometry.
DrawWithCurrentPen	Draws the QR code geometry using the pen currently set in the graphical context.
EQRCodeGeometry	Constructs an EQRCodeGeometry object.

GetFinderPatternCenters	Finder patterns centers.
GetPosition	Position of the QR code returned as an EQuadrangle object.
operator=	Assignment operator.

EQRCodeGeometry::Draw

Draws the QR code geometry.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* drawAdapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter
Draw adapter.

zoomX
Horizontal zooming factor.

zoomY
Vertical zooming factor.

panX
Horizontal panning value expressed in pixels.

panY
Vertical panning value expressed in pixels.

hDC
-

Remarks

The zoomX, zoomY, panX and panY parameters can be used to scale and/or translate the drawing operations. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQRCodeGeometry::DrawWithCurrentPen

This method is deprecated.

Draws the QR code geometry using the pen currently set in the graphical context.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC hDC,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hDC

Handle to the device context of the destination window.

zoomX

Horizontal zooming factor.

zoomY

Vertical zooming factor.

panX

Horizontal panning value expressed in pixels.

panY

Vertical panning value expressed in pixels.

Remarks

The zoomX, zoomY, panX and panY parameters can be used to scale and/or translate the drawing operations. Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQRCodeGeometry::EQRCodeGeometry

Constructs an EQRCodeGeometry object.

Namespace: Euresys::Open_eVision

[C++]

```
void EQRCodeGeometry(
 )

void EQRCodeGeometry(
 const EQRCodeGeometry& other
 )
```

```
void EQRCodeGeometry(
 const EQuadrangle& position,
 const std::vector<Euresys::Open_eVision::EPoint>& finderPatternCenters
 )
```

Parameters

other

Another EQRCodeGeometry.

position

The position of the QR code represented as an EQuadrangle object.

finderPatternCenters

The vector of Finder Pattern centers.

Remarks

In case of a Micro QR code (not yet supported), there must be only one finder pattern center. In case of another QR code, there must be three finder pattern centers, entered in the following order: top right, top left, bottom left.
The corners of the EQuadrangle must be entered in the following order : top right, top left, bottom left, bottom right.

EQRCodeGeometry::GetFinderPatternCenters

Finder patterns centers.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EPoint> GetFinderPatternCenters()
```

Remarks

In case of a Micro QR code, there is only one finder pattern center. In case of another QR code, there are three finder pattern centers, returned in the following order: top right, top left, bottom left.

EQRCodeGeometry::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EQRCodeGeometry& operator=(
 const EQRCodeGeometry& other
 )
```

Parameters

other

Another EQRCodeGeometry.

EQRCodeGeometry::GetPosition

Position of the QR code returned as an EQuadrangle object.

Namespace: Euresys::Open_eVision

[C++]

```
EQuadrangle GetPosition() const
```

4.205. EQRCodeGrid Class

Represents a grid of QR Codes

Namespace: Euresys::Open_eVision

Methods

EQRCodeGrid	Creates an EQRCodeGrid object.
GetCellEnabled	Returns true if Cell is enabled and false otherwise
GetNumCols	Returns the number of columns in the grid
GetNumRows	Returns the number of rows in the grid
GetResults	Returns the QR Codes detected
operator=	Assignment operator
SetEnableAll	Enable/Disable all cells
SetEnableCell	Enable/Disable Cell
SetEnableColumn	Enable/Disable Column
SetEnableRow	Enable/Disable Row

EQRCodeGrid::SetEnableAll

Enable/Disable all cells

Namespace: Euresys::Open_eVision

[C++]

```
void SetEnableAll(bool enable)
```

Remarks

By default, all grid cells are enabled.

EQRCodeGrid::EQRCodeGrid

Creates an EQRCodeGrid object.

Namespace: Euresys::Open_eVision

[C++]

```
void EQRCodeGrid(
 )

void EQRCodeGrid(
 const EQRCodeGrid& other
 )

void EQRCodeGrid(
 OEV_UINT32 numCols,
 OEV_UINT32 numRows
 )
```

Parameters

other

The reference EQRCodeGrid instance to copy this one from.

numCols

The number of columns in the grid.

numRows

The number of rows in the grid.

EQRCodeGrid::GetCellEnabled

Returns true if Cell is enabled and false otherwise

Namespace: Euresys::Open_eVision

[C++]

```
bool GetCellEnabled(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
```

Parameters

column

-

row

-

Remarks

By default, all grid cells are enabled.

EQRCodeGrid::GetResults

Returns the QR Codes detected

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EQRCode> GetResults(
 OEV_UINT32 column,
 OEV_UINT32 row
 )
std::vector<Euresys::Open_eVision::EQRCode> GetResults(
 )
```

Parameters

column

The column of a cell.

row

The row of a cell.

EQRCodeGrid::GetNumCols

Returns the number of columns in the grid

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumCols() const
```

EQRCodeGrid::GetNumRows

Returns the number of rows in the grid

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumRows() const
```

EQRCodeGrid::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
EQRCodeGrid& operator=(
 const EQRCodeGrid& other
 )
```

Parameters

other

The EQRCodeGrid instance to assign.

EQRCodeGrid::SetEnableCell

Enable/Disable Cell

Namespace: Euresys::Open_eVision

```
[C++]
void SetEnableCell(
 OEV_UINT32 column,
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

column

-

row

-

enable

-

Remarks

By default, all grid cells are enabled.

EQRCodeGrid::SetEnableColumn

Enable/Disable Column

Namespace: Euresys::Open_eVision

```
[C++]
void SetEnableColumn(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

EQRCodeGrid::SetEnableRow

Enable/Disable Row

Namespace: Euresys::Open_eVision

```
[C++]
void SetEnableRow(
 OEV_UINT32 row,
 bool enable
 )
```

Parameters

row

-

enable

-

Remarks

By default, all grid cells are enabled.

4.206. EQRCodeReader Class

Represents the QR code reader, that is a context for the detection and decoding of QR codes, represented by EQRCode objects.

Namespace: Euresys::Open_eVision

Methods

EQRCodeReader	Creates an EQRCodeReader object.
GetCellPolarityConfidenceThreshold	Sets the minimun cell polarity confidence threshold. When the cell confidence is under the threshold, additional processing is attempted to improve its polarity detection. The cell polarity confidence reflects the confidence in the cell digitization result, on a scale from 0 to 1. Increasing the threshold can improve reading of overprinted or underprinted QR codes. Default: 0.2f.
GetComputeGrading	Allows to choose whether the grading properties of the EQRCode object will be computed by the EQRCodeReader::Read method. The default setting for this property is `false`.
GetDetectionMethod	Sets the detection method for finding QR codes. The method can be any combination of the EQRDetectionMethod enums.
GetDetectionTradeOff	This setting controls the trade-off between computation speed versus reliability of the algorithm and particularly of the detection methods. Setting the trade-off will overwrite the current settings for EQRCodeScanPrecision and EQRDetectionMethod.
GetFilterOutUnreliablyDecodedQRCodes	Activate or deactivate the filtering of unreliably decoded QR codes.
GetForegroundDetectionThreshold	Foreground detection threshold. This parameter determines by how many grayscale-values a pixel should deviate from it's local background to be considered part of the foreground.
GetMaximumVersion	Maximum version of QR codes to be searched for.
GetMaxNumCodes	Maximum number of QR codes to find in a single Image/ROI.
GetMinimumIsotropy	QR code minimum isotropy.
GetMinimumScore	Minimum pattern finder score that must be reached to consider that a finder pattern has been found.
GetMinimumVersion	Minimum version of QR codes to be searched for.
GetScanPrecision	Precision of the EQRCodeReader when scanning the search field.
GetSearchedModels	QR code models to be searched for.
GetTimeOut	Time-out for the EQRCodeReader::Read method.
Load	Load the configuration for this EQRCodeReader instance.
operator=	Assignment operator

Read	Detects and decodes all the QR codes in the search field. Returns them as EQRCode objects.
Save	Save the configuration for this EQRCodeReader instance.
SetCellPolarityConfidenceThreshold	Sets the minimun cell polarity confidence threshold. When the cell confidence is under the threshold, additional processing is attempted to improve its polarity detection. The cell polarity confidence reflects the confidence in the cell digitization result, on a scale from 0 to 1. Increasing the threshold can improve reading of overprinted or underprinted QR codes. Default: 0.2f.
SetComputeGrading	Allows to choose whether the grading properties of the EQRCode object will be computed by the EQRCodeReader::Read method. The default setting for this property is `false`.
SetDetectionMethod	Sets the detection method for finding QR codes. The method can be any combination of the EQRDetectionMethod enums.
SetDetectionTradeOff	This setting controls the trade-off between computation speed versus reliability of the algorithm and particularly of the detection methods. Setting the trade-off will overwrite the current settings for EQRCodeScanPrecision and EQRDetectionMethod.
SetFilterOutUnreliablyDecodedQRCodes	Activate or deactivate the filtering of unreliably decoded QR codes.
SetForegroundDetectionThreshold	Foreground detection threshold. This parameter determines by how many grayscale-values a pixel should deviate from it's local background to be considered part of the foreground.
SetMaximumVersion	Maximum version of QR codes to be searched for.
SetMaxNumCodes	Maximum number of QR codes to find in a single Image/ROI.
SetMinimumIsotropy	QR code minimum isotropy.
SetMinimumScore	Minimum pattern finder score that must be reached to consider that a finder pattern has been found.
SetMinimumVersion	Minimum version of QR codes to be searched for.
SetScanPrecision	Precision of the EQRCodeReader when scanning the search field.
SetSearchedModels	QR code models to be searched for.
SetSearchField	Search field for the QR code reader.
SetTimeOut	Time-out for the EQRCodeReader::Read method.

EQRCodeReader::GetCellPolarityConfidenceThreshold

EQRCodeReader::SetCellPolarityConfidenceThreshold

Sets the minimun cell polarity confidence threshold.
When the cell confidence is under the threshold, additional processing is attempted to improve its polarity detection. The cell polarity confidence reflects the confidence in the cell digitization result, on a scale from 0 to 1. Increasing the threshold can improve reading of overprinted or underprinted QR codes. Default: 0.2f.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCellPolarityConfidenceThreshold() const
void SetCellPolarityConfidenceThreshold(float threshold)
```

Remarks

Large values can affect the speed and will increase the probability of false positive changes.

EQRCodeReader::GetComputeGrading

EQRCodeReader::SetComputeGrading

Allows to choose whether the grading properties of the EQRCode object will be computed by the EQRCodeReader::Read method. The default setting for this property is false.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetComputeGrading() const
void SetComputeGrading(bool computeGrading)
```

Remarks

This setting is not available for Micro QR code symbols.

EQRCodeReader::GetDetectionMethod

EQRCodeReader::SetDetectionMethod

Sets the detection method for finding QR codes. The method can be any combination of the EQRDetectionMethod enums.

Namespace: Euresys::Open_eVision

[C++]

```
int GetDetectionMethod() const

void SetDetectionMethod(int method)
```

Remarks

The default value is: 'EQRDetectionMethod_Gradient|EQRDetectionMethod_AdaptiveThreshold'.

EQRCodeReader::GetDetectionTradeOff

EQRCodeReader::SetDetectionTradeOff

This setting controls the trade-off between computation speed versus reliability of the algorithm and particularly of the detection methods.
Setting the trade-off will overwrite the current settings for EQRCodeScanPrecision and EQRDetectionMethod.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EQRDetectionTradeOff GetDetectionTradeOff() const

void SetDetectionTradeOff(Euresys::Open_eVision::EQRDetectionTradeOff tradeOff)
```

Remarks

Available values are defined by EQRDetectionTradeOff. The default value is **EQRDetectionTradeOff_Balanced**.

EQRCodeReader::EQRCodeReader

Creates an EQRCodeReader object.

Namespace: Euresys::Open_eVision

[C++]

```
void EQRCodeReader(
 )

void EQRCodeReader(
 const EQRCodeReader& other
 )
```

Parameters

other

Another EQRCodeReader object to be copied in the new EQRCodeReader object.

EQRCodeReader::GetFilterOutUnreliablyDecodedQRCodes

EQRCodeReader::SetFilterOutUnreliablyDecodedQRCodes

Activate or deactivate the filtering of unreliably decoded QR codes.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFilterOutUnreliablyDecodedQRCodes() const
void SetFilterOutUnreliablyDecodedQRCodes(bool filter)
```

Remarks

By default, the QR code reader does not return unreliably decoded QR codes.

EQRCodeReader::GetForegroundDetectionThreshold

EQRCodeReader::SetForegroundDetectionThreshold

Foreground detection threshold. This parameter determines by how many grayscale-values a pixel should deviate from it's local background to be considered part of the foreground.

Namespace: Euresys::Open_eVision

[C++]

```
int GetForegroundDetectionThreshold() const
void SetForegroundDetectionThreshold(int threshold)
```

Remarks

In most cases, changing the value of this parameter is not required. A small threshold value may help to locate codes in low contrast images. If you think you might need to use this parameter, contact technical support for advice. The default value for this parameter is 10.

EQRCodeReader::Load

Load the configuration for this EQRCodeReader instance.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )
```

```
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The path from which to load the configuration.

serializer

The given ESerializer, it must have been created for reading.

EQRCodeReader::GetMaximumVersion

EQRCodeReader::SetMaximumVersion

Maximum version of QR codes to be searched for.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMaximumVersion() const
void SetMaximumVersion(OEV_UINT32 version)
```

Remarks

This parameter value ranges from 1 to 40. Default value: 40.

EQRCodeReader::GetMaxNumCodes

EQRCodeReader::SetMaxNumCodes

Maximum number of QR codes to find in a single Image/ROI.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMaxNumCodes() const
void SetMaxNumCodes(OEV_UINT32 max)
```

Remarks

By default, this parameter is set to 1.

EQRCodeReader::GetMinimumIsotropy

EQRCodeReader::SetMinimumIsotropy

QR code minimum isotropy.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumIsotropy() const

void SetMinimumIsotropy(float isotropy)
```

Remarks

The isotropy of a QR code is defined as its short side divided by its long side.
This parameter value ranges from 0 to 1. Default value: 0.75.

EQRCodeReader::GetMinimumScore

EQRCodeReader::SetMinimumScore

Minimum pattern finder score that must be reached to consider that a finder pattern has been found.

Namespace: Euresys::Open_eVision

[C++]

```
float GetMinimumScore() const

void SetMinimumScore(float minScore)
```

Remarks

The pattern finder score is based on a normalized correlation with a perfect finder pattern model.
A perfect match with the model would return a score of 1.
This parameter value ranges from 0 to 1. Default value: 0.7.

EQRCodeReader::GetMinimumVersion

EQRCodeReader::SetMinimumVersion

Minimum version of QR codes to be searched for.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinimumVersion() const
void SetMinimumVersion(OEV_UINT32 version)
```

Remarks

This parameter value ranges from 1 to 40. Default value: 1.

EQRCodeReader::operator=

Assignment operator

Namespace: Euresys::Open_eVision

[C++]

```
EQRCodeReader& operator=(
 const EQRCodeReader& other
 )
```

Parameters

other

The object that should be copied

EQRCodeReader::Read

Detects and decodes all the QR codes in the search field. Returns them as EQRCode objects.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EQRCode> Read(
 )
std::vector<Euresys::Open_eVision::EQRCode> Read(
 EROIBW8& field
 )
std::vector<Euresys::Open_eVision::EQRCode> Read(
 EROIBW8& field,
 const ERegion& region
 )
EQRCodeGrid Read(
 EROIBW8& field,
 int numCellsX,
 int numCellsY,
 float extension
 )
```

```
EQRCodeGrid Read(
 EROIBW8& field,
 const ERectangleRegion& area,
 int numCellsX,
 int numCellsY,
 float extension
 )

EQRCodeGrid Read(
 EROIBW8& field,
 const ERectangleRegion& area,
 const EQRCodeGrid& grid,
 float extension
 )
```

Parameters

field

The search field.

region

Region into the search field where the qr codes have to be found.

numCellsX

Number of grid cells in the X direction

numCellsY

Number of grid cells in the Y direction

extension

Extension of the grid cells to allow cell overlap. For instance, 0.0f means no extension and 0.1f means a 10% cell size extension.

area

Rectangular Region used as the full grid area

grid

Grid with cell disabling capabilities

Remarks

The grid overload allows you to disable some cells of the grid if those cells are not supposed to contain QR Codes. See the EQRCodeGrid class documentation for more information.

EQRCodeReader::Save

Save the configuration for this EQRCodeReader instance.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
```

```
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The path to which to save the configuration.

serializer

The given ESerializer, it must have been created for writing.

EQRCodeReader::GetScanPrecision

EQRCodeReader::SetScanPrecision

Precision of the EQRCodeReader when scanning the search field.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EQRCodeScanPrecision GetScanPrecision() const
void SetScanPrecision(Euresys::Open_eVision::EQRCodeScanPrecision precision)
```

Remarks

Available values are defined by EQRCodeScanPrecision. The default value is **EQRCodeScanPrecision_Automatic**.

EQRCodeReader::GetSearchedModels

EQRCodeReader::SetSearchedModels

QR code models to be searched for.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EQRCodeModel> GetSearchedModels() const
void SetSearchedModels(const std::vector<Euresys::Open_eVision::EQRCodeModel>& models)
```

Remarks

By default, the QR code reader searches for EQRCodeModel_Model1 and EQRCodeModel_Model2.

EQRCodeReader::SetSearchField

Search field for the QR code reader.

Namespace: Euresys::Open_eVision

```
[C++]
void SetSearchField(EROIBW8& field)
```

EQRCodeReader::GetTimeOut

EQRCodeReader::SetTimeOut

Time-out for the EQRCodeReader::Read method.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT64 GetTimeOut() const
void SetTimeOut(OEV_UINT64 value)
```

Remarks

If the processing time of one of these functions becomes longer than the set time-out, the processing is stopped and an exception is thrown.
In that case, the error code of the exception is EError_TimeoutReached.
The time-out is set in microseconds.
This time-out is not a real time-out.
The processing is stopped as soon as possible after the time-out has been reached.
This means that the time elapsed effectively in the method can be greater than the time-out itself.

4.207. EQuadrangle Class

This class represents a polygon with four corners (with sub-pixel accuracy).

Remarks

A quadrangle especially arises when representing the corners of a rotated bounding box.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the quadrangle, by drawing lines between its corners.
DrawWithCurrentPen	Draws the quadrangle, by drawing lines between its corners.

EQuadrangle	Constructs a EQuadrangle. The default constructor initializes all points to 0.
GetArea	Returns the area of the quadrangle.
GetCornerAngle	Returns the angle of a given corner of the quadrangle.
GetCorners	The corners of the EQuadrangle.
GetGravityCenter	Returns the gravity center of the quadrangle.
GetPerimeter	Returns the perimeter of the quadrangle.
GetPoint	Returns the coordinate of a given corner of the quadrangle.
GetSideAngle	Returns the angle of a given side of the quadrangle.
GetSideLength	Returns the length of a given side of the quadrangle.
GetUpperLeftCorner	Returns the coordinates of the upper left corner of the quadrangle (the position with the minimum Y and then minimum X).
IsPointInside	Tests if a point is inside the quadrangle.
IsQuadrangleInside	Tests if a quadrangle is inside the quadrangle.
operator=	Assignment operator.
OverLaps	Tests if the quadrangles overlap.
SetPoint	Sets the coordinates of a given corner of the quadrangle.

EQuadrangle::GetArea

Returns the area of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetArea() const
```

EQuadrangle::GetCorners

The corners of the EQuadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EPoint> GetCorners() const
```

Remarks

If the EQuadrangle belongs to an EQRCodeGeometry object, the corners are returned in the following order: top right, top left, bottom left, bottom right.

EQuadrangle::Draw

Draws the quadrangle, by drawing lines between its corners.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals

Specifies whether or not lines are to be drawn between the 1st and 3rd corners, as well as between the 2nd and 4th corners.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQuadrangle::DrawWithCurrentPen

This method is deprecated.

Draws the quadrangle, by drawing lines between its corners.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

drawDiagonals

Specifies whether or not lines are to be drawn between the 1st and 3rd corners, as well as between the 2nd and 4th corners.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EQuadrangle::EQuadrangle

Constructs a EQuadrangle. The default constructor initializes all points to 0.

Namespace: Euresys::Open_eVision

```
[C++]
void EQuadrangle(
 )
void EQuadrangle(
 std::vector<Euresys::Open_eVision::EPoint> corners
 )
void EQuadrangle(
 const EQuadrangle& other
 )
```

Parameters

corners

The corners of the EQuadrangle.

other

The source EQuadrangle.

EQuadrangle::GetCornerAngle

Returns the angle of a given corner of the quadrangle.

Namespace: Euresys::Open_eVision

```
[C++]
float GetCornerAngle(
 OEV_UINT32 cornerIndex
 )
```

Parameters

cornerIndex

-

EQuadrangle::GetPoint

Returns the coordinate of a given corner of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 const OEV_UINT32 index
 )
```

Parameters

index

The index of the corner of interest (must lie in the range between 0 and 3, inclusive).

EQuadrangle::GetSideAngle

Returns the angle of a given side of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSideAngle(
 OEV_UINT32 sideIndex
 )
```

Parameters

sideIndex

The index of the side of interest (must lie in the range between 0 and 3, inclusive).

EQuadrangle::GetSideLength

Returns the length of a given side of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSideLength(
 OEV_UINT32 sideIndex
 )
```

Parameters

sideIndex

The index of the side of interest (must lie in the range between 0 and 3, inclusive).

EQuadrangle::GetGravityCenter

Returns the gravity center of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetGravityCenter() const
```

EQuadrangle::IsPointInside

Tests if a point is inside the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsPointInside(
 const EPoint& point,
 bool includeEdges
 )
```

Parameters

point

-

includeEdges

Takes the edges as part of the interior of the EQuadrangle. Default: false.

EQuadrangle::IsQuadrangleInside

Tests if a quadrangle is inside the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsQuadrangleInside(
 const EQuadrangle& quadrangle,
 bool includeEdges
 )
```

Parameters

quadrangle

-

includeEdges

Takes the edges as part of the interior of the EQuadrangle. Default: false.

EQuadrangle::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EQuadrangle& operator=(
 const EQuadrangle& other
 )
```

Parameters

other

The source EQuadrangle.

EQuadrangle::OverLaps

Tests if the quadrangles overlap.

Namespace: Euresys::Open_eVision

[C++]

```
bool OverLaps(
 const EQuadrangle& other
 )
```

Parameters

other

The EQuadrangle to test.

EQuadrangle::GetPerimeter

Returns the perimeter of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPerimeter() const
```

EQuadrangle::SetPoint

Sets the coordinates of a given corner of the quadrangle.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPoint(
 const OEV_UINT32& index,
 const EPoint& location
 )
```

Parameters

index

The index of the corner of interest (must lie in the range between 0 and 3, inclusive).

location

The coordinate.

EQuadrangle::GetUpperLeftCorner

Returns the coordinates of the upper left corner of the quadrangle (the position with the minimum Y and then minimum X).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetUpperLeftCorner() const
```

4.208. ERandomDecimator Class

Decimation of an EPointCloud.
The random decimator decimates a point cloud by copying a specified number of points, randomly selected, to a new point cloud.

Base Class:EDecimator

Namespace: Euresys::Open_eVision::Easy3D

Methods

Decimate	Decimates a given EPointCloud and writes the result into a new point cloud.
ERandomDecimator	Creates an ERandomDecimator object. If the required number of points (after decimation) is not specified, the default value is 10000.
GetNumberOfPoints	Sets/gets the parameter "number of points" (number of points after decimation).
operator!=	test inequality between two ERandomDecimator.
operator=	Assignment operator.
operator==	test equality between two ERandomDecimator.
SetNumberOfPoints	Sets/gets the parameter "number of points" (number of points after decimation).

ERandomDecimator::Decimate

Decimates a given EPointCloud and writes the result into a new point cloud.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Decimate(
 const EPointCloud& cloudIn,
 EPointCloud& cloudOut
 )
```

Parameters

cloudIn

The input point cloud.

cloudOut

The output point cloud.

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

ERandomDecimator::ERandomDecimator

Creates an ERandomDecimator object. If the required number of points (after decimation) is not specified, the default value is 10000.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ERandomDecimator(
 )

void ERandomDecimator(
 int numberOfPoints
 )

void ERandomDecimator(
 const ERandomDecimator& other
 )
```

Parameters

numberOfPoints

Number of points after decimation.

other

Reference to the ERandomDecimator object used for the initialization.

ERandomDecimator::GetNumberOfPoints

ERandomDecimator::SetNumberOfPoints

Sets/gets the parameter "number of points" (number of points after decimation).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetNumberOfPoints() const
void SetNumberOfPoints(int numberOfPoints)
```

ERandomDecimator::operator!=

test inequality between two ERandomDecimator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator!=(
 const EDecimator& other
 )
```

Parameters

other

the ERandomDecimator to be compared with

ERandomDecimator::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
ERandomDecimator& operator=(
 const ERandomDecimator& other
 )
```

Parameters

other

The ERandomDecimator object that should be copied.

ERandomDecimator::operator==

test equality between two ERandomDecimator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator==(
 const EDecimator& other
 )
```

Parameters

other

the ERandomDecimator to be compared with

4.209. ERectangle Class

Represents a model of a rectangle.

Base Class:EFrame

Namespace: Euresys::Open_eVision

Methods

CopyTo	Copies all the data of the current ERectangle object into another ERectangle object, and returns it.
ERectangle	Constructs a ERectangle
GetCorners	Retrieves the coordinates of each corner of a ERectangle object.
GetEdges	Retrieves each edge of a ERectangle object.
GetMidEdges	Retrieves the center coordinates of each edge of a ERectangle object.
GetPoint	Returns the coordinates of a particular point, specified by its location in the ERectangle area.
GetSizeX	X size of the ERectangle
GetSizeY	Y size of the ERectangle
operator=	Copies all the data from another ERectangle object into the current ERectangle object
SetFromOppositeCorners	Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.
SetFromOriginMiddleEnd	DEPRECATED (you should use ERectangle::SetFromThreeCorners): Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.
SetFromThreeCorners	Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.
SetFromTwoPoints	DEPRECATED (you should use ERectangle::SetFromOppositeCorners): Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.
SetSize	Sets the size of a ERectangle object.

ERectangle::CopyTo

Copies all the data of the current ERectangle object into another ERectangle object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ERectangle& other
 )

ERectangle* CopyTo(
 ERectangle* other
 )
```

Parameters

other

Pointer to the ERectangle object in which the current ERectangle object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ERectangle object will be created and returned.

ERectangle::ERectangle

Constructs a ERectangle

Namespace: Euresys::Open_eVision

[C++]

```
void ERectangle(
 )

void ERectangle(
 const EPoint& center,
 float sizeX,
 float sizeY,
 float angle
 )

void ERectangle(
 const EPoint& origin,
 const EPoint& end
 )

void ERectangle(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end
 )

void ERectangle(
 const ERectangle& other
 )
```

Parameters

center
Center coordinates of the rectangle at its nominal position. The default value is (0,0).
sizeX
Nominal size X/Y of the rectangle. Both default values are 100.
sizeY
Nominal size X/Y of the rectangle. Both default values are 100.
angle
Nominal rotation angle of the rectangle. The default value is 0.
origin
Upper left point coordinates of the rectangle.
end
Lower right point coordinates of the rectangle.
middle
A third corner point coordinates.
other
Another ERectangle object to be copied in the new ERectangle object.

ERectangle::GetCorners

Retrieves the coordinates of each corner of a ERectangle object.

Namespace: Euresys::Open_eVision

[C++]

```
void GetCorners(
 EPoint& xy,
 EPoint& XXy,
 EPoint& xYY,
 EPoint& XXYY
 )
```

Parameters

xy
Coordinates of the lower leftmost corner of the ERectangle object.
XXy
Coordinates of the lower rightmost corner of the ERectangle object.
xYY
Coordinates of the upper leftmost corner of the ERectangle object.
XXYY
Coordinates of the upper rightmost corner of the ERectangle object.

ERectangle::GetEdges

Retrieves each edge of a ERectangle object.

Namespace: Euresys::Open_eVision

```
[C++]
void GetEdges(
 ELine& x,
 ELine& XX,
 ELine& y,
 ELine& YY
 )
```

Parameters

x
Leftmost edge of the ERectangle object.

XX
Rightmost edge of the ERectangle object.

y
Lower edge of the ERectangle object.

YY
Upper edge of the ERectangle object.

ERectangle::GetMidEdges

Retrieves the center coordinates of each edge of a ERectangle object.

Namespace: Euresys::Open_eVision

```
[C++]
void GetMidEdges(
 EPoint& x,
 EPoint& XX,
 EPoint& y,
 EPoint& YY
 )
```

Parameters

x
Center coordinates of the leftmost edge of the ERectangle object.

XX
Center coordinates of the rightmost edge of the ERectangle object.

y
Center coordinates of the lower edge of the ERectangle object.

YY
Center coordinates of the upper edge of the ERectangle object.

ERectangle::GetPoint

Returns the coordinates of a particular point, specified by its location in the ERectangle area.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetPoint(
 float fractionX,
 float fractionY
 )
```

Parameters

fractionX

Point location expressed as a fraction of the ERectangle vertical edges (range -1, +1).

fractionY

Point location expressed as a fraction of the ERectangle horizontal edges (range -1, +1).

ERectangle::operator=

Copies all the data from another ERectangle object into the current ERectangle object

Namespace: Euresys::Open_eVision

```
[C++]
ERectangle& operator=(
 const ERectangle& other
 )
```

Parameters

other

ERectangle object to be copied

ERectangle::SetFromOppositeCorners

Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOppositeCorners(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin

Upper left point coordinates of the rectangle.

end

Lower right point coordinates of the rectangle.

ERectangle::SetFromOriginMiddleEnd

This method is deprecated.

DEPRECATED (you should use ERectangle::SetFromThreeCorners): Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end
 )
```

Parameters

origin

Upper left point coordinates of the rectangle.

middle

A third corner point coordinates.

end

Lower right point coordinates of the rectangle.

ERectangle::SetFromThreeCorners

Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromThreeCorners(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end
 )
```

Parameters

origin
Upper left point coordinates of the rectangle.

middle
A third corner point coordinates.

end
Lower right point coordinates of the rectangle.

ERectangle::SetFromTwoPoints

This method is deprecated.

DEPRECATED (you should use ERectangle::SetFromOppositeCorners): Sets the geometric parameters (center coordinates, size, and rotation angle) of an ERectangle object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromTwoPoints(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin
Upper left point coordinates of the rectangle.

end
Lower right point coordinates of the rectangle.

ERectangle::SetSize

Sets the size of a ERectangle object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSize(
 float sizeX,
 float sizeY
 )
```

Parameters

sizeX

Nominal size X of the ERectangle object. Default values is 100.

sizeY

Nominal size Y of the ERectangle object. Default values is 100.

Remarks

A ERectangle object is fully defined knowing its nominal position (given by the coordinates of its center), its nominal size, its rotation angle and its outline tolerance. By default, the width and height values are 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

ERectangle::GetSizeX

X size of the ERectangle

Namespace: Euresys::Open_eVision

```
[C++]
float GetSizeX() const
```

ERectangle::GetSizeY

Y size of the ERectangle

Namespace: Euresys::Open_eVision

```
[C++]
float GetSizeY() const
```

4.210. ERectangleGauge Class

Manages a rectangle fitting gauge.

Base Class:ERectangleShape

Namespace: Euresys::Open_eVision

Methods

AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.
CopyTo	Copies all the data of the current ERectangleGauge object into another ERectangleGauge object, and returns it.
DisableInnerFiltering	Disables inner sampled point filtering.

Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
ERectangleGauge	Constructs a rectangle measurement context.
GetActiveEdges	Active edges as defined in EDragHandle.
GetAverageDistance	Average distance between the sampled points and the fitted model.
GetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
GetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
GetInnerFilteringEnabled	Getter method for the GetInnerFilteringEnabled property. This property is the flag indicating if the inner sampled point filtering is enabled (true).
GetInnerFilteringThreshold	Sampled point inner filtering threshold.
GetKnownAngle	Flag indicating whether the rotation angle of the rectangle to be fitted is known or not.
GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.
GetMeasuredRectangle	Information pertaining to the fitted rectangle.
GetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
GetMinArea	Minimum area value.
GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetNumFilteringPasses	Number of filtering passes for a model fitting operation.
GetNumSamples	Number of sampled points during the model fitting operation.
GetNumSamplesLeftEdge	Number of sampled points found on leftmost edge during the measure operation.
GetNumSamplesLowerEdge	Number of sampled points found on lower edge during the measure operation.
GetNumSamplesRightEdge	Number of sampled points found on rightmost edge during the measure operation.
GetNumSamplesUpperEdge	Number of sampled points found on upper edge during the measure operation.
GetNumSamplesx	Number of sampled points found on edge x during the measure operation.
GetNumSamplesX	Number of sampled points found on edge X during the measure operation.

GetNumSamplesy	Number of sampled points found on edge y during the measure operation.
GetNumSamplesY	Number of sampled points found on edge Y during the measure operation.
GetNumSkipRanges	Number of skip ranges in the gauge after a call to ERectangleGauge::AddSkipRange.
GetNumValidSamples	Number of valid sample points remaining after a model fitting operation.
GetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
GetSampleLeftEdge	Allows to retrieve information on the samples found along the leftmost edge.
GetSampleLowerEdge	Allows to retrieve information on the samples found along the lower edge.
GetSampleRightEdge	Allows to retrieve information on the samples found along the rightmost edge.
GetSampleUpperEdge	Allows to retrieve information on the samples found along the upper edge.
GetSamplex	Allows to retrieve information on the samples found along the x edge.
GetSampleX	Allows to retrieve information on the samples found along the X edge.
GetSampley	Allows to retrieve information on the samples found along the y edge.
GetSampleY	Allows to retrieve information on the samples found along the Y edge.
GetSamplingStep	Approximate distance between sampled points during a model fitting operation.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ERectangleGauge::AddSkipRange method.
GetSmoothing	Number of pixels used for the low-pass filtering operation.
GetThickness	Number of parallel segments used to extract the data profile.
GetThreshold	Threshold level used to delimit significant peaks in the data profile.
GetTolerance	Searching area half thickness of the rectangle fitting gauge.
GetTransitionChoice	Transition choice.
GetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
GetTransitionType	Transition type.
GetType	Shape type.
GetValid	Flag indicating if at least one valid transition has been found.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Measure	Triggers the point location or the model fitting operation.

MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without rectangle fitting operation.
operator=	Copies all the data from another ERectangleGauge object into the current ERectangleGauge object
Plot	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
PlotWithCurrentPen	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.
RemoveAllSkipRanges	Removes all the skip ranges previously created by a call to ERectangleGauge::AddSkipRange.
RemoveSkipRange	After a call to ERectangleGauge::AddSkipRange, removes the skip range with the given index.
SetActive	Sets the flag indicating whether the gauge is active or not.
SetActiveEdges	Active edges as defined in EDragHandle.
SetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
SetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
SetInnerFilteringThreshold	Sampled point inner filtering threshold.
SetKnownAngle	Flag indicating whether the rotation angle of the rectangle to be fitted is known or not.
SetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
SetMinArea	Minimum area value.
SetMinNumFitSamples	Sets the minimum number of samples required for fitting on each side of the shape.
SetNumFilteringPasses	Number of filtering passes for a model fitting operation.
SetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
SetSamplingStep	Approximate distance between sampled points during a model fitting operation.
SetSmoothing	Number of pixels used for the low-pass filtering operation.
SetThickness	Number of parallel segments used to extract the data profile.
SetThreshold	Threshold level used to delimit significant peaks in the data profile.
SetTolerance	Searching area half thickness of the rectangle fitting gauge.
SetTransitionChoice	Transition choice.

SetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
SetTransitionType	Transition type.

ERectangleGauge::SetActive

Sets the flag indicating whether the gauge is active or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetActive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying ERectangleGauge::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (true).

ERectangleGauge::GetActiveEdges

ERectangleGauge::SetActiveEdges

Active edges as defined in EDragHandle.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetActiveEdges()

void SetActiveEdges(OEV_UINT32 un32ActiveEdges)
```

Remarks

In the case of a rectangle fitting gauge, each edge can have its own transition detection parameters. Updating the transition parameters only affect the current active edges. By default, all edges are active.

ERectangleGauge::AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 AddSkipRange(
 const OEV_UINT32 start,
 const OEV_UINT32 end
 )
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process.
The ERectangleGauge::AddSkipRange method allows to define skip ranges in an ERectangleGauge. This means that, at measure time, samples belonging to these ranges will not be taken into account.
A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another.
The range is allowed to be reversed (i.e. end is not required to be greater than start).
Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for ERectangleGauge::NumSamples).

ERectangleGauge::GetAverageDistance

Average distance between the sampled points and the fitted model.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAverageDistance()
```

Remarks

Irrelevant in case of a point location operation.

ERectangleGauge::CopyTo

Copies all the data of the current ERectangleGauge object into another ERectangleGauge object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 ERectangleGauge& other,
 bool recursive
 )

ERectangleGauge* CopyTo(
 ERectangleGauge* other,
 bool recursive
 )
```

Parameters

other

Pointer to the ERectangleGauge object in which the current ERectangleGauge object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ERectangleGauge object will be created and returned.

ERectangleGauge::DisableInnerFiltering

Disables inner sampled point filtering.

Namespace: Euresys::Open_eVision

[C++]

```
void DisableInnerFiltering(
 )
```

Remarks

The inner sampled point filtering is activated as soon as the corresponding ERectangleGauge::InnerFilteringThreshold is set.

ERectangleGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y
 )
```

Parameters

x

Cursor current X coordinate.

y

Cursor current Y coordinate.

ERectangleGauge::Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERectangleGauge::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERectangleGauge::ERectangleGauge

Constructs a rectangle measurement context.

Namespace: Euresys::Open_eVision

```
[C++]
void ERectangleGauge(
 )

void ERectangleGauge(
 const ERectangleGauge& other
 )
```

Parameters

other

Another ERectangleGauge object to be copied in the new ERectangleGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values.
With the copy constructor, the constructed rectangle measurement context is based on a pre-existing ERectangleGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the ERectangleGauge::CopyTo method.

ERectangleGauge::GetFilteringThreshold

ERectangleGauge::SetFilteringThreshold

Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

Namespace: Euresys::Open_eVision

[C++]

```
float GetFilteringThreshold()

void SetFilteringThreshold(float f32FilteringThreshold)
```

Remarks

Irrelevant in case of a point location operation.
During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).

ERectangleGauge::GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetMeasuredPoint(
 OEV_UINT32 index
 )
```

Parameters

index

This argument must be left unchanged from its default value, i.e. ~0 (= 0xFFFFFFFF).

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current ERectangleGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry. ERectangleGauge::GetMeasuredPoint returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to ERectangleGauge::MeasureSample, and 1. Among all the sample points along the latter sample path, it is the one selected by the ERectangleGauge::TransitionChoice property.

Note. For this method to succeed, it is necessary to previously call ERectangleGauge::MeasureSample.

ERectangleGauge::GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void GetMinNumFitSamples(
 int& side0,
 int& side1,
 int& side2,
 int& side3
 )
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

ERectangleGauge::GetSampleLeftEdge

Allows to retrieve information on the samples found along the leftmost edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleLeftEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleLeftEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleLeftEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge::GetSampleLowerEdge

Allows to retrieve information on the samples found along the lower edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleLowerEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleLowerEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleLowerEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge::GetSampleRightEdge

Allows to retrieve information on the samples found along the rightmost edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleRightEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleRightEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleRightEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge::GetSampleUpperEdge

Allows to retrieve information on the samples found along the upper edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleUpperEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleUpperEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleUpperEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

ERectangleGauge::GetSampleX

This method is deprecated.

Allows to retrieve information on the samples found along the X edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleX(
 EPoint& pt,
 OEV_UINT32 index
 )
```

```
void GetSampleX(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleX(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use ERectangleGauge::GetSampleRightEdge instead.

ERectangleGauge::GetSampleX

This method is deprecated.

Allows to retrieve information on the samples found along the X edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleX(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleX(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleX(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index
The sample index
sp
ESamplePoint structure that will receive the sample position.
pk
EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use ERectangleGauge::GetSampleRightEdge instead.

ERectangleGauge::GetSampleY

This method is deprecated.

Allows to retrieve information on the samples found along the Y edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleY(
 EPoint& pt,
 OEV_UINT32 index
 )
void GetSampleY(
 ESamplePoint& sp,
 OEV_UINT32 index
 )
bool GetSampleY(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt
EPoint structure that will receive the sample position.
index
The sample index
sp
ESamplePoint structure that will receive the sample position.
pk
EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use ERectangleGauge::GetSampleUpperEdge instead.

ERectangleGauge::GetSampleY

This method is deprecated.

Allows to retrieve information on the samples found along the Y edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleY(
 EPoint& pt,
 OEV_UINT32 index
 )
void GetSampleY(
 ESamplePoint& sp,
 OEV_UINT32 index
 )
bool GetSampleY(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use ERectangleGauge::GetSampleUpperEdge instead.

ERectangleGauge::GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the ERectangleGauge::AddSkipRange method.

Namespace: Euresys::Open_eVision

```
[C++]
void GetSkipRange(
 const OEV_UINT32 index,
 OEV_UINT32& start,
 OEV_UINT32& end
 )
```

Parameters

index
Index of the skip range.

start
Beginning of the skip range.

end
End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

ERectangleGauge::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 bool daughters
 )
```

Parameters

daughters
true if the daughters gauges handles have to be considered as well.

ERectangleGauge::GetHVConstraint

ERectangleGauge::SetHVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

Namespace: Euresys::Open_eVision

[C++]

```
BOOL GetHVConstraint()

void SetHVConstraint(BOOL bHVConstraint)
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

ERectangleGauge::GetInnerFilteringEnabled

Getter method for the GetInnerFilteringEnabled property. This property is the flag indicating if the inner sampled point filtering is enabled (true).

Namespace: Euresys::Open_eVision

```
[C++]
bool GetInnerFilteringEnabled()
```

Remarks

The inner sampled point filtering is activated as soon as the corresponding threshold is set, getting the ERectangleGauge.InnerFilteringThreshold property. To disable inner filtering, use the DisableInnerFiltering method.

ERectangleGauge::GetInnerFilteringThreshold

ERectangleGauge::SetInnerFilteringThreshold

Sampled point inner filtering threshold.

Namespace: Euresys::Open_eVision

```
[C++]
float GetInnerFilteringThreshold()
void SetInnerFilteringThreshold(float f32InnerFilteringThreshold)
```

Remarks

If inner filtering is enabled, the sampled points that have been found inside the measured rectangle are filtered in regard of their distance to it. If this distance is greater than the threshold, the sampled point is set as invalid, and removed from the measure. This distance is in physical units.
The inner sampled point filtering is activated as soon as the corresponding threshold is set. To disable inner filtering, use the DisableInnerFiltering method.

ERectangleGauge::GetKnownAngle

ERectangleGauge::SetKnownAngle

Flag indicating whether the rotation angle of the rectangle to be fitted is known or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetKnownAngle()
```

```
void SetKnownAngle(bool bKnownAngle)
```

Remarks

A rectangle model to be fitted may have a well-known orientation. It is possible to impose the value of this rotation angle, thus removing one degree of freedom. The rectangle fitting gauge orientation is set by means of the ERectangleGauge.Angle property.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

ERectangleGauge::Measure

Triggers the point location or the model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
void Measure(
 EROIBW8* sourceImage
 )

void Measure(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Pointer to the source image.

region

Region to use with the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ERectangleGauge::GetMeasuredRectangle

Information pertaining to the fitted rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
ERectangle GetMeasuredRectangle()
```

Remarks

Use method EShape::GetFound to get the status of the measurement. ERectangleGauge::MeasuredRectangle returns a successful fitted rectangle if EShape::GetFound is true, otherwise it returns the orginal (nominal) rectangle.

ERectangleGauge::MeasureSample

Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureSample(
 EROIBW8* sourceImage,
 OEV_UINT32 pathIndex
 )
void MeasureSample(
 EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 pathIndex
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

region

Region on which to measure.

Remarks

This method stores its results into a temporary variable inside the ERectangleGauge object.

ERectangleGauge::MeasureWithoutFitting

Triggers the point location without rectangle fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureWithoutFitting(
 EROIBW8* sourceImage
 )
```

```
void MeasureWithoutFitting(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Source image.

region

Region on which to measure.

Remarks

This method performs the actual measurement for each transition, but does not perform the rectangle fitting. This means that individual samples will be available for each edges through the ERectangleGauge::GetSamplex (Edge x), ERectangleGauge::GetSampley (Edge y), ERectangleGauge::GetSampleX (Edge X), ERectangleGauge::GetSampleY (Edge Y) methods, but the gauge position will not be changed.
Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

ERectangleGauge::GetMinAmplitude

ERectangleGauge::SetMinAmplitude

Offset added to the Threshold when a peak is to be detected.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetMinAmplitude()

void SetMinAmplitude(OEV_UINT32 un32MinAmplitude)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value.
To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected.
When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

ERectangleGauge::GetMinArea

ERectangleGauge::SetMinArea

Minimum area value.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMinArea()
void SetMinArea(OEV_UINT32 un32MinArea)
```

Remarks

A transition is detected if its derivative peak reaches Threshold + MinAmplitude value, and then declared valid if the area between the peak curve and the horizontal at level Threshold reaches the MinArea value.

ERectangleGauge::GetNumFilteringPasses

ERectangleGauge::SetNumFilteringPasses

Number of filtering passes for a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumFilteringPasses()
void SetNumFilteringPasses(OEV_UINT32 un32NumFilteringPasses)
```

Remarks

Irrelevant in case of a point location operation.
During a filtering pass, the points that are too distant from the model are discarded.
During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).
By default (the number of filtering passes is 0), the outliers rejection process is disabled.

ERectangleGauge::GetNumSamples

Number of sampled points during the model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamples() const
```

Remarks

Irrelevant in case of a point location operation.
After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

ERectangleGauge::GetNumSamplesLeftEdge

Number of sampled points found on leftmost edge during the measure operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamplesLeftEdge() const
```

ERectangleGauge::GetNumSamplesLowerEdge

Number of sampled points found on lower edge during the measure operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamplesLowerEdge() const
```

ERectangleGauge::GetNumSamplesRightEdge

Number of sampled points found on rightmost edge during the measure operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamplesRightEdge() const
```

ERectangleGauge::GetNumSamplesUpperEdge

Number of sampled points found on upper edge during the measure operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamplesUpperEdge() const
```

ERectangleGauge::GetNumSamplesX

This property is deprecated.

Number of sampled points found on edge X during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesX() const
```

Remarks

Deprecation notice: Use ERectangleGauge instead.

ERectangleGauge::GetNumSamplesX

This property is deprecated.

Number of sampled points found on edge X during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesX() const
```

Remarks

Deprecation notice: Use ERectangleGauge instead.

ERectangleGauge::GetNumSamplesY

This property is deprecated.

Number of sampled points found on edge Y during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesY() const
```

Remarks

Deprecation notice: Use ERectangleGauge instead.

ERectangleGauge::GetNumSamplesY

This property is deprecated.

Number of sampled points found on edge Y during the measure operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSamplesY() const
```

Remarks

Deprecation notice: Use ERectangleGauge instead.

ERectangleGauge::GetNumSkipRanges

Number of skip ranges in the gauge after a call to ERectangleGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumSkipRanges() const
```

ERectangleGauge::GetNumValidSamples

Number of valid sample points remaining after a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumValidSamples()
```

Remarks

Irrelevant in case of a point location operation.
After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

ERectangleGauge::operator=

Copies all the data from another ERectangleGauge object into the current ERectangleGauge object

Namespace: Euresys::Open_eVision

[C++]

```
ERectangleGauge& operator=(
 const ERectangleGauge& other
 )
```

Parameters

other

ERectangleGauge object to be copied

ERectangleGauge::Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void Plot(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.

height
Height of the plot.

originX
Origin point coordinates of the plot along the X axis.

originY
Origin point coordinates of the plot along the Y axis.

color

The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to ERectangleGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call ERectangleGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERectangleGauge::PlotWithCurrentPen

This method is deprecated.

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void PlotWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawItems

Boolean combination of EPlotItem members, that indicates which items must be displayed.

width

Width of the plot.

height

Height of the plot.

originX

Origin point coordinates of the plot along the X axis.

originY

Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to ERectangleGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call ERectangleGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERectangleGauge::Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

Namespace: Euresys::Open_eVision

[C++]

```
void Process(
 EROIBW8* sourceImage,
 bool daughters
 )

void Process(
 EROIBW8* sourceImage,
 const ERegion& region,
 bool daughters
 )
```

Parameters

sourceImage

Pointer to the source image.

daughters

Flag indicating whether the daughters shapes inherit of the same behavior.

region

Region to use with the source image.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

ERectangleGauge::GetRectangularSamplingArea

ERectangleGauge::SetRectangularSamplingArea

Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRectangularSamplingArea()
void SetRectangularSamplingArea(bool bRectangularSamplingArea)
```

Remarks

By default, this flag is set to true: the sampling area always remains a rectangle.
Setting this property is only useful when the thickness transition parameter is greater than 1.
In fact, when thickness transition parameter is equal to 1, rectangle and parallelogram reduce to the same segment.

ERectangleGauge::RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to ERectangleGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveAllSkipRanges(
 )
```

ERectangleGauge::RemoveSkipRange

After a call to ERectangleGauge::AddSkipRange, removes the skip range with the given index.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveSkipRange(
 const OEV_UINT32 index
 )
```

Parameters

index

Index of the skip range to remove, as returned by ERectangleGauge::AddSkipRange.

ERectangleGauge::GetSamplingStep

ERectangleGauge::SetSamplingStep

Approximate distance between sampled points during a model fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
float GetSamplingStep()
void SetSamplingStep(float f32SamplingStep)
```

Remarks

Irrelevant in case of a point location operation.
To fit a model, a series of point location operations are performed along the model. The point location gauges spacing is given by the sampling step.
By default, the sampling step is set to 5, which means 5 pixels when the field of view is not calibrated, and 5 physical units in case of a calibrated field of view.
Be aware that if the sampling step is too large, the number of sampled points along the model will not be sufficient enough to accurately locate it.

ERectangleGauge::SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void SetMinNumFitSamples(
 int side0,
 int side1,
 int side2,
 int side3
 )
```

Parameters

side0

Minimum number of samples on the *top side* of the rectangle. The default value is 2.

side1

Minimum number of samples on the *left side* of the rectangle. If this value is not specified, it is equal to n32Side0. The default value is 2.

side2

Minimum number of samples on the *bottom side* of the rectangle. If this value is not specified, it is equal to n32Side0. The default value is 2.

side3

Minimum number of samples on the *right side* of the rectangle. If this value is not specified, it is equal to n32Side1. The default value is 2.

Remarks

When the ERectangleGauge::Measure method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

ERectangleGauge::GetSmoothing

ERectangleGauge::SetSmoothing

Number of pixels used for the low-pass filtering operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetSmoothing()
void SetSmoothing(OEV_UINT32 un32Smoothing)
```

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

ERectangleGauge::GetThickness

ERectangleGauge::SetThickness

Number of parallel segments used to extract the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThickness()
void SetThickness(OEV_UINT32 un32Thickness)
```

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

ERectangleGauge::GetThreshold

ERectangleGauge::SetThreshold

Threshold level used to delimit significant peaks in the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 un32Threshold)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value.
To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected.
When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

ERectangleGauge::GetTolerance

ERectangleGauge::SetTolerance

Searching area half thickness of the rectangle fitting gauge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetTolerance()
void SetTolerance(float f32Tolerance)
```

Remarks

A rectangle fitting gauge is fully defined knowing its nominal position (its center coordinates), its nominal size, its rotation angle, and its outline tolerance.
By default, the searching area thickness of the rectangle fitting gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

ERectangleGauge::GetTransitionChoice

ERectangleGauge::SetTransitionChoice

Transition choice.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionChoice GetTransitionChoice()
void SetTransitionChoice(Euresys::Open_eVision::ETransitionChoice eTransitionChoice)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition.
In case of ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd transition choice, set ERectangleGauge::TransitionIndex to specify the desired transition.
By default, the selected transition corresponds to the one with the largest amplitude (ETransitionChoice_LargestAmplitude).

ERectangleGauge::GetTransitionIndex

ERectangleGauge::SetTransitionIndex

Index (from 0 on) of the transition to be retained when the transition choice parameter is set to NthFromBegin or NthFromEnd.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTransitionIndex()

void SetTransitionIndex(OEV_UINT32 un32TransitionIndex)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition.
By default, the first transition is retained (the index value is 0).

ERectangleGauge::GetTransitionType

ERectangleGauge::SetTransitionType

Transition type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionType GetTransitionType()

void SetTransitionType(Euresys::Open_eVision::ETransitionType eTransitionType)
```

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object.
By default, the searched transition type is indifferently a black to white or a white to black transition (ETransitionType_BwOrWb).

ERectangleGauge::GetType

Shape type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EShapeType GetType()
```

ERectangleGauge::GetValid

Flag indicating if at least one valid transition has been found.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetValid()
```

Remarks

A false value means that no measurement has been performed.
A true value means that a transition was found along the sample path inspected with the last call to ERectangleGauge::MeasureSample, and thus a point has been measured.

4.211. ERectangleRegion Class

Manages a complete context for an ERegion shaped like a rectangle.

Base Class:ERegion

Namespace: Euresys::Open_eVision

Methods

Drag	Moves the specified handle to a new position and updates all placement parameters of the region.
ERectangleRegion	Constructs an ERectangleRegion context.
GetAngle	Angle of the region
GetCenter	Center of the region
GetCorners	Corners of the region
GetHeight	Height of the region
GetWidth	Width of the region
HitTest	Detects if the cursor is placed over one of the dragging handles.
Load	Loads the ERectangleRegion. The given ESerializer must have been created for reading.

operator!=	Checks if this ERectangleRegion instance is not strictly equal to another
operator=	Assignment operator.
operator==	Checks if this ERectangleRegion instance is strictly equal to another
Rotate	Creates a new ERectangleRegion by rotating the ERectangleRegion around its center.
Save	Saves the ERectangleRegion. The given ESerializer must have been created for writing.
Scale	Creates a new ERectangleRegion by scaling the ERectangleRegion. The center of the ERectangleRegion remains at the same place.
SetAngle	Angle of the region
SetCenter	Center of the region
SetHeight	Height of the region
SetWidth	Width of the region
Translate	Creates a new ERectangleRegion by translating the ERectangleRegion.

ERectangleRegion::GetAngle

ERectangleRegion::SetAngle

Angle of the region

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
void SetAngle(float angle)
```

ERectangleRegion::GetCenter

ERectangleRegion::SetCenter

Center of the region

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

ERectangleRegion::GetCorners

Corners of the region

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::EPoint> GetCorners() const
```

ERectangleRegion::Drag

Moves the specified handle to a new position and updates all placement parameters of the region.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
: x-coordinate of the mouse cursor.

y
: y-coordinate of the mouse cursor.

zoomX
: Horizontal zoom factor. By default, true scale is used.

zoomY
: Vertical zoom factor. By default, true scale is used.

panX
: Horizontal pan offset. By default, no pan is added.

panY
: Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with ERectangleRegion::HitTest and ERectangleRegion::Drag.

ERectangleRegion::ERectangleRegion

Constructs an ERectangleRegion context.

Namespace: Euresys::Open_eVision

[C++]

```
void ERectangleRegion(
 )

void ERectangleRegion(
 float originX,
 float originY,
 float width,
 float height
 )

void ERectangleRegion(
 const EPoint& origin,
 float width,
 float height
 )

void ERectangleRegion(
 float centerX,
 float centerY,
 float width,
 float height,
 float angle
 )

void ERectangleRegion(
 const EPoint& center,
 float width,
 float height,
 float angle
 )

void ERectangleRegion(
 const ERectangle& rectangle
 )

void ERectangleRegion(
 const ERectangleRegion& other
 )
```

Parameters

originX
The abcissa of the ERectangleRegion's top left corner.
originY
The ordinate of the ERectangleRegion's top left corner.
width
The width of the ERectangleRegion.
height
The height of the ERectangleRegion.
origin
The top left corner of the ERectangleRegion.
centerX
The abcissa of the ERectangleRegion center.
centerY
The ordinate of the ERectangleRegion center.
angle
The angle of the ERectangleRegion.
center
The center of the ERectangleRegion.
rectangle
The result of an ERectangleGauge object.
other
ERectangleRegion context to copy.

Remarks

A ERectangleRegion aligned with the image axes is defined from the top left corner. Otherwise, an oriented ERectangleRegion is defined from its center point.

ERectangleRegion::GetHeight

ERectangleRegion::SetHeight

Height of the region

Namespace: Euresys::Open_eVision

[C++]

```
float GetHeight() const
void SetHeight(float height)
```

ERectangleRegion::HitTest

Detects if the cursor is placed over one of the dragging handles.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EEditionMode HitTest(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
x-coordinate of the mouse cursor.

y
y-coordinate of the mouse cursor.

zoomX
Horizontal zoom factor. By default, true scale is used.

zoomY
Vertical zoom factor. By default, true scale is used.

panX
Horizontal pan offset. By default, no pan is added.

panY
Vertical pan offset. By default, no pan is added.

Remarks

Returns a handle identifier, as defined by EEditionMode.
If zooming and/or panning were used when drawing the region, the same values must be used with ERectangleRegion::HitTest and ERectangleRegion::Drag.

ERectangleRegion::Load

Loads the ERectangleRegion. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ERectangleRegion::operator!=

Checks if this ERectangleRegion instance is not strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const ERectangleRegion& other
 )
```

Parameters

other
Reference to the other ERectangleRegion instance

ERectangleRegion::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
ERectangleRegion& operator=(
 const ERectangleRegion& other
 )
```

Parameters

other
Reference to the ERectangleRegion used for the assignment

ERectangleRegion::operator==

Checks if this ERectangleRegion instance is strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ERectangleRegion& other
 )
```

Parameters

other

Reference to the other ERectangleRegion instance

ERectangleRegion::Rotate

Creates a new ERectangleRegion by rotating the ERectangleRegion around its center.

Namespace: Euresys::Open_eVision

```
[C++]
ERectangleRegion Rotate(
 float angle
 )
```

Parameters

angle

rotation angle

ERectangleRegion::Save

Saves the ERectangleRegion. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

ERectangleRegion::Scale

Creates a new ERectangleRegion by scaling the ERectangleRegion. The center of the ERectangleRegion remains at the same place.

Namespace: Euresys::Open_eVision

```
[C++]
ERectangleRegion Scale(
 float scale
 )
ERectangleRegion Scale(
 float scaleX,
 float scaleY
 )
```

Parameters

scale
Isotropic scale

scaleX
Horizontal scale

scaleY
Vertical scale

ERectangleRegion::Translate

Creates a new ERectangleRegion by translating the ERectangleRegion.

Namespace: Euresys::Open_eVision

```
[C++]
ERectangleRegion Translate(
 float dx,
 float dy
 )
```

Parameters

dx
Horizontal translation in pixel value

dy
Vertical translation in pixel value

ERectangleRegion::GetWidth

ERectangleRegion::SetWidth

Width of the region

Namespace: Euresys::Open_eVision

[C++]

```
float GetWidth() const
void SetWidth(float width)
```

4.212. ERectangleShape Class

Manages a rectangle shape.

Base Class:EShape
Derived Class(es):ERectangleGauge
Namespace: Euresys::Open_eVision

Methods

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copies all the data of the current ERectangleShape object into another ERectangleShape object and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the shape.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
GetAngle	Orientation of the shape.
GetCenter	Center point of the shape.
GetCenterX	Abscissa of the origin point of the shape.
GetCenterY	Ordinate of the origin point of the shape.
GetCorners	Retrieves the coordinates of each corner of a ERectangleShape object.
GetEdges	Retrieves each edge of a ERectangleShape object.
GetMidEdges	Retrieves the center coordinates of each edge of a ERectangleShape object.
GetPoint	Returns the coordinates of a particular point, specified by its location in the ERectangleShape area.
GetScale	Horizontal sensor resolution, in pixels per unit.
GetSizeX	X size of the ERectangleShape
GetSizeY	Y size of the ERectangleShape
GetType	Shape type.
HitTest	Checks if there is a handle under the cursor.

operator=	Copies all the data from another ERectangleShape object into the current ERectangleShape object
SetAngle	Orientation of the shape.
SetCenter	Center point of the shape.
SetCenterXY	Sets the center coordinates of a ERectangleShape object.
SetFromOppositeCorners	Sets the geometric parameters of an ERectangleShape object using two opposed corners.
SetFromOriginMiddleEnd	DEPRECATED (you should use ERectangleShape::SetFromThreeCorners): Sets the geometric parameters of an ERectangleShape object using three corners.
SetFromThreeCorners	Sets the geometric parameters of an ERectangleShape object using three corners.
SetFromTwoPoints	DEPRECATED (you should use ERectangleShape::SetFromOppositeCorners): Sets the geometric parameters of an ERectangleShape object using two opposed corners.
SetRectangle	Sets the parameters of the rectangle according to a known ERectangle object.
SetScale	Horizontal sensor resolution, in pixels per unit.
SetSize	Sets the size of a ERectangleShape object.

ERectangleShape::GetAngle

ERectangleShape::SetAngle

Orientation of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const

void SetAngle(float f32Angle)
```

ERectangleShape::GetCenter

ERectangleShape::SetCenter

Center point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

ERectangleShape::GetCenterX

Abscissa of the origin point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

ERectangleShape::GetCenterY

Ordinate of the origin point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

ERectangleShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

ERectangleShape::CopyTo

Copies all the data of the current ERectangleShape object into another ERectangleShape object and returns it.

Namespace: Euresys::Open_eVision

```
[C++]
void CopyTo(
 ERectangleShape& dest,
 bool bRecursive
 )
ERectangleShape* CopyTo(
 ERectangleShape* dest,
 bool bRecursive
 )
```

Parameters

dest

Pointer to the ERectangleShape object in which the current ERectangleShape object data have to be copied.

bRecursive

true if the children shapes have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new ERectangleShape object will be created and returned.

ERectangleShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

ERectangleShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERectangleShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERectangleShape::GetCorners

Retrieves the coordinates of each corner of a ERectangleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void GetCorners(
 EPoint& xy,
 EPoint& XXy,
 EPoint& xYY,
 EPoint& XXYY
 )
```

Parameters

xy

Coordinates of the lower leftmost corner of the ERectangleShape object.

XXy

Coordinates of the lower rightmost corner of the ERectangleShape object.

xYY

Coordinates of the upper leftmost corner of the ERectangleShape object.

XXYY

Coordinates of the upper rightmost corner of the ERectangleShape object.

ERectangleShape::GetEdges

Retrieves each edge of a ERectangleShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void GetEdges(
 ELine& x,
 ELine& XX,
 ELine& y,
 ELine& YY
 )
```

Parameters

x
Leftmost edge of the ERectangleShape object.

XX
Rightmost edge of the ERectangleShape object.

y
Lower edge of the ERectangleShape object.

YY
Upper edge of the ERectangleShape object.

ERectangleShape::GetMidEdges

Retrieves the center coordinates of each edge of a ERectangleShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void GetMidEdges(
 EPoint& x,
 EPoint& XX,
 EPoint& y,
 EPoint& YY
 )
```

Parameters

x
Center coordinates of the leftmost edge of the ERectangleShape object.

XX
Center coordinates of the rightmost edge of the ERectangleShape object.

y
Center coordinates of the lower edge of the ERectangleShape object.

YY
Center coordinates of the upper edge of the ERectangleShape object.

ERectangleShape::GetPoint

Returns the coordinates of a particular point, specified by its location in the ERectangleShape area.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 float fractionX,
 float fractionY
 )
```

Parameters

fractionX

Point location expressed as a fraction of the ERectangleShape vertical edges (range -1, +1).

fractionY

Point location expressed as a fraction of the ERectangleShape horizontal edges (range -1, +1).

ERectangleShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

ERectangleShape::operator=

Copies all the data from another ERectangleShape object into the current ERectangleShape object

Namespace: Euresys::Open_eVision

[C++]

```
ERectangleShape& operator=(
 const ERectangleShape& other
 )
```

Parameters

other

ERectangleShape object to be copied

ERectangleShape::SetRectangle

Sets the parameters of the rectangle according to a known ERectangle object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetRectangle(const ERectangle& rectangle)
```

ERectangleShape::GetScale

ERectangleShape::SetScale

Horizontal sensor resolution, in pixels per unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetScale() const
void SetScale(float f32Scale)
```

ERectangleShape::SetCenterXY

Sets the center coordinates of a ERectangleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the ERectangleShape object.

centerY

Center coordinates of the ERectangleShape object.

ERectangleShape::SetFromOppositeCorners

Sets the geometric parameters of an ERectangleShape object using two opposed corners.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOppositeCorners(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin

Origin point coordinates of the rectangle.

end

End point coordinates of the rectangle.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

ERectangleShape::SetFromOriginMiddleEnd

This method is deprecated.

DEPRECATED (you should use ERectangleShape::SetFromThreeCorners): Sets the geometric parameters of an ERectangleShape object using three corners.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end
 )
```

Parameters

origin

Origin point coordinates of the rectangle.

middle

Middle point coordinates of the rectangle.

end

End point coordinates of the rectangle.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

ERectangleShape::SetFromThreeCorners

Sets the geometric parameters of an ERectangleShape object using three corners.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromThreeCorners(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end
 )
```

Parameters

origin

Origin point coordinates of the rectangle.

middle

Middle point coordinates of the rectangle.

end

End point coordinates of the rectangle.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

ERectangleShape::SetFromTwoPoints

This method is deprecated.

DEPRECATED (you should use ERectangleShape::SetFromOppositeCorners): Sets the geometric parameters of an ERectangleShape object using two opposed corners.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromTwoPoints(
 const EPoint& origin,
 const EPoint& end
 )
```

Parameters

origin

Origin point coordinates of the rectangle.

end

End point coordinates of the rectangle.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

ERectangleShape::SetSize

Sets the size of a ERectangleShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSize(
 float sizeX,
 float sizeY
 )
```

Parameters

sizeX

Nominal size X of the ERectangleShape object. Default values is 100.

sizeY

Nominal size Y of the ERectangleShape object. Default values is 100.

Remarks

A ERectangleShape object is fully defined knowing its nominal position (given by the coordinates of its center), its nominal size, its rotation angle and its outline tolerance. By default, the width and height values are 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

ERectangleShape::GetSizeX

X size of the ERectangleShape

Namespace: Euresys::Open_eVision

[C++]

```
float GetSizeX() const
```

ERectangleShape::GetSizeY

Y size of the ERectangleShape

Namespace: Euresys::Open_eVision

```
[C++]
float GetSizeY() const
```

ERectangleShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EShapeType GetType()
```

4.213. ERectangularCropper Class

Manages a point cloud cropper in the shape of a rectangular parallelepiped.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Crop	Crops a EPointCloud.
ERectangularCropper	Creates an ERectangularCropper object.
operator=	Assignment operator.
operator==	Equality operator.

ERectangularCropper::Crop

Crops a EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Crop(
 EPointCloud& cloudIn,
 EPointCloud& cloudOut,
 bool invertCrop
 )
```

Parameters

cloudIn

Cloud to be cropped.

cloudOut

Cropped cloud.

invertCrop

Indicates if the points kept must be the points inside (true) or outside (false) the rectangular parallelepiped.

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

ERectangularCropper::ERectangularCropper

Creates an ERectangularCropper object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ERectangularCropper(
 const E3DPoint& center,
 float xSize,
 float ySize,
 float zSize,
 float roll,
 float pitch,
 float yaw
 )

void ERectangularCropper(
 const ERectangularCropper& other
 )

void ERectangularCropper(
 const E3DBox& box
 )
```

Parameters

center

Center of the rectangular parallelepiped.

xSize

Size along the X axis before rotation of the rectangular parallelepiped.

ySize

Size along the Y axis before rotation of the rectangular parallelepiped.

zSize

Size along the Z axis before rotation of the rectangular parallelepiped.

roll

Roll (rotation along the X axis) of the rectangular parallelepiped.

pitch
Pitch (rotation along the Y axis) of the rectangular parallelepiped.
yaw
Yaw (rotation along the Z axis) of the rectangular parallelepiped.
other
Reference ERectangularCropper used for the initialization.
box
E3DBox used to delimit the zone the cropper will crop or keep.

ERectangularCropper::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ERectangularCropper& operator=(
 const ERectangularCropper& other
 )
```

Parameters

other
An other ERectangularCropper.

ERectangularCropper::operator==

Equality operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const ERectangularCropper& other
 )
```

Parameters

other
An other ERectangularCropper.

4.214. EReferenceImageSegmenter Class

Segments an image using a pixel-by-pixel single threshold given as an image.

Remarks

This segmenter is applicable to EROIBW8, EROIBW16 and EROIC24 images. It produces coded images with two layers. The threshold is defined for each pixel individually by means of a reference image of the same type as the source image.

For grayscales images, the White layer (usually, with index 1) contains unmasked pixels having a gray value in a range defined by the gray value of the respective pixel in the reference image and the white color.

For RGB color images, the White layer (usually, with index 1) contains unmasked pixels having a color inside the cube of the RGB color space defined by the color of the respective pixel in the reference image and the white color (255,255,255).

The Black layer (usually, with index 0) contains the remaining unmasked pixels.

Base Class:ETwoLayersImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetReferenceImageBW16	Reference image for the segmentation of EROIBW16 images.
GetReferenceImageBW8	Reference image for the segmentation of EROIBW8 images.
GetReferenceImageC24	Reference image for the segmentation of EROIC24 images.
Load	Load the EReferenceImageSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EReferenceImageSegmenter configuration. The given ESerializer must have been created for writing.
SetBlackLayerEncoded	Black layer encoding status.
SetBlackLayerIndex	Index of the black layer in the destination coded image.
SetReferenceImageBW16	Reference image for the segmentation of EROIBW16 images.
SetReferenceImageBW8	Reference image for the segmentation of EROIBW8 images.
SetReferenceImageC24	Reference image for the segmentation of EROIC24 images.
SetWhiteLayerEncoded	White layer encoding status.
SetWhiteLayerIndex	Index of the white layer in the destination coded image.

EReferenceImageSegmenter::SetBlackLayerEncoded

Black layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void SetBlackLayerEncoded(bool encode)
```

EReferenceImageSegmenter::SetBlackLayerIndex

Index of the black layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void SetBlackLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the black layer.

EReferenceImageSegmenter::Load

Load the EReferenceImageSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EReferenceImageSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool operator==(
 const EReferenceImageSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to.

EReferenceImageSegmenter::GetReferenceImageBW16

EReferenceImageSegmenter::SetReferenceImageBW16

Reference image for the segmentation of EROIBW16 images.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
const EROIBW16* GetReferenceImageBW16() const
void SetReferenceImageBW16(const EROIBW16* image)
```

Remarks

Note that the image is copied internally.

EReferenceImageSegmenter::GetReferenceImageBW8

EReferenceImageSegmenter::SetReferenceImageBW8

Reference image for the segmentation of EROIBW8 images.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
const EROIBW8* GetReferenceImageBW8() const
void SetReferenceImageBW8(const EROIBW8* image)
```

Remarks

Note that the image is copied internally.

EReferenceImageSegmenter::GetReferenceImageC24

EReferenceImageSegmenter::SetReferenceImageC24

Reference image for the segmentation of EROIC24 images.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
const EROIC24* GetReferenceImageC24() const
void SetReferenceImageC24(const EROIC24* image)
```

Remarks

Note that the image is copied internally.

EReferenceImageSegmenter::Save

Save the EReferenceImageSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EReferenceImageSegmenter::SetWhiteLayerEncoded

White layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void SetWhiteLayerEncoded(bool encode)
```

EReferenceImageSegmenter::SetWhiteLayerIndex

Index of the white layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void SetWhiteLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the white layer.

4.215. ERegion Class

Manages a complete context for a ERegion (Abritrary Shaped ROI)

Derived Class(es):ECircleRegionEEllipseRegionEPolygonRegionERectangleRegion

Namespace: Euresys::Open_eVision

Methods

Contour	Creates a new ERegion containing the contour of the ERegion.
CropRuns	Creates a new ERegion by cropping the ERegion runs within a given area.
Drag	Moves the specified handle to a new position and updates all placement parameters of the region.
Draw	Draws the ERegion shape.
DrawContour	Draws the ERegion contour.
DrawContourWithCurrentPen	Draws the ERegion contour with the current pen.
DrawHandles	Draws the region handles.
DrawHandlesWithCurrentPen	Draws the region handles with the current pen.
DrawWithCurrentPen	Draws the ERegion area.
ERegion	Constructs an ERegion context.
GetArea	Area of the ERegion.
GetBoundingBoxHeight	Bounding box height.
GetBoundingBoxOrgX	Bounding box origin abscissa.
GetBoundingBoxOrgY	Bounding box origin ordinate.
GetBoundingBoxWidth	Bounding box width.
GetEditionMode	Graphical edition mode

GetRuns	Runs of the region
Grow	Creates a new ERegion by growing the ERegion runs with a dilation using a circular structuring element.
HitTest	Detects if the cursor is placed over one of the dragging handles.
Intersection	Creates a new ERegion by intersecting two ERegion.
IsPointInRegion	Checks if a point is inside the region
Load	Loads the ERegion. The given ESerializer must have been created for reading.
operator!=	Checks if this ERegion instance is not strictly equal to another (order of the runs included)
operator=	Assignment operator.
operator==	Checks if this ERegion instance is strictly equal to another (order of the runs included)
Prepare	Computes the values necessary for the ERegion to be used during processing.
Save	Saves the ERegion. The given ESerializer must have been created for writing.
SetEditionMode	Graphical edition mode
SetRuns	Runs of the region
Shrink	Creates a new ERegion by shrinking the ERegion runs with an erosion using a circular structuring element.
Subtraction	Creates a new ERegion by substracting one ERegion from another.
ToImage	Exports the ERegion to a mask-type image.
TranslateRuns	Creates a new ERegion by translating the region runs.
Union	Creates a new ERegion by combining two ERegion.

ERegion::GetArea

Area of the ERegion.

Namespace: Euresys::Open_eVision

[C++]

```
int GetArea() const
```

Remarks

The number of pixels of the ERegion

ERegion::GetBoundingBoxHeight

Bounding box height.

Namespace: Euresys::Open_eVision

```
[C++]
int GetBoundingBoxHeight() const
```

Remarks

The height is the height of the upright rectangle encompassing the ERun.

ERegion::GetBoundingBoxOrgX

Bounding box origin abscissa.

Namespace: Euresys::Open_eVision

```
[C++]
int GetBoundingBoxOrgX() const
```

Remarks

The origin is the top left corner of the upright rectangle encompassing the ERun.

ERegion::GetBoundingBoxOrgY

Bounding box origin ordinate.

Namespace: Euresys::Open_eVision

```
[C++]
int GetBoundingBoxOrgY() const
```

Remarks

The origin is the top left corner of the upright rectangle encompassing the ERun.

ERegion::GetBoundingBoxWidth

Bounding box width.

Namespace: Euresys::Open_eVision

```
[C++]
int GetBoundingBoxWidth() const
```

Remarks

The width is the width of the upright rectangle encompassing the ERun.

ERegion::Contour

Creates a new ERegion containing the contour of the ERegion.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion Contour(
 int thickness,
 bool centered
 )
```

Parameters

thickness

The thickness of the returned contour of the ERegion border. A negative value will compute the inner contour and a positive one will compute the outer contour using the absolute value of thickness as thickness.

centered

If true, the contour will be centered with the border of the ERegion. If false, the contour will be either the inner or outer one depending on the sign of thickness

Remarks

If thickness is even and centered is true, the thickness will be increased by 1.

ERegion::CropRuns

Creates a new ERegion by cropping the ERegion runs within a given area.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion CropRuns(
 int orgX,
 int orgY,
 int width,
 int height
 )
```

Parameters

orgX

X origin of the cropping area.

orgY

Y origin of the cropping area.

width

Width of the cropping area.

height

Height of the cropping area.

ERegion::Drag

Moves the specified handle to a new position and updates all placement parameters of the region.

Namespace: Euresys::Open_eVision

```
[C++]
void Drag(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

x-coordinate of the mouse cursor.

y

y-coordinate of the mouse cursor.

zoomX

Horizontal zoom factor. By default, true scale is used.

zoomY

Vertical zoom factor. By default, true scale is used.

panX

Horizontal pan offset. By default, no pan is added.

panY

Vertical pan offset. By default, no pan is added.

Remarks

If zooming and/or panning were used when drawing the region, the same values must be used with ERegion::HitTest and ERegion::Drag.

ERegion::Draw

Draws the ERegion shape.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* drawAdapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* drawAdapter,
 const ERGBColor& color,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the ERegion.

opacity

Opacity of the drawn area (range: 0.0 to 1.0).

graphicContext

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegion::DrawContour

Draws the ERegion contour.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawContour(
 EDrawAdapter* drawAdapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawContour(
 EDrawAdapter* drawAdapter,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawContour(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawContour(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

A pointer to an EDrawAdapter (like the EWindowsDrawAdapter).

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the ERegion.

graphicContext

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegion::DrawContourWithCurrentPen

This method is deprecated.

Draws the ERegion contour with the current pen.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawContourWithCurrentPen(
 HDC hdc,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hdc

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegion::DrawHandles

Draws the region handles.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawHandles(
 EDrawAdapter* drawAdapter,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHandles(
 EDrawAdapter* drawAdapter,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHandles(
 HDC hdc,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawHandles(
 HDC hdc,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the region.

hdc

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegion::DrawHandlesWithCurrentPen

This method is deprecated.

Draws the region handles with the current pen.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawHandlesWithCurrentPen(
 HDC hdc,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

hdc

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegion::DrawWithCurrentPen

This method is deprecated.

Draws the ERegion area.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

opacity

Opacity of the drawn area (range: 0.0 to 1.0).

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegion::GetEditionMode

ERegion::SetEditionMode

Graphical edition mode

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EEditionMode GetEditionMode() const
void SetEditionMode(Euresys::Open_eVision::EEditionMode mode)
```

ERegion::ERegion

Constructs an ERegion context.

Namespace: Euresys::Open_eVision

[C++]

```
void ERegion(
 )
void ERegion(
 const ERegion& other
 )
void ERegion(
 const EROIBW8& roi,
 EBW8 threshold
 )
void ERegion(
 const EROIBW16& roi,
 EBW16 threshold
 )
void ERegion(
 const std::vector<Euresys::Open_eVision::ERun>& runs
 )
```

Parameters

other

ERegion context to copy.

roi

Mask ROI.

threshold

Mask Threshold (a pixel belongs to the ERegion if its value is >= threshold).

runs

List of ERun.

ERegion::Grow

Creates a new ERegion by growing the ERegion runs with a dilation using a circular structuring element.

Namespace: Euresys::Open_eVision

```
[C++]

ERegion Grow(
 int radius
 )
```

Parameters

radius

-

ERegion::HitTest

Detects if the cursor is placed over one of the dragging handles.

Namespace: Euresys::Open_eVision

```
[C++]

Euresys::Open_eVision::EEditionMode HitTest(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x
x-coordinate of the mouse cursor.

y
y-coordinate of the mouse cursor.

zoomX
Horizontal zoom factor. By default, true scale is used.

zoomY
Vertical zoom factor. By default, true scale is used.

panX
Horizontal pan offset. By default, no pan is added.

panY
Vertical pan offset. By default, no pan is added.

Remarks

Returns a handle identifier, as defined by EEditionMode.
If zooming and/or panning were used when drawing the region, the same values must be used with ERegion::HitTest and ERegion::Drag.

ERegion::Intersection

Creates a new ERegion by intersecting two ERegion.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion Intersection(
 const ERegion& region1,
 const ERegion& region2
 )
```

Parameters

region1
First region.

region2
Second region.

ERegion::IsPointInRegion

Checks if a point is inside the region

Namespace: Euresys::Open_eVision

[C++]

```
bool IsPointInRegion(
 const EPoint& point
 )
```

Parameters

point

The point to check.

Remarks

The region must have been prepared before calling this method.

ERegion::Load

Loads the ERegion. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

ERegion::operator!=

Checks if this ERegion instance is not strictly equal to another (order of the runs included)

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const ERegion& other
 )
```

Parameters

other

Reference to the other ERegion instance

ERegion::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion& operator=(
 const ERegion& other
 )
```

Parameters

other

Reference to the ERegion used for the assignment.

ERegion::operator==

Checks if this ERegion instance is strictly equal to another (order of the runs included)

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ERegion& other
 )
```

Parameters

other

Reference to the other ERegion instance

ERegion::Prepare

Computes the values necessary for the ERegion to be used during processing.

Namespace: Euresys::Open_eVision

```
[C++]
void Prepare(
 const EROIBW8& roi
 )

void Prepare(
 const EROIBW16& roi
 )

void Prepare(
 const EROIBW32& roi
 )
```

```
void Prepare(
 const EROIBW32f& roi
 )

void Prepare(
 const EROIC24& roi
 )

void Prepare(
 const EROIC24A& roi
 )

void Prepare(
 const EROIC15& roi
 )

void Prepare(
 const EROIC16& roi
 )

void Prepare(
 const EROIC48& roi
 )

void Prepare(
 int width,
 int height
 )

void Prepare(
 int orgX,
 int orgY,
 int width,
 int height
 )
```

Parameters

roi

Destination or source EBaseROI.

width

Width of the source or destination context.

height

Height of the source or destination context.

orgX

X-Axis origin of the source or destination context.

orgY

Y-Axis origin of the source or destination context.

Remarks

This method should be called once after the ERegion has been parameterized and before the ERegion is used.
If necessary,it will be done automatically before any usage but it will increase the processing time.

ERegion::GetRuns

ERegion::SetRuns

Runs of the region

Namespace: Euresys::Open_eVision

[C++]

```
std::vector<Euresys::Open_eVision::ERun> GetRuns() const
void SetRuns(const std::vector<Euresys::Open_eVision::ERun>& runs)
```

ERegion::Save

Saves the ERegion. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

ERegion::Shrink

Creates a new ERegion by shrinking the ERegion runs with an erosion using a circular structuring element.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion Shrink(
 int radius
 )
```

Parameters

radius

-

ERegion::Subtraction

Creates a new ERegion by substracting one ERegion from another.

Namespace: Euresys::Open_eVision

```
[C++]
ERegion Subtraction(
 const ERegion& region1,
 const ERegion& region2
 )
```

Parameters

region1

Original region.

region2

Region to substract.

ERegion::ToImage

Exports the ERegion to a mask-type image.

Namespace: Euresys::Open_eVision

```
[C++]
void ToImage(
 EImageC24& img,
 EC24 background,
 EC24 foreground
 )

void ToImage(
 EImageC24A& img,
 EC24A background,
 EC24A foreground
 )

void ToImage(
 EImageBW8& img,
 EBW8 background,
 EBW8 foreground
 )
```

```
void ToImage(
 EImageBW16& img,
 EBW16 background,
 EBW16 foreground
 )
```

Parameters

img
Destination image.

background
Background color.

foreground
Foreground color.

ERegion::TranslateRuns

Creates a new ERegion by translating the region runs.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion TranslateRuns(
 int dx,
 int dy
 )
```

Parameters

dx
x component of the translation vector.

dy
y component of the translation vector.

ERegion::Union

Creates a new ERegion by combining two ERegion.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion Union(
 const ERegion& region1,
 const ERegion& region2
 )
```

Parameters

region1
First region.

region2
Second region.

4.216. ERegionFreeHandPainter Class

Manages a complete context for a ERegionFreeHandPainter. This class allows to paint a region using a free hand method.

Namespace: Euresys::Open_eVision

Methods

ClearCanvas	Clear the canvas.
Draw	Draws the ERegionFreeHandPainter shape.
DrawContour	Draws the ERegionFreeHandPainter contour.
ERegionFreeHandPainter	Constructs an ERegionFreeHandPainter context.
Paint	Paints the region by applying the brush on the canvas at the designed coordinates.
RetrieveRegion	Retrieves the painted region.
SetBrush	Sets the brush to use to paint the region.
SetCanvasSize	Sets the canvas size.

ERegionFreeHandPainter::SetBrush

Sets the brush to use to paint the region.

Namespace: Euresys::Open_eVision

```
[C++]
void SetBrush(const ERegion& brush)
```

Remarks

Any region can be used as a brush. By default, the brush is a 8-pixel radius circle.

ERegionFreeHandPainter::ClearCanvas

Clear the canvas.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearCanvas(
 )
```

Remarks

The canvas is the area on which to paint the region.

ERegionFreeHandPainter::Draw

Draws the ERegionFreeHandPainter shape.

Namespace: Euresys::Open_eVision

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 const ERGBColor& color,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float opacity,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

opacity

Opacity of the drawn area (range: 0.0 to 1.0).

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the ERegionFreeHandPainter.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegionFreeHandPainter::DrawContour

Draws the ERegionFreeHandPainter contour.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawContour(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawContour(
 EDrawAdapter* graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawContour(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawContour(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

color

The color in which to draw the ERegionFreeHandPainter.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERegionFreeHandPainter::ERegionFreeHandPainter

Constructs an ERegionFreeHandPainter context.

Namespace: Euresys::Open_eVision

```
[C++]
void ERegionFreeHandPainter(
 )
```

ERegionFreeHandPainter::Paint

Paints the region by applying the brush on the canvas at the designed coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void Paint(
 int x,
 int y,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

x

X position on which to apply the brush.

y

Y position on which to apply the brush.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

The canvas is the area on which to paint the region. The center of the brush region bounding box will be applied at the given location. By default, the brush is a 8-pixel radius circle.

ERegionFreeHandPainter::RetrieveRegion

Retrieves the painted region.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion& RetrieveRegion(
 )
```

ERegionFreeHandPainter::SetCanvasSize

Sets the canvas size.

Namespace: Euresys::Open_eVision

[C++]

```
void SetCanvasSize(
 int width,
 int height
 )

void SetCanvasSize(
 const EBaseROI& roi
 )
```

Parameters

width
Width of the canvas.

height
Height of the canvas.

roi
ROI/Image from which the canvas size will be adapted.

Remarks

The canvas is the area on which to paint the region.

4.217. EROIBW1 Class

This class is deprecated.

The EROIBW1 class is used to represent rectangular regions of interest inside BW1 black and white images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageBW1

Namespace: Euresys::Open_eVision

Methods

EROIBW1	Constructs a EROIBW1 object.
GetBitIndex	Gets the index of the bit at the given position in the image.

GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIBW1::EROIBW1

This method is deprecated.

Constructs a EROIBW1 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIBW1(
 )
void EROIBW1(
 const EROIBW1& other
 )
```

Parameters

other

-

EROIBW1::GetFirstSubROI

This property is deprecated.

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW1* GetFirstSubROI()
```

EROIBW1::GetBitIndex

This method is deprecated.

Gets the index of the bit at the given position in the image.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT64 GetBitIndex(
 int x,
 int y
 )
```

Parameters

x

-

y

-

EROIBW1::GetNextROI

This method is deprecated.

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW1* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIBW1::GetPixel

This method is deprecated.

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
EBW1 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW1 and EROIBW1 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW1::GetNextSiblingROI

This property is deprecated.

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW1* GetNextSiblingROI()
```

EROIBW1::operator=

This method is deprecated.

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW1& operator=(
 const EROIBW1& other
 )
```

Parameters

other

-

EROIBW1::GetParent

This property is deprecated.

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW1* GetParent()
```

EROIBW1::SetPixel

This method is deprecated.

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixel(
 EBW1 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW1 and EROIBW1 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW1::GetTopParent

This property is deprecated.

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW1* GetTopParent()
```

4.218. EROIBW16 Class

The EROIBW16 class is used to represent rectangular regions of interest inside BW16 gray-level images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageBW16

Namespace: Euresys::Open_eVision

Methods

EROIBW16	Constructs a EROIBW16 image.
GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIBW16::EROIBW16

Constructs a EROIBW16 image.

Namespace: Euresys::Open_eVision

[C++]

```
void EROIBW16(
 )

void EROIBW16(
 const EROIBW16& other
 )
```

Parameters

other

-

EROIBW16::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW16* GetFirstSubROI()
```

EROIBW16::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW16* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIBW16::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
EBW16 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW16 and EROIBW16 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW16::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW16* GetNextSiblingROI()
```

EROIBW16::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW16& operator=(
 const EROIBW16& other
 )
```

Parameters

other

-

EROIBW16::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW16* GetParent()
```

EROIBW16::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixel(
 EBW16 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW16 and EROIBW16 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW16::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW16* GetTopParent()
```

4.219. EROIBW32 Class

The EROIBW32 class is used to represent rectangular regions of interest inside BW32 gray-level images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageBW32

Namespace: Euresys::Open_eVision

Methods

EROIBW32	Constructs the EROIBW32 object.
GetFirstSubROI	See GetFirstSubROI in the derived classes

GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIBW32::EROIBW32

Constructs the EROIBW32 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIBW32(
 )
void EROIBW32(
 const EROIBW32& other
 )
```

Parameters

other

-

EROIBW32::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32* GetFirstSubROI()
```

EROIBW32::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW32* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIBW32::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
EBW32 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW32 and EROIBW32 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW32::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW32* GetNextSiblingROI()
```

EROIBW32::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32& operator=(
 const EROIBW32& other
 )
```

Parameters

other

-

EROIBW32::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32* GetParent()
```

EROIBW32::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EBW32 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW32 and EROIBW32 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW32::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW32* GetTopParent()
```

4.220. EROIBW32f Class

The EROIBW32f class is used to represent rectangular regions of interest inside BW32f gray-level images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageBW32f

Namespace: Euresys::Open_eVision

Methods

Draw	Draws an ROI/image in a device context.
DrawFrame	Draws a rectangular frame around an image or ROI.
DrawFrameWithCurrentPen	Draws a rectangular frame around an image or ROI.
EROIBW32f	Constructs the EROIBW32f object.
GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.

GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
Save	Saves the EROIBW32f object to the given file.
SaveJpeg	As the JPEG format does not work with floating point data, calling this function throws an exception.
SaveJpeg2K	As the JPEG 2000 format does not work with floating point data, calling this function throws an exception.
SavePng	As the PNG format does not work with floating point data, calling this function throws an exception.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIBW32f::Draw

Draws an ROI/image in a device context.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

c24Vector

When supplied, this parameter allows using a LUT that maps from BW8 to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from BW8 to BW8 when drawing.

Remarks

An ROI/image can be drawn (its pixels rendered) using a device context.
The horizontal and vertical zooming factors can be different and must be contained in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.)

EROIBW32f::DrawFrame

Draws a rectangular frame around an image or ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawFrame(
 HDC graphicContext,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawFrame(
 HDC graphicContext,
 Euresys::Open_eVision::EFramePosition framePosition,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawFrame(
 HDC graphicContext,
 const ERGBColor& color,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void DrawFrame(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EFramePosition framePosition,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void DrawFrame(
 EDrawAdapter* graphicContext,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

handles

true if handles are to be drawn.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

framePosition

Positioning of the frame relative to the ROI.

color

Color in which to draw the frame.

Remarks

A frame can be drawn (using a device context) around an image or ROI, possibly with 9 sizing handles.
A suitable default pen is used (see EROIBW32f::DrawFrameWithCurrentPen if you wish to use the pen currently selected into the device context).
Zooming and panning are possible. Please note that panning is applied *before* zooming.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.)

EROIBW32f::DrawFrameWithCurrentPen

Draws a rectangular frame around an image or ROI.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawFrameWithCurrentPen(
 HDC graphicContext,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void DrawFrameWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EFramePosition framePosition,
 bool handles,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

handles

true if handles are to be drawn.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in true scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

framePosition

Positioning of the frame relative to the ROI.

Remarks

A frame can be drawn (using a device context) around an image or ROI, possibly with 9 sizing handles.
The current device context pen is used. Zooming and panning are possible. Please note that panning is applied *before* zooming.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.)

EROIBW32f::EROIBW32f

Constructs the EROIBW32f object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIBW32f(
 )
void EROIBW32f(
 const EROIBW32f& other
 )
```

Parameters

other

-

EROIBW32f::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32f* GetFirstSubROI()
```

EROIBW32f::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32f* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIBW32f::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
EBW32f GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW32f and EROIBW32f functions.

EROIBW32f::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32f* GetNextSiblingROI()
```

EROIBW32f::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW32f& operator=(
 const EROIBW32f& other
 )
```

Parameters

other

-

EROIBW32f::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW32f* GetParent()
```

EROIBW32f::Save

Saves the EROIBW32f object to the given file.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

path

The full path of the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.
For floating point images, only tiff and Euresys proprietary file format are supported.

Remarks

By default (if no format is specified), the file format is determined from the file extension.
If a serializer is used, then the Euresys proprietary file format is used. This format preserves attributes and sub-ROIs.
See Supported Image File Types for details about supported files.
See Image File Access - Save, Load - for details about file format and compatibility.
For floating point images, only tiff and Euresys proprietary file format are supported.

EROIBW32f::SaveJpeg

As the JPEG format does not work with floating point data, calling this function throws an exception.

Namespace: Euresys::Open_eVision

[C++]

```
void SaveJpeg(
 const std::string& path,
 int quality
 )
```

Parameters

path

-

quality

-

EROIBW32f::SaveJpeg2K

As the JPEG 2000 format does not work with floating point data, calling this function throws an exception.

Namespace: Euresys::Open_eVision

[C++]

```
void SaveJpeg2K(
 const std::string& path,
 int quality
 )
```

Parameters

path

-

quality

-

EROIBW32f::SavePng

As the PNG format does not work with floating point data, calling this function throws an exception.

Namespace: Euresys::Open_eVision

[C++]

```
void SavePng(
 const std::string& path,
 int compression
 )
```

Parameters

path

-

compression

-

EROIBW32f::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EBW32f value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW32f and EROIBW32f functions.

EROIBW32f::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EImageBW32f* GetTopParent()
```

4.221. EROIBW8 Class

The EROIBW8 class is used to represent rectangular regions of interest inside BW8 gray-level images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageBW8

Namespace: Euresys::Open_eVision

Methods

EROIBW8	Constructs the EROIBW8 object.
GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIBW8::EROIBW8

Constructs the EROIBW8 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIBW8(
 )
void EROIBW8(
 const EROIBW8& other
 )
```

Parameters

other

-

EROIBW8::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW8* GetFirstSubROI()
```

EROIBW8::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW8* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIBW8::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
EBW8 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW8 and EROIBW8 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW8::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW8* GetNextSiblingROI()
```

EROIBW8::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW8& operator=(
 const EROIBW8& other
 )
```

Parameters

other

-

EROIBW8::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIBW8* GetParent()
```

EROIBW8::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixel(
 EBW8 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIBW8 and EROIBW8 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIBW8::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageBW8* GetTopParent()
```

4.222. EROIC15 Class

The EROIC15 class is used to represent rectangular regions of interest inside C15 color images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageC15

Namespace: Euresys::Open_eVision

Methods

EROIC15	Constructs the EROIC15 object.
GetFirstSubROI	See GetFirstSubROI in the derived classes

GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIC15::EROIC15

Constructs the EROIC15 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EROIC15(
 )
void EROIC15(
 const EROIC15& other
 )
```

Parameters

other

-

EROIC15::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIC15* GetFirstSubROI()
```

EROIC15::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC15* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIC15::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
EC15 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC15 and EROIC15 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC15::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC15* GetNextSiblingROI()
```

EROIC15::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC15& operator=(
 const EROIC15& other
 )
```

Parameters

other

-

EROIC15::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC15* GetParent()
```

EROIC15::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC15 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC15 and EROIC15 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC15::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageC15* GetTopParent()
```

4.223. EROIC16 Class

The EROIC16 class is used to represent rectangular regions of interest inside C16 color images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageC16

Namespace: Euresys::Open_eVision

Methods

EROIC16	Constructs the EROIC16 object.
GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.

operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIC16::EROIC16

Constructs the EROIC16 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIC16(
 )
void EROIC16(
 const EROIC16& other
 )
```

Parameters

other

-

EROIC16::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC16* GetFirstSubROI()
```

EROIC16::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC16* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIC16::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
EC16 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC16 and EROIC16 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC16::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC16* GetNextSiblingROI()
```

EROIC16::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC16& operator=(
 const EROIC16& other
 )
```

Parameters

other

-

EROIC16::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC16* GetParent()
```

EROIC16::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC16 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC16 and EROIC16 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC16::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EImageC16* GetTopParent()
```

4.224. EROIC24 Class

The EROIC24 class is used to represent rectangular regions of interest inside C24 color images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageC24

Namespace: Euresys::Open_eVision

Methods

EROIC24	Constructs the EROIC24 object.
GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIC24::EROIC24

Constructs the EROIC24 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIC24(
 )
void EROIC24(
 const EROIC24& other
 )
```

Parameters

other

-

EROIC24::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC24* GetFirstSubROI()
```

EROIC24::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC24* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIC24::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
EC24 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC24 and EROIC24 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

[C++]

```
EROIC24* GetNextSiblingROI()
```

EROIC24::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
EROIC24& operator=(
 const EROIC24& other
 )
```

Parameters

other

-

EROIC24::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EROIC24* GetParent()
```

EROIC24::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixel(
 EC24 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC24 and EROIC24 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageC24* GetTopParent()
```

4.225. EROIC24A Class

The EROIC24A class is used to represent rectangular regions of interest inside C24A color images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageC24A

Namespace: Euresys::Open_eVision

Methods

EROIC24A	Constructs the EROIC24A object.
GetFirstSubROI	See GetFirstSubROI in the derived classes
GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIC24A::EROIC24A

Constructs the EROIC24A object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIC24A(
 )
void EROIC24A(
 const EROIC24A& other
 )
```

Parameters

other

-

EROIC24A::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIC24A* GetFirstSubROI()
```

EROIC24A::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

[C++]

```
EROIC24A* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIC24A::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
EC24A GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC24A and EROIC24A functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24A::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC24A* GetNextSiblingROI()
```

EROIC24A::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC24A& operator=(
 const EROIC24A& other
 )
```

Parameters

other

-

EROIC24A::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC24A* GetParent()
```

EROIC24A::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPixel(
 EC24A value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC24A and EROIC24A functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC24A::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageC24A* GetTopParent()
```

4.226. EROIC48 Class

The EROIC48 class is used to represent rectangular regions of interest inside C48 color images. See ROIs.

Base Class:EBaseROI

Derived Class(es):EImageC48

Namespace: Euresys::Open_eVision

Methods

EROIC48	Constructs the EROIC48 object.
GetFirstSubROI	See GetFirstSubROI in the derived classes

GetNextROI	See GetNextROI in the derived classes
GetNextSiblingROI	This method returns the ROI that is the next sibling of this ROI.
GetParent	Returns the hierarchical parent of this object.
GetPixel	Allows reading a single pixel value in the ROI or image.
GetTopParent	Returns the top parent of this object.
operator=	Assignment operator.
SetPixel	Allows writing a single pixel value in the ROI or image.

EROIC48::EROIC48

Constructs the EROIC48 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EROIC48(
 )
void EROIC48(
 const EROIC48& other
 )
```

Parameters

other

-

EROIC48::GetFirstSubROI

See GetFirstSubROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC48* GetFirstSubROI()
```

EROIC48::GetNextROI

See GetNextROI in the derived classes

Namespace: Euresys::Open_eVision

```
[C++]
EROIC48* GetNextROI(
 EBaseROI* startROI
 )
```

Parameters

startROI

-

EROIC48::GetPixel

Allows reading a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
EC48 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC48 and EROIC48 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

EROIC48::GetNextSiblingROI

This method returns the ROI that is the next sibling of this ROI.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC48* GetNextSiblingROI()
```

EROIC48::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC48& operator=(
 const EROIC48& other
 )
```

Parameters

other

-

EROIC48::GetParent

Returns the hierarchical parent of this object.

Namespace: Euresys::Open_eVision

```
[C++]
EROIC48* GetParent()
```

EROIC48::SetPixel

Allows writing a single pixel value in the ROI or image.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPixel(
 EC48 value,
 int x,
 int y
 )
```

Parameters

value

value to set the pixel to.

x

Offset of the pixel on the x axis.

y

Offset of the pixel on the y axis.

Remarks

Although coordinates outside of the ROI can be supplied, this function will raise an error condition if the coordinates are outside of the top parent image. It should be noted that calling this function several thousand times can be slow. The recommended way to access pixel content in an image or ROI is to use the EROIC48 and EROIC48 functions. For BW8 images/ROIs, using the C++ API only, it is possible to access pixels data faster through an intermediate object: EBW8PixelAccessor.

ERoIC48::GetTopParent

Returns the top parent of this object.

Namespace: Euresys::Open_eVision

[C++]

```
EImageC48* GetTopParent()
```

4.227. ERotatedBoundingBox Class

This class represents a rotated bounding box.

Remarks

The rotated bounding box is a rotated, rectangular surface. Its coordinates are floating-point, which makes this class appropriate to handle sub-pixel surfaces.

Namespace: Euresys::Open_eVision

Methods

Draw	Draws the rotated bounding box.
DrawWithCurrentPen	Draws the rotated bounding box.
ERotatedBoundingBox	Constructor of the rotated bounding box.
GetAngle	Returns the angle of the bounding box (in the current angle units).
GetCenter	Returns the coordinate of the center of the bounding box.
GetCenterX	Returns the abscissa of the center of the bounding box.
GetCenterY	Returns the ordinate of the center of the bounding box.

GetHeight	Returns the height of the bounding box.
GetMajorAxis	Returns the major axis of the ERotatedBoundingBox as an ELine.
GetQuadrangle	Returns the coordinates of the four corners of the bounding box.
GetUpperLeftCorner	Returns the coordinates of the upper left corner of the bounding box (the position with the minimum Y and then minimum X).
GetWidth	Returns the width of the bounding box.
LocalToGlobalBox	Transforms a (local) rotated bounding box, as another (global) rotated bounding box whose coordinates are defined relatively to the current rotated bounding box.
LocalToGlobalPoint	Transforms a (local) point, as another (global) point whose coordinates are defined relatively to the current rotated bounding box.
operator=	Assignment operator.
operator==	Checks if this ERotatedBoundingBox instance is strictly equal to another.
Rotate	Applies a rotation to the bounding box. The bounding box center is the center of rotation.
Scale	Applies a scale to the size of the bounding box.
Translate	Applies a translation on the center of the bounding box.

ERotatedBoundingBox::GetAngle

Returns the angle of the bounding box (in the current angle units).

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
```

ERotatedBoundingBox::GetCenter

Returns the coordinate of the center of the bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
```

ERotatedBoundingBox::GetCenterX

Returns the abscissa of the center of the bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

ERotatedBoundingBox::GetCenterY

Returns the ordinate of the center of the bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

ERotatedBoundingBox::Draw

Draws the rotated bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the bounding box are drawn.

color

The color in which to draw the overlay.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERotatedBoundingBox::DrawWithCurrentPen

This method is deprecated.

Draws the rotated bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 bool drawDiagonals
 )
```

Parameters

graphicContext

Graphic context on which to draw.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning factor. By default, no panning occurs.

drawDiagonals

Specifies whether or not the diagonals of the bounding box are drawn.

Remarks

Drawing is done in the device context associated to the desired window.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ERotatedBoundingBox::ERotatedBoundingBox

Constructor of the rotated bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
void ERotatedBoundingBox(
 float centerX,
 float centerY,
 float width,
 float height,
 float angle
 )

void ERotatedBoundingBox(
 )

void ERotatedBoundingBox(
 const ERotatedBoundingBox& other
 )
```

Parameters

centerX

The abscissa of the center of the bounding box.

centerY

The ordinate of the center of the bounding box.

width

The width of the bounding box.

height

The height of the bounding box.

angle

The angle of the bounding box (in the current angle units).

other

-

ERotatedBoundingBox::GetHeight

Returns the height of the bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
float GetHeight() const
```

ERotatedBoundingBox::LocalToGlobalBox

Transforms a (local) rotated bounding box, as another (global) rotated bounding box whose coordinates are defined relatively to the current rotated bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
ERotatedBoundingBox LocalToGlobalBox(
 const ERotatedBoundingBox& localBox
 )
```

Parameters

localBox

-

ERotatedBoundingBox::LocalToGlobalPoint

Transforms a (local) point, as another (global) point whose coordinates are defined relatively to the current rotated bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint LocalToGlobalPoint(
 const EPoint& localPoint
 )
```

Parameters

localPoint

-

ERotatedBoundingBox::GetMajorAxis

Returns the major axis of the ERotatedBoundingBox as an ELine.

Namespace: Euresys::Open_eVision

[C++]

```
ELine GetMajorAxis() const
```

ERotatedBoundingBox::operator=

Assignment operator.

Namespace: Euresys::Open_eVision

[C++]

```
ERotatedBoundingBox& operator=(
 const ERotatedBoundingBox& other
 )
```

Parameters

other

-

ERotatedBoundingBox::operator==

Checks if this ERotatedBoundingBox instance is strictly equal to another.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const ERotatedBoundingBox& other
 )
```

Parameters

other

Reference to the other ERotatedBoundingBox instance

ERotatedBoundingBox::GetQuadrangle

Returns the coordinates of the four corners of the bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
EQuadrangle GetQuadrangle() const
```

ERotatedBoundingBox::Rotate

Applies a rotation to the bounding box. The bounding box center is the center of rotation.

Namespace: Euresys::Open_eVision

```
[C++]
void Rotate(
 float angle
 )
```

Parameters

angle
The rotation angle.

ERotatedBoundingBox::Scale

Applies a scale to the size of the bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
void Scale(
 float scaleWidth,
 float scaleHeight
 )
```

Parameters

scaleWidth
The scale to apply to the width.

scaleHeight
The scale to apply to the height.

ERotatedBoundingBox::Translate

Applies a translation on the center of the bounding box.

Namespace: Euresys::Open_eVision

```
[C++]
void Translate(
 float offsetX,
 float offsetY
 )
```

Parameters

offsetX

The offset along the X-axis.

offsetY

The offset along the Y-axis.

ERotatedBoundingBox::GetUpperLeftCorner

Returns the coordinates of the upper left corner of the bounding box (the position with the minimum Y and then minimum X).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetUpperLeftCorner() const
```

ERotatedBoundingBox::GetWidth

Returns the width of the bounding box.

Namespace: Euresys::Open_eVision

[C++]

```
float GetWidth() const
```

4.228. ESamplePoint Class

A point sampled by a gauge.

Namespace: Euresys::Open_eVision

Methods

ESamplePoint	Constructor
GetIsOutlier	Outlier status of the sample point
GetIsValid	Validity of the sample point
GetPosition	Position of the sample point
operator=	Copies all the data from another ESamplePoint object into the current ESamplePoint object.

ESamplePoint::ESamplePoint

Constructor

Namespace: Euresys::Open_eVision

```
[C++]
void ESamplePoint(
 )
void ESamplePoint(
 const ESamplePoint& other
 )
```

Parameters

other

-

ESamplePoint::GetIsOutlier

Outlier status of the sample point

Namespace: Euresys::Open_eVision

```
[C++]
bool GetIsOutlier()
```

ESamplePoint::GetIsValid

Validity of the sample point

Namespace: Euresys::Open_eVision

```
[C++]
bool GetIsValid()
```

ESamplePoint::operator=

Copies all the data from another ESamplePoint object into the current ESamplePoint object.

Namespace: Euresys::Open_eVision

[C++]

```
ESamplePoint& operator=(
 const ESamplePoint& other
 )
```

Parameters

other

ESamplePoint object to be copied.

ESamplePoint::GetPosition

Position of the sample point

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPosition()
```

4.229. EScaleCalibrationModel Class

EScaleCalibrationModel is used to convert depth map 2.5D point to 3D world position, only by applying a scale factor.
That kind of "calibration" does not correct the perspective or distortion present in depth maps.
It is a simple and fast way to get a 3D point cloud by applying a scale to each coordinate axis of the depth map pixels.

Base Class:ECalibrationModel

Namespace: Euresys::Open_eVision::Easy3D

Methods

EScaleCalibrationModel	Constructs a EScaleCalibrationModel. By default parameters are initialized to default unit values.
GetFactorX	Returns the width of a pixel in metric unit.
GetFactorY	Returns the distance between 2 profiles (depth map lines) in metric unit.
GetFactorZ	Returns the scale of the pixel value in metric unit.
GetType	Returns the type of calibration model, see ECalibrationType.
Load	Loads the EScaleCalibrationModel calibration model. The given ESerializer must have been created for reading.
operator=	Assignment operator.
operator==	Comparison operator.

Save	Saves the EScaleCalibrationModel calibration model. The given ESerializer must have been created for writing.

EScaleCalibrationModel::EScaleCalibrationModel

Constructs a EScaleCalibrationModel. By default parameters are initialized to default unit values.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EScaleCalibrationModel(
 )
void EScaleCalibrationModel(
 float factorX,
 float factorY,
 float factorZ
 )
void EScaleCalibrationModel(
 const EScaleCalibrationModel& other
 )
```

Parameters

factorX
Width of a pixel in metric unit (factor for X coordinate).

factorY
Distance between 2 profiles (depth map lines) in metric unit (factor for Y coordinate).

factorZ
Scale of the pixel value in metric unit (factor for Z coordinate).

other
Another EScaleCalibrationModel used for the initialization.

EScaleCalibrationModel::GetFactorX

Returns the width of a pixel in metric unit.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetFactorX() const
```

EScaleCalibrationModel::GetFactorY

Returns the distance between 2 profiles (depth map lines) in metric unit.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetFactorY() const
```

EScaleCalibrationModel::GetFactorZ

Returns the scale of the pixel value in metric unit.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetFactorZ() const
```

EScaleCalibrationModel::Load

Loads the EScaleCalibrationModel calibration model. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EScaleCalibrationModel::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EScaleCalibrationModel& operator=(
 const EScaleCalibrationModel& other
 )
```

Parameters

other

Another EScaleCalibrationModel.

EScaleCalibrationModel::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const EScaleCalibrationModel& other
 )
```

Parameters

other

Another EScaleCalibrationModel.

EScaleCalibrationModel::Save

Saves the EScaleCalibrationModel calibration model. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EScaleCalibrationModel::GetType

Returns the type of calibration model, see ECalibrationType.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::ECalibrationType GetType() const
```

4.230. ESearchParamsType Class

This class is deprecated.

This class is instantiated once in each EMatrixCodeReader and represents the search parameters that are explored when reading a EMatrixCode.

Remarks

This class contains 4 sets of search parameters that are scanned at read time. At EMatrixCodeReader construction time, these sets of values are initialized with all possible values. This means, for instance, that a data matrix code read in a freshly created reader is matched against all possible logical sizes. As a consequence, the default values of these sets are not repeated here. They are assumed to represent every possible search parameters. The ESearchParamsType object needs to be used only if you wish to "override" the learning process.

Namespace: Euresys::Open_eVision

Methods

AddContrast	Adds a new contrast mode to the list of EMatrixCodeContrastMode the candidate is matched against at read time.
AddFamily	Adds a new family to the list of EFamily the candidate is matched against at read time.
AddFlipping	Adds a new flipping to the list of EFlipping the candidate is matched against at read time.
AddLogicalSize	Adds a new logical size to the list of ELogicalSize the candidate is matched against at read time.
ClearContrast	Clears the list of contrast modes the candidate is matched against at read time.
ClearFamily	Clears the list of families the candidate is matched against at read time.
ClearFlipping	Clears the list of flippings the candidate is matched against at read time.
ClearLogicalSize	Clears the list of logical sizes the candidate is matched against at read time.
GetContrast	Gets an item in the list of EMatrixCodeContrastMode the candidate is matched against at read time.
GetContrastCount	The size of the list of EMatrixCodeContrastMode the candidate is matched against at read time.
GetFamily	Gets an item in the list of EFamily the candidate is matched against at read time.

GetFamilyCount	The size of the list of EFamily the candidate is matched against at read time.
GetFlipping	Gets an item in the list of EFlipping the candidate is matched against at read time.
GetFlippingCount	The size of the list of EFlipping the candidate is matched against at read time.
GetLogicalSize	Gets an item in the list of ELogicalSize the candidate is matched against at read time.
GetLogicalSizeCount	The size of the list of ELogicalSize the candidate is matched against at read time.
RemoveContrast	Removes the contrast mode from the list of EMatrixCodeContrastMode the candidate is matched against at read time.
RemoveFamily	Removes the family from the list of EFamily the candidate is matched against at read time.
RemoveFlipping	Removes the flipping from the list of EFlipping the candidate is matched against at read time.
RemoveLogicalSize	Removes the logical size from the list of ELogicalSize the candidate is matched against at read time.

ESearchParamsType::AddContrast

This method is deprecated.

Adds a new contrast mode to the list of EMatrixCodeContrastMode the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
void AddContrast(
 Euresys::Open_eVision::EMatrixCodeContrastMode searchContrast
 )
```

Parameters

searchContrast

Contrast mode to add to the list.

ESearchParamsType::AddFamily

This method is deprecated.

Adds a new family to the list of EFamily the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void AddFamily(
 Euresys::Open_eVision::EFamily searchFamily
 )
```

Parameters

searchFamily

Family to add to the list.

ESearchParamsType::AddFlipping

This method is deprecated.

Adds a new flipping to the list of EFlipping the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void AddFlipping(
 Euresys::Open_eVision::EFlipping searchFlipping
 )
```

Parameters

searchFlipping

Flipping to add to the list.

ESearchParamsType::AddLogicalSize

This method is deprecated.

Adds a new logical size to the list of ELogicalSize the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void AddLogicalSize(
 Euresys::Open_eVision::ELogicalSize searchLogicalSize
 )
```

Parameters

searchLogicalSize

Logical size to add to the list.

ESearchParamsType::ClearContrast

This method is deprecated.

Clears the list of contrast modes the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearContrast(
 )
```

ESearchParamsType::ClearFamily

This method is deprecated.

Clears the list of families the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearFamily(
 )
```

ESearchParamsType::ClearFlipping

This method is deprecated.

Clears the list of flippings the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearFlipping(
 )
```

ESearchParamsType::ClearLogicalSize

This method is deprecated.

Clears the list of logical sizes the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
void ClearLogicalSize(
 )
```

ESearchParamsType::GetContrastCount

This property is deprecated.

The size of the list of EMatrixCodeContrastMode the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
int GetContrastCount() const
```

ESearchParamsType::GetFamilyCount

This property is deprecated.

The size of the list of EFamily the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFamilyCount() const
```

ESearchParamsType::GetFlippingCount

This property is deprecated.

The size of the list of EFlipping the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
int GetFlippingCount() const
```

ESearchParamsType::GetContrast

This method is deprecated.

Gets an item in the list of EMatrixCodeContrastMode the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EMatrixCodeContrastMode GetContrast(
 int index
 )
```

Parameters

index

Position in the list.

ESearchParamsType::GetFamily

This method is deprecated.

Gets an item in the list of EFamily the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EFamily GetFamily(
 int index
 )
```

Parameters

index

Position in the list.

ESearchParamsType::GetFlipping

This method is deprecated.

Gets an item in the list of EFlipping the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EFlipping GetFlipping(
 int index
 )
```

Parameters

index

Position in the list.

ESearchParamsType::GetLogicalSize

This method is deprecated.

Gets an item in the list of ELogicalSize the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ELogicalSize GetLogicalSize(
 int index
 )
```

Parameters

index

Position in the list.

ESearchParamsType::GetLogicalSizeCount

This property is deprecated.

The size of the list of ELogicalSize the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
int GetLogicalSizeCount() const
```

ESearchParamsType::RemoveContrast

This method is deprecated.

Removes the contrast mode from the list of EMatrixCodeContrastMode the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveContrast(
 Euresys::Open_eVision::EMatrixCodeContrastMode searchContrast
 )
```

Parameters

searchContrast

Contrast mode to remove from the list.

ESearchParamsType::RemoveFamily

This method is deprecated.

Removes the family from the list of EFamily the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveFamily(
 Euresys::Open_eVision::EFamily searchFamily
 )
```

Parameters

searchFamily

Family to remove from the list.

ESearchParamsType::RemoveFlipping

This method is deprecated.

Removes the flipping from the list of EFlipping the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveFlipping(
 Euresys::Open_eVision::EFlipping searchFlipping
 )
```

Parameters

searchFlipping

Flipping to remove from the list.

ESearchParamsType::RemoveLogicalSize

This method is deprecated.

Removes the logical size from the list of ELogicalSize the candidate is matched against at read time.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveLogicalSize(
 Euresys::Open_eVision::ELogicalSize searchLogicalSize
 )
```

Parameters

searchLogicalSize

Logical size to remove from the list.

4.231. ESerializer Class

Abstract interface for file-like objects.

Remarks

The ESerializer object manages operations of reading from and writing to an archive (a file on the system hard disk, for instance). ESerializer objects cannot be instantiated directly. To create an ESerializer object, one of the following static factory methods has to be used:

Note. An ESerializer object can not be used in the same time for reading and writing. So, ESerializer::CreateFileWriter creates an ESerializer object that should be used with Save methods and ESerializer::CreateFileReader creates an ESerializer object that should be used with Load methods.

Derived Class(es):EFilePointerSerializerEFileSerializerEMemorySerializer

Namespace: Euresys::Open_eVision

Methods

Close	Closes the file associated with the ESerializer object.
CreateFileReader	Returns an ESerializer object suitable for reading from a file.
CreateFileWriter	Returns an ESerializer object suitable for opening a file and writing into it.
CreateMemoryReader	Returns an ESerializer object suitable for reading from a buffer.
CreateMemoryWriter	Returns an ESerializer object suitable for serializing into a buffer.
GetWriting	Returns true if the ESerializer object has been created for writing and false otherwise.

ESerializer::Close

Closes the file associated with the ESerializer object.

Namespace: Euresys::Open_eVision

```
[C++]
void Close(
 )
```

ESerializer::CreateFileReader

Returns an ESerializer object suitable for reading from a file.

Namespace: Euresys::Open_eVision

[C++]

```
std::unique_ptr<ESerializer> CreateFileReader(
 const std::string& filePath
 )
```

Parameters

filePath

Full path and name specification of the file to be used to create the ESerializer object.

Remarks

It is up to users to delete the ESerializer object when they have done using it in Load calls. If the call does not succeed, it returns NULL. Please check the Open eVision error code to get further informations.

ESerializer::CreateFileWriter

Returns an ESerializer object suitable for opening a file and writing into it.

Namespace: Euresys::Open_eVision

[C++]

```
std::unique_ptr<ESerializer> CreateFileWriter(
 const std::string& filePath,
 Euresys::Open_eVision::ESerializerFileWriterMode mode
 )
```

Parameters

filePath

Full path and name specification of the file to be used to create the ESerializer object.

mode

Creation mode of the storage file, as defined by ESerializerFileWriterMode (by default, Create).

Remarks

The ESerializerFileWriterMode parameter is an enumerated type that allows to control what happens when the file already exists: * If mode is Create, the call will not succeed if the file already exists. * If mode is Overwrite, the existing file will be overwritten. * If mode is Append, the new data will be appended to the existing file content. It is up to users to delete the ESerializer object when they have done using it in Save calls. If the call does not succeed, it returns NULL. Please check the Open eVision error code to get further information.

ESerializer::CreateMemoryReader

Returns an ESerializer object suitable for reading from a buffer.

Namespace: Euresys::Open_eVision

```
[C++]
std::unique_ptr<ESerializer> CreateMemoryReader(
 void* buffer,
 OEV_UINT32 size
 )
```

Parameters

buffer

Address of the buffer to be used by the memory serializer.

size

Size of the buffer to be used by the memory serializer.

Remarks

The buffer is copied inside the internal memory of the serializer. It is up to users to delete the ESerializer object when they have done using it in Load calls.

ESerializer::CreateMemoryWriter

Returns an ESerializer object suitable for serializing into a buffer.

Namespace: Euresys::Open_eVision

```
[C++]
std::unique_ptr<ESerializer> CreateMemoryWriter(
 )
std::unique_ptr<ESerializer> CreateMemoryWriter(
 OEV_UINT32 initialBufferSize
 )
std::unique_ptr<ESerializer> CreateMemoryWriter(
 void* buffer,
 OEV_UINT32 size
 )
```

Parameters

initialBufferSize

-

buffer

Address of the buffer to be used by the memory serializer.

size

Size of the buffer to be used by the memory serializer.

Remarks

If a buffer is not provided, a buffer will be automatically created. Please note that, in this case, if you didn't provide a size, the buffer will be automatically resized as needed. It is up to users to delete the ESerializer object when they have done using it in Save calls. The returned serializer underlying type is EMemorySerializer, for more information, see the class documentation.

ESerializer::GetWriting

Returns true if the ESerializer object has been created for writing and false otherwise.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetWriting() const
```

4.232. EShape Class

Abstract class to federate the classes that can be hierarchically attached together (from a geometrical point of view).

Derived Class (es):
ECircleShape
EFrameShape
ELineShapeEPointShapeEPolygonShapeERectangleShapeEWedgeShapeEWorldShape

Namespace: Euresys::Open_eVision

Methods

Attach	Attaches the gauge to a mother gauge or shape.
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
Detach	Detaches the gauge from its mother gauge or shape.
DetachDaughters	Detaches the daughter gauges or shapes.
DisableBehaviorFilter	Disables (i.e. removes) a condition from the list of conditions in the behavior filter.
DisableTypeFilter	Enables all shape types
Drag	Moves a handle to a new position and updates the position parameters of the shape.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
EnableBehaviorFilter	Modifies the so-called **behavior filter** that is a conjunction of conditions specifying when a shape or a gauge should be displayed or taken into consideration when monitoring the mouse interactions.
EnableTypeFilter	Enables the filter of the specified shape type
GetActive	Flag indicating whether the shape is active or not.

GetActualShape	Flag indicating whether an inquiry returns a result pertaining to the nominal gauge (false, default) or the fitted model (true).
GetAllocated	Gets the allocated flag.
GetClosestShape	Closest shape among the daughters.
GetDaughter	Returns a pointer to the specified daughter gauge or shape.
GetDragable	Flag indicating whether the shape can be dragged or not.
GetDraggingMode	Gets the dragging mode which defines how the shape could be dragged.
GetFound	Flag indicating whether a measured shape has been found.
GetHitHandle	Handle currently under the cursor.
GetHitShape	Pointer to the shape currently under the cursor.
GetLabeled	Flag indicating whether the shape label should be displayed or not.
GetMother	Pointer to the mother shape.
GetName	Name of the EShape object.
GetNumDaughters	Number of daughters attached to the shape.
GetPanX	Current horizontal panning value for drawing operations, expressed in pixels. By default, no panning occurs.
GetPanY	Current vertical panning value for drawing operations, expressed in pixels. By default, no panning occurs.
GetProperty	Gets the value of property aProperty. Throws an exception if no such property exists.
GetResizable	Flag indicating whether the shape can be resized or not.
GetRotatable	Flag indicating whether the shape can be rotated or not.
GetSelectable	Flag indicating whether the shape can be selected or not.
GetSelected	Flag indicating whether the shape is selected or not.
GetShapeNamed	Returns a pointer to a daughter gauge or shape specified by its name.
GetType	Shape type.
GetVisible	Flag indicating whether the shape is visible or not.
GetWorldShape	World ancestor.
GetZoomX	Current horizontal zooming factor for drawing operations.
GetZoomY	Current vertical zooming factor for drawing operations.
HasProperty	Returns whether a EShape has a property named aProperty.
HasValidPolarityProperty	Returns whether the EShape has a valid polarity property. I.e. if it has a property named "polarity" whose value is either "inverted" or "direct".
HitTest	Checks if there is a handle under the cursor.
InvalidateWorld	Invalidates the world shape for this shape and all its ancestors

Load	Loads a Shape.The given ESerializer must have been created for reading.
LocalToSensor	Transforms a point from local coordinates to sensor coordinates.
RemoveProperty	Removes the property name aProperty from the EShape.
Save	Loads a Shape.The given ESerializer must have been created for writing.
SensorToLocal	Transforms a point from sensor coordinates to local coordinates.
SetActive	Flag indicating whether the shape is active or not.
SetActiveRecursive	Sets the flag indicating whether the shape and all its daughters are active or not.
SetActualShape	Flag indicating whether an inquiry returns a result pertaining to the nominal gauge (false, default) or the fitted model (true).
SetActualShapeRecursive	Sets the flag indicating whether an inquiry returns a result pertaining to the nominal gauge (false, default) or the fitted model (true). The flag is set in this shape and all its daughters.
SetAllocated	Sets the allocated flag.
SetCursor	Sets the cursor current coordinates.
SetDragable	Flag indicating whether the shape can be dragged or not.
SetDragableRecursive	Sets the flag indicating whether the shape and all its daughters can be dragged or not.
SetDraggingMode	Sets the dragging mode which defines how the shape could be dragged.
SetLabeled	Flag indicating whether the shape label should be displayed or not.
SetLabeledRecursive	Sets the flag indicating whether the shape label should be displayed or not. The flag is set in this shape and all its daughters.
SetName	Name of the EShape object.
SetPan	Sets the horizontal and vertical panning values for drawing operations, expressed in pixels.
SetProperty	Sets the value of property aProperty to aValue for this EShape.
SetPropertyRecursive	Sets the value of property aProperty to aValue for this EShape and all of its daughters.
SetResizable	Flag indicating whether the shape can be resized or not.
SetResizableRecursive	Sets the flag indicating whether the shape and all its daughters can be resized or not.
SetRotatable	Flag indicating whether the shape can be rotated or not.
SetRotatableRecursive	Sets the flag indicating whether the shape and all its daughters can be rotated or not.
SetSelectable	Flag indicating whether the shape can be selected or not.
SetSelectableRecursive	Sets the flag indicating whether the shape and all its daughters can be selected or not.

SetSelected	Flag indicating whether the shape is selected or not.
SetSelectedRecursive	Sets the flag indicating whether the shape and all its daughters are selected or not.
SetVisible	Flag indicating whether the shape is visible or not.
SetVisibleRecursive	Sets the flag indicating whether the shape and its daughter shapes are visible or not.
SetZoom	Sets the horizontal and vertical zooming factors for drawing operations.

EShape::GetActive

EShape::SetActive

Flag indicating whether the shape is active or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetActive()
void SetActive(bool active)
```

EShape::SetActiveRecursive

Sets the flag indicating whether the shape and all its daughters are active or not.

Namespace: Euresys::Open_eVision

```
[C++]
void SetActiveRecursive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying EShape::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EShape::GetActualShape

EShape::SetActualShape

Flag indicating whether an inquiry returns a result pertaining to the nominal gauge (false, default) or the fitted model (true).

Namespace: Euresys::Open_eVision

[C++]

```
bool GetActualShape()
void SetActualShape(bool actualShape)
```

EShape::SetActualShapeRecursive

Sets the flag indicating whether an inquiry returns a result pertaining to the nominal gauge (false, default) or the fitted model (true). The flag is set in this shape and all its daughters.

Namespace: Euresys::Open_eVision

[C++]

```
void SetActualShapeRecursive(bool actualShape)
```

EShape::Attach

Attaches the gauge to a mother gauge or shape.

Namespace: Euresys::Open_eVision

[C++]

```
void Attach(
 EShape* mother
 )
```

Parameters

mother

Pointer to the mother gauge or shape.

Remarks

When attached to a mother gauge, be aware that daughter gauges are not positioned according to the nominal mother gauge position, but to its corresponding fitted model.

EShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

EShape::GetClosestShape

Closest shape among the daughters.

Namespace: Euresys::Open_eVision

```
[C++]
EShape* GetClosestShape()
```

Remarks

Use EShape::Closest to recompute the closest shape.

EShape::Detach

Detaches the gauge from its mother gauge or shape.

Namespace: Euresys::Open_eVision

```
[C++]
void Detach(
 )
```

EShape::DetachDaughters

Detaches the daughter gauges or shapes.

Namespace: Euresys::Open_eVision

```
[C++]
void DetachDaughters(
 )
```

EShape::DisableBehaviorFilter

Disables (i.e. removes) a condition from the list of conditions in the behavior filter.

Namespace: Euresys::Open_eVision

```
[C++]
void DisableBehaviorFilter(
 Euresys::Open_eVision::EShapeBehavior behavior
 )
```

Parameters

behavior

The behavior of the shape to be removed from the behavior filter.

Remarks

The condition to be disabled is identified by the behavior about which the condition is. Disabling a behavior leads to less restrictive conditions for the Draw and HitTest methods to be actually carried on.

EShape::DisableTypeFilter

Enables all shape types

Namespace: Euresys::Open_eVision

[C++]

```
void DisableTypeFilter(
 )
```

EShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

EShape::GetDragable

EShape::SetDragable

Flag indicating whether the shape can be dragged or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetDragable()
void SetDragable(bool dragable)
```

EShape::SetDragableRecursive

Sets the flag indicating whether the shape and all its daughters can be dragged or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetDragableRecursive(bool dragable)
```

EShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EShape::EnableBehaviorFilter

Modifies the so-called **behavior filter** that is a conjunction of conditions specifying when a shape or a gauge should be displayed or taken into consideration when monitoring the mouse interactions.

Namespace: Euresys::Open_eVision

[C++]

```
void EnableBehaviorFilter(
 Euresys::Open_eVision::EShapeBehavior behavior,
 bool value
 )
```

Parameters

behavior

The behavior of the shape to be tested.

value

The value at which the behavior property should be set to pass the test. By default, equals True.

Remarks

This method registers a new necessary condition for the Draw and HitTest families of methods to be actually carried on. Such a condition is about the behavior of the shape, as specified by the behavior argument. Initially, the behavior filter contains an empty list of conditions, which means that the Draw and HitTest methods will always be executed. Adding a new condition through EShape::EnableBehaviorFilter will introduce a new restriction on the effective execution of these methods. Use EShape::DisableBehaviorFilter to remove a condition from the behavior filter.

EShape::EnableTypeFilter

Enables the filter of the specified shape type

Namespace: Euresys::Open_eVision

[C++]

```
void EnableTypeFilter(
 OEV_UINT32 un32Types
 )
```

Parameters

un32Types

The type of the shape to filter.

EShape::GetAllocated

Gets the allocated flag.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetAllocated(
 )
```

EShape::GetDaughter

Returns a pointer to the specified daughter gauge or shape.

Namespace: Euresys::Open_eVision

```
[C++]
EShape* GetDaughter(
 OEV_UINT32 index
 )
```

Parameters

index

Daughter gauge or shape index.

EShape::GetDraggingMode

Gets the dragging mode which defines how the shape could be dragged.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EDraggingMode GetDraggingMode(
 )
```

EShape::GetFound

Flag indicating whether a measured shape has been found.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetFound(
 )
```

Remarks

After calling Process() or Measure() on a gauge object, use GetFound() to get the status of the measurement.

EShape::GetProperty

Gets the value of property aProperty. Throws an exception if no such property exists.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetProperty(
 const std::string& aProperty
 )
```

Parameters

aProperty

The name of the property.

EShape::GetShapeNamed

Returns a pointer to a daughter gauge or shape specified by its name.

Namespace: Euresys::Open_eVision

[C++]

```
EShape* GetShapeNamed(
 const std::string& name
 )
```

Parameters

name

Name of the daughter gauge or shape.

EShape::HasProperty

Returns whether a EShape has a property named aProperty.

Namespace: Euresys::Open_eVision

[C++]

```
bool HasProperty(
 const std::string& aProperty
 )
```

Parameters

aProperty

The name of the property.

EShape::HasValidPolarityProperty

Returns whether the EShape has a valid polarity property.
I.e. if it has a property named "polarity" whose value is either "inverted" or "direct".

Namespace: Euresys::Open_eVision

[C++]

```
bool HasValidPolarityProperty(
 )
```

Remarks

The "polarity" property is mostly used to define the orientation of the gradient in a shape of a EVectorModel.

EShape::GetHitHandle

Handle currently under the cursor.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EDragHandle GetHitHandle()
```

Remarks

When the cursor is over a particular handle, its shape could be changed for feedback.

EShape::GetHitShape

Pointer to the shape currently under the cursor.

Namespace: Euresys::Open_eVision

[C++]

```
EShape* GetHitShape()
```

Remarks

When the cursor is over a particular shape, its shape could be changed for feedback.

EShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

EShape::InvalidateWorld

Invalidates the world shape for this shape and all its ancestors

Namespace: Euresys::Open_eVision

```
[C++]
void InvalidateWorld(
 )
```

EShape::GetLabeled

EShape::SetLabeled

Flag indicating whether the shape label should be displayed or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetLabeled()

void SetLabeled(bool labeled)
```

EShape::SetLabeledRecursive

Sets the flag indicating whether the shape label should be displayed or not. The flag is set in this shape and all its daughters.

Namespace: Euresys::Open_eVision

[C++]

```
void SetLabeledRecursive(bool labeled)
```

EShape::Load

Loads a Shape.The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path,
 bool daughters
 )

void Load(
 ESerializer* serializer,
 bool daughters
 )
```

Parameters

path
The file path.

daughters
Indicates if the load must be done on the whole hierarchy or just this object.

serializer
Pointer to the ESerializer created for reading.

EShape::LocalToSensor

Transforms a point from local coordinates to sensor coordinates.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint LocalToSensor(
 const EPoint& LPoint
 )
```

Parameters

LPoint
The point in local coordinates.

EShape::GetMother

Pointer to the mother shape.

Namespace: Euresys::Open_eVision

[C++]

```
EShape* GetMother()
```

EShape::GetName

EShape::SetName

Name of the EShape object.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetName()
void SetName(const std::string& name)
```

EShape::GetNumDaughters

Number of daughters attached to the shape.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumDaughters()
```

EShape::GetPanX

Current horizontal panning value for drawing operations, expressed in pixels. By default, no panning occurs.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPanX()
```

EShape::GetPanY

Current vertical panning value for drawing operations, expressed in pixels. By default, no panning occurs.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPanY()
```

EShape::RemoveProperty

Removes the property name aProperty from the EShape.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveProperty(
 const std::string& aProperty
 )
```

Parameters

aProperty

The name of the property.

EShape::GetResizable

EShape::SetResizable

Flag indicating whether the shape can be resized or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetResizable()

void SetResizable(bool resizable)
```

EShape::SetResizableRecursive

Sets the flag indicating whether the shape and all its daughters can be resized or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetResizableRecursive(bool resizable)
```

EShape::GetRotatable

EShape::SetRotatable

Flag indicating whether the shape can be rotated or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRotatable()
```

```
void SetRotatable(bool rotatable)
```

EShape::SetRotatableRecursive

Sets the flag indicating whether the shape and all its daughters can be rotated or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetRotatableRecursive(bool rotatable)
```

EShape::Save

Loads a Shape.The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path,
 bool daughters
 )
```

```
void Save(
 ESerializer* serializer,
 bool daughters
 )
```

Parameters

path
The file path.

daughters
Indicates if the save must be done on the whole hierarchy or just this object.

serializer
Pointer to the ESerializer created for writing.

EShape::GetSelectable

EShape::SetSelectable

Flag indicating whether the shape can be selected or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSelectable()
void SetSelectable(bool selectable)
```

EShape::SetSelectableRecursive

Sets the flag indicating whether the shape and all its daughters can be selected or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSelectableRecursive(bool selectable)
```

EShape::GetSelected

EShape::SetSelected

Flag indicating whether the shape is selected or not.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSelected()
void SetSelected(bool selected)
```

EShape::SetSelectedRecursive

Sets the flag indicating whether the shape and all its daughters are selected or not.

Namespace: Euresys::Open_eVision

```
[C++]
void SetSelectedRecursive(bool selected)
```

EShape::SensorToLocal

Transforms a point from sensor coordinates to local coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint SensorToLocal(
 const EPoint& SPoint
 )
```

Parameters

SPoint

The point in sensor coordinates.

EShape::SetAllocated

Sets the allocated flag.

Namespace: Euresys::Open_eVision

```
[C++]
void SetAllocated(
 bool bAllocated,
 bool bDaughters
 )
```

Parameters

bAllocated

Whether to set the allocated flag or not.

bDaughters

Indicates if the set must be done in the whole hierarchy or just this object.

EShape::SetCursor

Sets the cursor current coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void SetCursor(
 int x,
 int y
 )
```

Parameters

x

Cursor current coordinates.

y

Cursor current coordinates.

EShape::SetDraggingMode

Sets the dragging mode which defines how the shape could be dragged.

Namespace: Euresys::Open_eVision

```
[C++]
void SetDraggingMode(
 Euresys::Open_eVision::EDraggingMode eDraggingMode,
 bool bDaughters
 )
```

Parameters

eDraggingMode

The draggingMode.

bDaughters

Indicates if the set must be done in the whole hierarchy or just this object.

EShape::SetPan

Sets the horizontal and vertical panning values for drawing operations, expressed in pixels.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPan(
 float panX,
 float panY
 )
```

Parameters

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

All objects attached to an EWorldShape object inherit of the same panning factor.

EShape::SetProperty

Sets the value of property aProperty to aValue for this EShape.

Namespace: Euresys::Open_eVision

[C++]

```
void SetProperty(
 const std::string& aProperty,
 const std::string& aValue
 )
```

Parameters

aProperty

The name of the property.

aValue

The value of the property.

Remarks

You may use a shape property to define the orientation of the gradient in a shape of a EVectorModel. This is done by setting a property named "polarity" to "inverted" or "direct". Additionally, you may set any property to any value for your own needs.

EShape::SetPropertyRecursive

Sets the value of property aProperty to aValue for this EShape and all of its daughters.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPropertyRecursive(
 const std::string& aProperty,
 const std::string& aValue
 )
```

Parameters

aProperty

The name of the property.

aValue

The value of the property.

EShape::SetZoom

Sets the horizontal and vertical zooming factors for drawing operations.

Namespace: Euresys::Open_eVision

[C++]

```
void SetZoom(
 float zoomX,
 float zoomY
 )
```

Parameters

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

Remarks

All objects attached to an EWorldShape inherit of the same zooming factor.

EShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

EShape::GetVisible

EShape::SetVisible

Flag indicating whether the shape is visible or not.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetVisible()
void SetVisible(bool visible)
```

EShape::SetVisibleRecursive

Sets the flag indicating whether the shape and its daughter shapes are visible or not.

Namespace: Euresys::Open_eVision

```
[C++]
void SetVisibleRecursive(bool visible)
```

EShape::GetWorldShape

World ancestor.

Namespace: Euresys::Open_eVision

```
[C++]
EWorldShape* GetWorldShape()
```

EShape::GetZoomX

Current horizontal zooming factor for drawing operations.

Namespace: Euresys::Open_eVision

```
[C++]
float GetZoomX()
```

EShape::GetZoomY

Current vertical zooming factor for drawing operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetZoomY()
```

4.233. ESimpleCropper Class

Manages a point cloud cropper based on X/Y/Z value ranges.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Crop	Crops a point cloud.
ESimpleCropper	Creates an ESimpleCropper object.
operator=	Assignment operator
operator==	Equality operator.
SetXRange	Allowed value range along the X axis. Pass NULL if no cropping should be done along this axis.
SetYRange	Allowed value range along the Y axis. Pass NULL if no cropping should be done along this axis.
SetZRange	Allowed value range along the Z axis. Pass NULL if no cropping should be done along this axis.

ESimpleCropper::Crop

Crops a point cloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Crop(
 EPointCloud& cloudIn,
 EPointCloud& cloudOut
 )
```

Parameters

cloudIn

Cloud to be cropped.

cloudOut

Cropped cloud.

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

ESimpleCropper::ESimpleCropper

Creates an ESimpleCropper object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ESimpleCropper(
 )

void ESimpleCropper(
 const EFloatRange* rangeX,
 const EFloatRange* rangeY,
 const EFloatRange* rangeZ
 )

void ESimpleCropper(
 const ESimpleCropper& other
 )
```

Parameters

rangeX

Allowed value range along the X axis. Pass NULL if no cropping should be done along this axis.

rangeY

Allowed value range along the Y axis. Pass NULL if no cropping should be done along this axis.

rangeZ

Allowed value range along the Z axis. Pass NULL if no cropping should be done along this axis.

other

An other ESimpleCropper

ESimpleCropper::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ESimpleCropper& operator=(
 const ESimpleCropper& other
 )
```

Parameters

other

An other ESimpleCropper

ESimpleCropper::operator==

Equality operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const ESimpleCropper& other
 )
```

Parameters

other

An other ESimpleCropper.

ESimpleCropper::SetXRange

Allowed value range along the X axis. Pass NULL if no cropping should be done along this axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetXRange(const EFloatRange* rangeX)
```

ESimpleCropper::SetYRange

Allowed value range along the Y axis. Pass NULL if no cropping should be done along this axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetYRange(const EFloatRange* rangeY)
```

ESimpleCropper::SetZRange

Allowed value range along the Z axis. Pass NULL if no cropping should be done along this axis.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetZRange(const EFloatRange* rangeZ)
```

4.234. ESphereFitter Class

A ESphereFitter object is used to fit an E3DSphere on an EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

Methods

ESphereFitter	Constructor of an ESphereFitter object.
Fit	Fits an E3DSphere on a given EPointCloud.
Load	Loads the ESphereFitter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator.
Save	Saves the ESphereFitter configuration. The given ESerializer must have been created for writing.

ESphereFitter::ESphereFitter

Constructor of an ESphereFitter object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ESphereFitter(
 )

void ESphereFitter(
 const ESphereFitter& other
 )
```

Parameters

other

Reference to the ESphereFitter used for the initialization.

ESphereFitter::Fit

Fits an E3DSphere on a given EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DSphere Fit(
 const EPointCloud& pc
 )

E3DSphere Fit(
 const EPointCloud& pc,
 float& averageDistance
 )
```

Parameters

pc

The reference to the point cloud.

averageDistance

The reference to a float which will store the average distance from this sphere to the points that were used for the fit.

ESphereFitter::Load

Loads the ESphereFitter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

ESphereFitter::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ESphereFitter& operator=(
 const ESphereFitter& other
 )
```

Parameters

other

The ESphereFitter object that should be copied.

ESphereFitter::Save

Saves the ESphereFitter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

4.235. ESphericalCropper Class

Manages a spherical point cloud cropper.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Crop	Crops a point cloud.
ESphericalCropper	Creates an ESphericalCropper object.
operator=	Assignment operator.
operator==	Equality operator.

ESphericalCropper::Crop

Crops a point cloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Crop(
 EPointCloud& cloudIn,
 EPointCloud& cloudOut,
 bool invertCrop
 )
```

Parameters

cloudIn

Cloud to be cropped.

cloudOut

Cropped cloud.

invertCrop

Indicates if the points kept must be the points inside (true) or outside (false) the sphere.

Remarks

The input point cloud 'cloudIn' should be different from the output point cloud 'cloudOut'. If not a exception will be thrown.

ESphericalCropper::ESphericalCropper

Creates an ESphericalCropper object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ESphericalCropper(
 const E3DPoint& center,
 float radius
 )

void ESphericalCropper(
 const ESphericalCropper& other
 )
```

Parameters

center

Center of the sphere.

radius

Radius of the sphere.

other

An other ESphericalCropper.

ESphericalCropper::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
ESphericalCropper& operator=(
 const ESphericalCropper& other
 )
```

Parameters

other

An other ESphericalCropper.

ESphericalCropper::operator==

Equality operator.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool operator==(
 const ESphericalCropper& other
 )
```

Parameters

other

An other ESphericalCropper.

4.236. ESpot Class

The ESpot store information about the detections produced by ESpotDetector. A spot is defined by a type (particle or scratch), a polarity (black or white) and a geometry. The geometry is an arbitrary rectangle, represented by a ERotatedBoundingBox.

Namespace: Euresys::Open_eVision

Methods

Draw	Draw the ESpot.
ESpot	Constructs an ESpot with default (invalid) values
GetArea	Get the area of the spot (in number of pixels).
GetBox	Returns the rotated rectangle representing the spot.

GetCenter	Get the center point of the spot.
GetDLLabel	Returns the Deep Learning classification label. This information is available when the Deep Learning classification has been activated by ESpotDetector. See also ESpot.
GetDLProbability	Returns the Deep Learning classification probability. This information is available when the Deep Learning classification has been activated by ESpotDetector. See also ESpot.
GetHeight	Get the height of the bounding box of spot.
GetPolarity	Get the polarity of the spot, i.e. whether it is ESpotPolarity_White, ESpotPolarity_Black or ESpotPolarity_Any. A Spot's polarity is the same as the polarity setting of the detector when the spot was detected.
GetRegion	Returns the ERegion of the spot (the pixels composing the spot)
GetROI	Returns the EROIBW8 of the spot (axis aligned bounding box). The returned ROI is not attached to an image.
GetStrength	Get the strength of the spot (the average grey scale value over the local image background).
GetType	Get the type of the spot, i.e. whether it is a ESpotType_Scratch or a ESpotType_Particle.
GetWidth	Get the width of the bounding box of spot.
Load	Loads the ESpot. The given ESerializer must have been created for reading.
operator!=	Check if two ESpot are different.
operator=	Assignment operator
operator==	Check if two ESpot are identical.
Save	Saves the ESpot. The given ESerializer must have been created for writing.

ESpot::GetArea

Get the area of the spot (in number of pixels).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetArea() const
```

ESpot::GetBox

Returns the rotated rectangle representing the spot.

Namespace: Euresys::Open_eVision

```
[C++]
const ERotatedBoundingBox& GetBox() const
```

ESpot::GetCenter

Get the center point of the spot.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetCenter() const
```

ESpot::GetDLLabel

Returns the Deep Learning classification label. This information is available when the Deep Learning classification has been activated by ESpotDetector.
See also ESpot.

Namespace: Euresys::Open_eVision

```
[C++]
std::string GetDLLabel() const
```

ESpot::GetDLProbability

Returns the Deep Learning classification probability. This information is available when the Deep Learning classification has been activated by ESpotDetector.
See also ESpot.

Namespace: Euresys::Open_eVision

```
[C++]
float GetDLProbability() const
```

ESpot::Draw

Draw the ESpot.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* drawAdapter,
 float extension,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 float extension,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

extension

Extension in pixels of the draw rectangle around the spot. Only used for ESpotType_Particle.

color

Color of the draw rectangle.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

graphicContext

-

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ESpot::ESpot

Constructs an ESpot with default (invalid) values

Namespace: Euresys::Open_eVision

```
[C++]
void ESpot(
 )
void ESpot(
 const ESpot& other
 )
```

Parameters

other

-

ESpot::GetHeight

Get the height of the bounding box of spot.

Namespace: Euresys::Open_eVision

```
[C++]
float GetHeight() const
```

ESpot::Load

Loads the ESpot. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

ESpot::operator!=

Check if two ESpot are different.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const ESpot& other
 )
```

Parameters

other

The other object.

ESpot::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
ESpot& operator=(
 const ESpot& other
 )
```

Parameters

other

-

ESpot::operator==

Check if two ESpot are identical.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ESpot& other
 )
```

Parameters

other

The other object.

ESpot::GetPolarity

Get the polarity of the spot, i.e. whether it is White, Black or Any. A Spot's polarity is the same as the polarity setting of the detector when the spot was detected.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ESpotPolarity GetPolarity() const
```

ESpot::GetRegion

Returns the ERegion of the spot (the pixels composing the spot)

Namespace: Euresys::Open_eVision

[C++]

```
const ERegion& GetRegion() const
```

ESpot::GetROI

Returns the EROIBW8 of the spot (axis aligned bounding box). The returned ROI is not attached to an image.

Namespace: Euresys::Open_eVision

[C++]

```
EROIBW8 GetROI() const
```

ESpot::Save

Saves the ESpot. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The ESerializer object that is written to.

ESpot::GetStrength

Get the strength of the spot (the average grey scale value over the local image background).

Namespace: Euresys::Open_eVision

[C++]

```
float GetStrength() const
```

ESpot::GetType

Get the type of the spot, i.e. whether it is a Scratch or a Particle.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ESpotType GetType() const
```

ESpot::GetWidth

Get the width of the bounding box of spot.

Namespace: Euresys::Open_eVision

[C++]

```
float GetWidth() const
```

4.237. ESpotDetector Class

ESpotDetector is dedicated to the detection of spots, particles and scratches on images. It is especially efficient at detecting faint contaminations on noisy surface images. It produces a list of ESpot objects.

Namespace: Euresys::Open_eVision

Methods

AddDetectedSpotsToDLProject	To be called after ESpotDetector::Detect to add the detected spots to a new or existing EDeepLearningProject. This method returns the number of spots added. The image used by ESpotDetector::Detect is copied to a subdirectory 'Images' of the project file and linked in the project. All detected spots are added to the project as ROI of the image, with the label depending on the spot type ('Particle' or 'Scratch'). Optional parameter roiExtension can be used to increase the ROI size by a pixel amount. It could help the deep learning inference to have a 'context' around the ROI. Automatic image names are generated unless the parameter imageName is set. All images must have a different name. The optional parameter noDefectSamples sets the number of extra ROIs added to the project representing areas without defect. The label of these defects are given by parameter noDefectLabel.
Detect	Performs the spot detection in the given EROIBW8. It returns the number of spot detected. Use ESpotDetector the retrieve the list of detected spots.
Draw	Draw the ESpotDetector.
ESpotDetector	Constructs an ESpotDetector with default configuration.
GetAlignmentArea	Sets/Gets an initial estimate of the rectangle on which spots will be detected in the image. The alignment process is enabled with ESpotDetector::EnableAlignment. To perform the alignment, a tolerance must be set as well, see ESpotDetector::AlignmentTolerance.
GetAlignmentTolerance	Sets/Gets the tolerance around the initial estimate of the alignment area, in pixels. The default is 100 pixels. See also ESpotDetector::AlignmentArea.
GetDetectionThreshold	Gets the detection threshold for the given spot type.
GetDLClassifierPath	Opens a DL classifier and use it to classify all further detected spots. The resulting class is stored in ESpot. Use ESpotDetector::UnsetDLClassifier to remove the Deep Learning classification.
GetDLExecutionSettings	Controls the execution settings of the DL Classifier. Set the class EDeepLearningExecutionSettings to choose between engines, devices and other inference parameters.
GetEnableAlignment	Enables or disables the optional alignment process. The alignment area and the tolerance must be set as well, see ESpotDetector::AlignmentArea and ESpotDetector::AlignmentTolerance.
GetEnableType	Returns the state for the detection of a spot type. See also ESpotDetector::SetEnableType.
GetGuardBand	Gets the guard band borders in pixels. See also ESpotDetector::SetGuardBand.
GetMinimumArea	Gets the minimum spot area in pixels.

GetNumSpots	Gets the number of spots detected by a previous call to ESpotDetector::Detect.
GetPolarity	Gets the spot polarity configuration. See also ESpotDetector::SetPolarity.
GetSizes	Gets the spot size range configuration.
GetSourceROI	Gets the last used Region Of Interest, after optionnal alignment and guard band clipping. See also ESpotDetector::EnableAlignment and ESpotDetector::SetGuardBand.
GetSpots	Gets the spots detected by a previous call to ESpotDetector::Detect.
IsDLClassifierSet	Returns True if a classifier has been loaded. See also ESpotDetector and ESpotDetector::UnsetDLClassifier.
Load	Loads the ESpotDetector. The given ESerializer must have been created for reading.
operator=	Assignment operator
operator==	Comparison operator
Save	Saves the ESpotDetector. The given ESerializer must have been created for writing.
SetAlignmentArea	Sets/Gets an initial estimate of the rectangle on which spots will be detected in the image. The alignment process is enabled with ESpotDetector::EnableAlignment. To perform the alignment, a tolerance must be set as well, see ESpotDetector::AlignmentTolerance.
SetAlignmentTolerance	Sets/Gets the tolerance around the initial estimate of the alignment area, in pixels. The default is 100 pixels. See also ESpotDetector::AlignmentArea.
SetDetectionThreshold	Sets the detection threshold. Default: 20 for ESpotType_Particle, 10 for ESpotType_Scratch.
SetDLClassifierPath	Opens a DL classifier and use it to classify all further detected spots. The resulting class is stored in ESpot. Use ESpotDetector::UnsetDLClassifier to remove the Deep Learning classification.
SetDLExecutionSettings	Controls the execution settings of the DL Classifier. Set the class EDeepLearningExecutionSettings to choose between engines, devices and other inference parameters.
SetEnableAlignment	Enables or disables the optional alignment process. The alignment area and the tolerance must be set as well, see ESpotDetector::AlignmentArea and ESpotDetector::AlignmentTolerance.
SetEnableType	Enables or disables the detection of a spot type. Default: true for ESpotType_Particle, false for ESpotType_Scratch. See also ESpotDetector::GetEnableType.
SetGuardBand	Sets a border, in pixels, to exclude perturbed image data and avoid misdetections near the edges of the alignment area. The default is 0.
SetMinimumArea	Sets the minimum spot area in pixels. Default: 20 for ESpotType_Particle and 100 for ESpotType_Scratch.

SetPolarity	Sets the spot polarity configuration. Default: ESpotPolarity_Any for ESpotType_Particle and ESpotType_Scratch.
SetSizes	Sets the spot size range configuration. Default: 5 and 64 for ESpotType_Particle, 8 and 256 for ESpotType_Scratch.
UnsetDLClassifier	Disable the usage of the Deep Learning classifier. Throws an exception if no DL classifier is loaded. See also ESpotDetector and ESpotDetector::IsDLClassifierSet.

ESpotDetector::AddDetectedSpotsToDLProject

To be called after ESpotDetector::Detect to add the detected spots to a new or existing EDeepLearningProject. This method returns the number of spots added.
The image used by ESpotDetector::Detect is copied to a subdirectory 'Images' of the project file and linked in the project. All detected spots are added to the project as ROI of the image, with the label depending on the spot type ('Particle' or 'Scratch'). Optional parameter roiExtension can be used to increase the ROI size by a pixel amount. It could help the deep learning inference to have a 'context' around the ROI. Automatic image names are generated unless the parameter imageName is set. All images must have a different name. The optional parameter noDefectSamples sets the number of extra ROIs added to the project representing areas without defect. The label of these defects are given by parameter noDefectLabel.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 AddDetectedSpotsToDLProject(
 const std::string& projectPath,
 OEV_UINT32 roiExtension,
 const std::string& imageName,
 OEV_UINT32 noDefectSamples,
 const std::string& noDefectLabel
 )
```

Parameters

projectPath
The path to the Deep Learning project (usually a file with edlproject extension).

roiExtension
The ROI extension in pixels. Default value is 5.

imageName
Override the default image name.

noDefectSamples
Number of extra ROI to be created to represent areas without defect. Default value is 0.

noDefectLabel
The label for the extra ROI (if noDefectSamples>0). Default value is 'NotADefect'.

ESpotDetector::GetAlignmentArea

ESpotDetector::SetAlignmentArea

Sets/Gets an initial estimate of the rectangle on which spots will be detected in the image. The alignment process is enabled with ESpotDetector::EnableAlignment. To perform the alignment, a tolerance must be set as well, see ESpotDetector::AlignmentTolerance.

Namespace: Euresys::Open_eVision

[C++]

```
const ERectangle& GetAlignmentArea() const

void SetAlignmentArea(const ERectangle& rectangle)
```

ESpotDetector::GetAlignmentTolerance

ESpotDetector::SetAlignmentTolerance

Sets/Gets the tolerance around the initial estimate of the alignment area, in pixels. The default is 100 pixels. See also ESpotDetector::AlignmentArea.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetAlignmentTolerance() const

void SetAlignmentTolerance(OEV_UINT32 tolerance)
```

ESpotDetector::Detect

Performs the spot detection in the given EROIBW8. It returns the number of spot detected. Use ESpotDetector the retrieve the list of detected spots.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 Detect(
 EROIBW8& sourceImage
 )
```

Parameters

sourceImage

The image in which the spots are detected.

ESpotDetector::GetDLClassifierPath

ESpotDetector::SetDLClassifierPath

Opens a DL classifier and use it to classify all further detected spots. The resulting class is stored in ESpot. Use ESpotDetector::UnsetDLClassifier to remove the Deep Learning classification.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetDLClassifierPath() const

void SetDLClassifierPath(const std::string& classifierPath)
```

ESpotDetector::GetDLExecutionSettings

ESpotDetector::SetDLExecutionSettings

Controls the execution settings of the DL Classifier. Set the class EDeepLearningExecutionSettings to choose between engines, devices and other inference parameters.

Namespace: Euresys::Open_eVision

[C++]

```
EDeepLearningExecutionSettings GetDLExecutionSettings() const

void SetDLExecutionSettings(const EDeepLearningExecutionSettings& settings)
```

ESpotDetector::Draw

Draw the ESpotDetector.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* drawAdapter,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

drawAdapter

-

color

Color of the drawn rectangle.

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0 (default), the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

graphicContext

-

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ESpotDetector::GetEnableAlignment

ESpotDetector::SetEnableAlignment

Enables or disables the optional alignment process. The alignment area and the tolerance must be set as well, see ESpotDetector::AlignmentArea and ESpotDetector::AlignmentTolerance.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetEnableAlignment() const

void SetEnableAlignment(bool state)
```

ESpotDetector::ESpotDetector

Constructs an ESpotDetector with default configuration.

Namespace: Euresys::Open_eVision

```
[C++]
void ESpotDetector(
 )
void ESpotDetector(
 const ESpotDetector& other
 )
```

Parameters

other

-

ESpotDetector::GetDetectionThreshold

Gets the detection threshold for the given spot type.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetDetectionThreshold(
 Euresys::Open_eVision::ESpotType type
 )
```

Parameters

type

The selected spot type as an ESpotType.

ESpotDetector::GetEnableType

Returns the state for the detection of a spot type. See also ESpotDetector::SetEnableType.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetEnableType(
 Euresys::Open_eVision::ESpotType type
 )
```

Parameters

type

The selected spot type as an ESpotType.

ESpotDetector::GetGuardBand

Gets the guard band borders in pixels. See also ESpotDetector::SetGuardBand.

Namespace: Euresys::Open_eVision

```
[C++]
void GetGuardBand(
 OEV_UINT32& left,
 OEV_UINT32& right,
 OEV_UINT32& top,
 OEV_UINT32& bottom
 )
```

Parameters

left
Size of the left border.

right
Size of the right border.

top
Size of the top border.

bottom
Size of the bottom border.

ESpotDetector::GetMinimumArea

Gets the minimum spot area in pixels.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMinimumArea(
 Euresys::Open_eVision::ESpotType type
 )
```

Parameters

type
The selected spot type as an ESpotType.

ESpotDetector::GetPolarity

Gets the spot polarity configuration. See also ESpotDetector::SetPolarity.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ESpotPolarity GetPolarity(
 Euresys::Open_eVision::ESpotType type
 )
```

Parameters

type

The selected spot type as an ESpotType.

ESpotDetector::GetSizes

Gets the spot size range configuration.

Namespace: Euresys::Open_eVision

[C++]

```
void GetSizes(
 Euresys::Open_eVision::ESpotType type,
 OEV_UINT32& first,
 OEV_UINT32& second
 )
```

Parameters

type

The selected spot type as an ESpotType.

first

First size parameter, depending on the spot type: minimum bbox edge length for ESpotType_Particle, minimum width for ESpotType_Scratch.

second

Second size parameter, depending on the spot type: maximum bbox edge length for ESpotType_Particle, minimum length for ESpotType_Scratch.

ESpotDetector::IsDLClassifierSet

Returns True if a classifier has been loaded.
See also ESpotDetector and ESpotDetector::UnsetDLClassifier.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsDLClassifierSet(
 )
```

ESpotDetector::Load

Loads the ESpotDetector. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

ESpotDetector::GetNumSpots

Gets the number of spots detected by a previous call to ESpotDetector::Detect.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSpots() const
```

ESpotDetector::operator=

Assignment operator

Namespace: Euresys::Open_eVision

```
[C++]
ESpotDetector& operator=(
 const ESpotDetector& other
 )
```

Parameters

other

-

ESpotDetector::operator==

Comparison operator

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ESpotDetector& other
 )
```

Parameters

other

The other object.

ESpotDetector::Save

Saves the ESpotDetector. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

ESpotDetector::SetDetectionThreshold

Sets the detection threshold. Default: 20 for Particle, 10 for Scratch.

Namespace: Euresys::Open_eVision

```
[C++]
void SetDetectionThreshold(
 Euresys::Open_eVision::ESpotType type,
 OEV_UINT32 threshold
 )
```

Parameters

type

The selected spot type as an ESpotType.

threshold

The threshold on the pixel intensity difference with background. A low threshold (5-10) implies a sensitive detection, with possibly a lot of detected spots. A high threshold (30-40) implies a conservative detection, with possible a few or no spot detected.

ESpotDetector::SetEnableType

Enables or disables the detection of a spot type. Default: true for Particle, false for Scratch. See also ESpotDetector::GetEnableType.

Namespace: Euresys::Open_eVision

[C++]

```
void SetEnableType(
 Euresys::Open_eVision::ESpotType type,
 bool state
 )
```

Parameters

type

The selected spot type as an ESpotType.

state

The state of the detection.

ESpotDetector::SetGuardBand

Sets a border, in pixels, to exclude perturbed image data and avoid misdetections near the edges of the alignment area. The default is 0.

Namespace: Euresys::Open_eVision

[C++]

```
void SetGuardBand(
 OEV_UINT32 left,
 OEV_UINT32 right,
 OEV_UINT32 top,
 OEV_UINT32 bottom
 )
```

Parameters

left
Size of the left border.
right
Size of the right border.
top
Size of the top border.
bottom
Size of the bottom border.

ESpotDetector::SetMinimumArea

Sets the minimum spot area in pixels. Default: 20 for Particle and 100 for Scratch.

Namespace: Euresys::Open_eVision

[C++]

```
void SetMinimumArea(
 Euresys::Open_eVision::ESpotType type,
 OEV_UINT32 area
 )
```

Parameters

type
The selected spot type as an ESpotType.
area
-

ESpotDetector::SetPolarity

Sets the spot polarity configuration. Default: Any for Particle and Scratch.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPolarity(
 Euresys::Open_eVision::ESpotType type,
 Euresys::Open_eVision::ESpotPolarity polarity
 )
```

Parameters

type
The selected spot type as an ESpotType.
polarity
The polarity for the given spot type as an ESpotPolarity.

ESpotDetector::SetSizes

Sets the spot size range configuration. Default: 5 and 64 for Particle, 8 and 256 for Scratch.

Namespace: Euresys::Open_eVision

```
[C++]
void SetSizes(
 Euresys::Open_eVision::ESpotType type,
 OEV_UINT32 first,
 OEV_UINT32 second
 )
```

Parameters

type

The selected spot type as an ESpotType.

first

First size parameter, depending on the spot type: minimum bbox edge length for ESpotType_Particle, minimum width for ESpotType_Scratch.

second

Second size parameter, depending on the spot type: maximum bbox edge length for ESpotType_Particle, minimum length for ESpotType_Scratch.

ESpotDetector::GetSourceROI

Gets the last used Region Of Interest, after optionnal alignment and guard band clipping. See also ESpotDetector::EnableAlignment and ESpotDetector::SetGuardBand.

Namespace: Euresys::Open_eVision

```
[C++]
const EROIBW8& GetSourceROI() const
```

ESpotDetector::GetSpots

Gets the spots detected by a previous call to ESpotDetector::Detect.

Namespace: Euresys::Open_eVision

```
[C++]
std::vector<Euresys::Open_eVision::ESpot> GetSpots() const
```

ESpotDetector::UnsetDLClassifier

Disable the usage of the Deep Learning classifier.
Throws an exception if no DL classifier is loaded.
See also ESpotDetector and ESpotDetector::IsDLClassifierSet.

Namespace: Euresys::Open_eVision

```
[C++]
void UnsetDLClassifier(
 )
```

4.238. EStatistics Class

Calculates various statistics on the pixels values of an EDepthMap or EZMap.

Namespace: Euresys::Open_eVision::Easy3D

Methods

ComputeAverageMap	Given an input EDepthMap or EZMap, calculates a depthmap where every pixel is the average of the input DepthMap pixels within a window centered on that pixel.
ComputePixelStatistics	Calculates the minimum, maximum, the average and optionally the standard deviation of the pixels values of an EDepthMap or EZMap. EStatistics::ComputePixelStatistics does not take the resolution of the depthmap into account: it calculates statistics on the pixels gray values.
ComputeStandardDeviationMap	Given an input EDepthMap or EZMap, calculates an image where every pixel is the standard deviation of the input depthmap or ZMap pixels within a window centered on that pixel.
ComputeStatistics	Calculates the minimum, maximum, the average and optionally the standard deviation of an EDepthMap or EZMap values. The standard deviation is optionally calculated. EStatistics::ComputeStatistics takes the resolution of the depthmap or the resolution of the ZMap into account: it calculates statistics on the metric values:

EStatistics::ComputeAverageMap

Given an input EDepthMap or EZMap, calculates a depthmap where every pixel is the average of the input DepthMap pixels within a window centered on that pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ComputeAverageMap(
 const EDepthMap16& sourceMap,
 EDepthMap16& destinationMap,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeAverageMap(
 const EDepthMap8& sourceMap,
 EDepthMap8& destinationMap,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeAverageMap(
 const EZMap16& sourceMap,
 EZMap16& destinationMap,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeAverageMap(
 const EZMap8& sourceMap,
 EZMap8& destinationMap,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )
```

Parameters

sourceMap

The input DepthMap/ZMap.

destinationMap

The destination DepthMap/ZMap. It should have the same dimensions as the input depthmap.

halfKernelSize

The half-size of the window used for the calculation of the standard deviation. The filter window size (= kernel size) is halfKernelSize * 2 + 1, should be positive, smaller than (or equal to) the image size, and may not exceed 256.

minValidRatio

required ratio of valid pixels in the filter window to process the calculation. If not enough, the output pixel will be marked as invalid. The default value of this parameter is 0.25

fillUndefinedPixels

This boolean controls how undefined pixels are handled: either they filled by the calculated average value, or they are left undefined (default behavior).

EStatistics::ComputePixelStatistics

Calculates the minimum, maximum, the average and optionally the standard deviation of the pixels values of an EDepthMap or EZMap.
EStatistics::ComputePixelStatistics does not take the resolution of the depthmap into account: it calculates statistics on the pixels gray values.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ComputePixelStatistics(
 const EDepthMap8& sourceMap,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EDepthMap16& sourceMap,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EDepthMap32f& sourceMap,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EDepthMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EDepthMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )
```

```
void ComputePixelStatistics(
 const EDepthMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EDepthMap8& sourceMap,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EDepthMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EDepthMap16& sourceMap,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EDepthMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EDepthMap32f& sourceMap,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average,
 float& stddev
 )
```

```
void ComputePixelStatistics(
 const EDepthMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EZMap8& sourceMap,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EZMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EZMap16& sourceMap,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EZMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EZMap32f& sourceMap,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average
 )
```

```
void ComputePixelStatistics(
 const EZMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average
 )

void ComputePixelStatistics(
 const EZMap8& sourceMap,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EZMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW8& minimumValue,
 EBW8& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EZMap16& sourceMap,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EZMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW16& minimumValue,
 EBW16& maximumValue,
 float& average,
 float& stddev
 )

void ComputePixelStatistics(
 const EZMap32f& sourceMap,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average,
 float& stddev
 )
```

```
void ComputePixelStatistics(
 const EZMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 EBW32f& minimumValue,
 EBW32f& maximumValue,
 float& average,
 float& stddev
 )
```

Parameters

sourceMap
The input DepthMap/ZMap.

validCount
Variable to store the number of valid pixels in sourceMap.

minimumValue
Variable to store the minimum value.

maximumValue
Variable to store the maximum value.

average
Variable to store the average value.

region
The ERegion where the statistics has to be calculated.

stddev
Variable to store the standard deviation.

EStatistics::ComputeStandardDeviationMap

Given an input EDepthMap or EZMap, calculates an image where every pixel is the standard deviation of the input depthmap or ZMap pixels within a window centered on that pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ComputeStandardDeviationMap(
 const EDepthMap8& sourceMap,
 EImageBW8& destinationImage,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )
```

```
void ComputeStandardDeviationMap(
 const EDepthMap16& sourceMap,
 EImageBW16& destinationImage,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeStandardDeviationMap(
 const EDepthMap8& sourceMap,
 EImageBW16& destinationImage,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeStandardDeviationMap(
 const EZMap16& sourceMap,
 EImageBW16& destinationImage,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeStandardDeviationMap(
 const EZMap8& sourceMap,
 EImageBW8& destinationImage,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )

void ComputeStandardDeviationMap(
 const EZMap8& sourceMap,
 EImageBW16& destinationImage,
 short halfKernelSize,
 float minValidRatio,
 bool fillUndefinedPixels
 )
```

Parameters

sourceMap

The input DepthMap/ZMap.

destinationImage

The destination image. It should have the same dimensions as the input depthmap.

halfKernelSize

The half-size of the window used for the calculation of the standard deviation.
The filter window size (= kernel size) is halfKernelSize * 2 + 1, should be positive, smaller than (or equal to) the image size, and may not exceed 256.

minValidRatio

required ratio of valid pixels in the filter window to process the calculation.
If not enough, the output pixel value will be 0. The default value of this parameter is 0.25 .

fillUndefinedPixels

This boolean controls how undefined pixels are handled: either they filled by the calculated standard deviation, or they are left undefined (default behavior).

Remarks

When the input depthmap is on 8 bits and the destination image is on 16 bits, the resulting standard deviation scale is 256 times larger than in the source depthmap.

EStatistics::ComputeStatistics

Calculates the minimum, maximum, the average and optionally the standard deviation of an EDepthMap or EZMap values.
The standard deviation is optionally calculated. EStatistics::ComputeStatistics takes the resolution of the depthmap or the resolution of the ZMap into account: it calculates statistics on the metric values:

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ComputeStatistics(
 const EDepthMap8& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EDepthMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )
```

```
void ComputeStatistics(
 const EDepthMap16& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EDepthMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EDepthMap32f& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EDepthMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EZMap8& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EZMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )
```

```
void ComputeStatistics(
 const EZMap16& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EZMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EZMap32f& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EZMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average
 )

void ComputeStatistics(
 const EDepthMap8& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EDepthMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )
```

```
void ComputeStatistics(
 const EDepthMap16& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EDepthMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EDepthMap32f& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EDepthMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EZMap8& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )
```

```
void ComputeStatistics(
 const EZMap8& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EZMap16& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EZMap16& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EZMap32f& sourceMap,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )

void ComputeStatistics(
 const EZMap32f& sourceMap,
 ERegion& region,
 OEV_UINT32& validCount,
 float& minimumValue,
 float& maximumValue,
 float& average,
 float& stddev
 )
```

Parameters

sourceMap
The input DepthMap/ZMap.
validCount
Variable to store the number of valid pixels in sourceMap.
minimumValue
Variable to store the minimum value.
maximumValue
Variable to store the maximum value.
average
Variable to store the average value.
region
The ERegion where the statistics has to be calculated.
stddev
Variable to store the standard deviation value.

4.239. EStringPair Class

Represent a Key/Value pair of strings.

Namespace: Euresys::Open_eVision

Methods

EStringPair	Constructs an EStringPair context.
GetKey	Returns the key.
GetValue	Returns the value.
operator=	Assignment operator

EStringPair::EStringPair

Constructs an EStringPair context.

Namespace: Euresys::Open_eVision

[C++]

```
void EStringPair(
 const std::string& key,
 const std::string& value
 )

void EStringPair(
 const EStringPair& other
 )
```

Parameters

key
The key.
value
The value associated to the key.
other
-

EStringPair::GetKey

Returns the key.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetKey()
```

EStringPair::operator=

Assignment operator

Namespace: Euresys::Open_eVision

[C++]

```
EStringPair& operator=(
 const EStringPair& other
 )
```

Parameters

other
-

EStringPair::GetValue

Returns the value.

Namespace: Euresys::Open_eVision

[C++]

```
std::string GetValue()
```

4.240. ESupervisedSegmenter Class

Supervised segmentation tool.

The supervised segmentation tool segments the pixels of an image into various labels by learning a deep learning model on a dataset of segmented images.
The tool can work with images of any resolution higher than ESupervisedSegmenter::PatchSize by merging the results obtained by applying the deep neural network using a sliding window algorithm. The overlap between the sliding windows is controled by ESupervisedSegmenter::SamplingDensity.
For defect detection/foreground blobs detection, the supervised segmenter offers a tradeoff between a high good detection rate and a high defect detection rate through a classification threshold that can be configured after training (ESupervisedSegmenter::ClassificationThreshold).

Base Class:EDeepLearningTool

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Apply	Applies the unsupervised segmenter on the given image and its mask region. We recommend to use pointer-based versions of this method to avoid unnecessary image and result copies.
ESupervisedSegmenter	Constructs a ESupervisedSegmenter object.
Evaluate	Evaluates the dataset.
GetCapacity	Capacity of the ESupervisedSegmenter. A higher capacity makes the supervised segmenter capable of learning more information at the cost of a slower processing speed.
GetClassificationThreshold	Classification threshold for determining whether an image contains blobs with a label different than background. By default, its value is set during training to maximize the weighted accuracy (see ESupervisedSegmenterMetrics).
GetForceGrayscale	Forces the ESupervisedSegmenter to convert all images to grayscale (default: false). Setting this property to false will make the underlying neural network operates with the same number of channels as the images in the dataset used for training. Otherwise, color images will be converted to grayscale before using them.
GetInferenceScale	The effective scale applied when performing inference. If ESupervisedSegmenter::ScaleDisabledAtInference is true, it is equal to 1.0. Otherwise, it is equal to ESupervisedSegmenter::Scale.
GetNumPatchesForImage	Number of patches that will be extracted from an input image to perform inference. For EasyClassify and EasyLocate, this will always be equal to 1. For EasySegment, the number of patches will depend on the scale, patch size and sampling density parameters.

GetPatchSize	Patch size (width and height of the patches processed by the neural network).
GetSamplingDensity	Sampling density (default value: 1.25, its value must be equal or greater than 1). The sampling density is the parameter of the sliding window algorithm used for processing a whole image using subwindows of size ESupervisedSegmenter::PatchSize. It indicates how much overlap there will be between the image patches: the stride between two consecutive patches is ESupervisedSegmenter::PatchSize / ESupervisedSegmenter::SamplingDensity.
GetScale	Down-scaling applied to images before processing them. Its value is between 0 and 1 (default value: 1).
GetScaleDisabledAtInference	Whether to apply the scale parameter at inference or not. If you disable the scale at inference, make sure your images are already scaled. For example, if you trained with images of 1024x1024 and a scale of 0.5, the inference images should be 512 x 512 images with the same field of view as the training images.
GetToolType	Type of the deep learning tool.
GetTrainingMetrics	Training metrics at the given iteration
GetValidationMetrics	Validation metrics at the given iteration
operator=	Assignment operator
SerializeSettings	Serializes the settings of the supervised segmenter.
SetCapacity	Capacity of the ESupervisedSegmenter. A higher capacity makes the supervised segmenter capable of learning more information at the cost of a slower processing speed.
SetClassificationThreshold	Classification threshold for determining whether an image contains blobs with a label different than background. By default, its value is set during training to maximize the weighted accuracy (see ESupervisedSegmenterMetrics).
SetForceGrayscale	Forces the ESupervisedSegmenter to convert all images to grayscale (default: false). Setting this property to false will make the underlying neural network operates with the same number of channels as the images in the dataset used for training. Otherwise, color images will be converted to grayscale before using them.
SetPatchSize	Patch size (width and height of the patches processed by the neural network).
SetSamplingDensity	Sampling density (default value: 1.25, its value must be equal or greater than 1). The sampling density is the parameter of the sliding window algorithm used for processing a whole image using subwindows of size ESupervisedSegmenter::PatchSize. It indicates how much overlap there will be between the image patches: the stride between two consecutive patches is ESupervisedSegmenter::PatchSize / ESupervisedSegmenter::SamplingDensity.
SetScale	Down-scaling applied to images before processing them. Its value is between 0 and 1 (default value: 1).

SetScaleDisabledAtInference	Whether to apply the scale parameter at inference or not. If you disable the scale at inference, make sure your images are already scaled. For example, if you trained with images of 1024x1024 and a scale of 0.5, the inference images should be 512 x 512 images with the same field of view as the training images.

ESupervisedSegmenter::Apply

Applies the unsupervised segmenter on the given image and its mask region.
We recommend to use pointer-based versions of this method to avoid unnecessary image and result copies.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterResult Apply(
 const EBaseROI& img
 )

ESupervisedSegmenterResult Apply(
 const EBaseROI& img,
 const ERegion& mask
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW8>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW8>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW16>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageBW16>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageC24>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterResult> Apply(
 const std::vector<Euresys::Open_eVision::EImageC24>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )
```

```
void Apply(
 const std::vector<Euresys::Open_eVision::EBaseROI*>& imgs,
 const std::vector<Euresys::Open_
eVision::EasyDeepLearning::ESupervisedSegmenterResult*>& results
 )
void Apply(
 const std::vector<Euresys::Open_eVision::EBaseROI*>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion*>& masks,
 const std::vector<Euresys::Open_
eVision::EasyDeepLearning::ESupervisedSegmenterResult*>& results
 )
```

Parameters

img
Image to classify and segment defects in.

mask
Mask region of the image.

imgs
Vector of image to classify and segment defects in.

masks
Vector of mask regions for the images.

results
-

ESupervisedSegmenter::GetCapacity

ESupervisedSegmenter::SetCapacity

Capacity of the ESupervisedSegmenter.
A higher capacity makes the supervised segmenter capable of learning more information at the cost of a slower processing speed.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterCapacity GetCapacity()
const
void SetCapacity(Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterCapacity
capacity)
```

ESupervisedSegmenter::GetClassificationThreshold

ESupervisedSegmenter::SetClassificationThreshold

Classification threshold for determining whether an image contains blobs with a label different than background.
By default, its value is set during training to maximize the weighted accuracy (see ESupervisedSegmenterMetrics).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetClassificationThreshold() const

void SetClassificationThreshold(float threshold)
```

ESupervisedSegmenter::ESupervisedSegmenter

Constructs a ESupervisedSegmenter object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ESupervisedSegmenter(
 )

void ESupervisedSegmenter(
 const ESupervisedSegmenter& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenter object that should be copied

ESupervisedSegmenter::Evaluate

Evaluates the dataset.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterMetrics Evaluate(
 const EClassificationDataset& dataset
 )
```

Parameters

dataset

Dataset to evaluate

ESupervisedSegmenter::GetForceGrayscale

ESupervisedSegmenter::SetForceGrayscale

Forces the ESupervisedSegmenter to convert all images to grayscale (default: false).
Setting this property to false will make the underlying neural network operates with the same number of channels as the images in the dataset used for training. Otherwise, color images will be converted to grayscale before using them.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetForceGrayscale() const

void SetForceGrayscale(bool forceGrayscale)
```

ESupervisedSegmenter::GetNumPatchesForImage

Number of patches that will be extracted from an input image to perform inference.
For EasyClassify and EasyLocate, this will always be equal to 1.
For EasySegment, the number of patches will depend on the scale, patch size and sampling density parameters.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumPatchesForImage(
 int imageWidth,
 int imageHeight
 )
```

Parameters

imageWidth

Width of the image for which to get the number of patch

imageHeight

Height of the image for which to get the number of patch

ESupervisedSegmenter::GetTrainingMetrics

Training metrics at the given iteration

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterMetrics GetTrainingMetrics(
 int iteration
 )
```

Parameters

iteration

Iteration at which to get the metrics

ESupervisedSegmenter::GetValidationMetrics

Validation metrics at the given iteration

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterMetrics GetValidationMetrics(
 int iteration
 )
```

Parameters

iteration

Iteration at which to get the metrics

ESupervisedSegmenter::GetInferenceScale

The effective scale applied when performing inference.
If ESupervisedSegmenter::ScaleDisabledAtInference is true, it is equal to 1.0. Otherwise, it is equal to ESupervisedSegmenter::Scale.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetInferenceScale() const
```

ESupervisedSegmenter::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenter& operator=(
 const ESupervisedSegmenter& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenter object that should be copied

ESupervisedSegmenter::GetPatchSize

ESupervisedSegmenter::SetPatchSize

Patch size (width and height of the patches processed by the neural network).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetPatchSize() const
void SetPatchSize(int patchSize)
```

Remarks

There are three supported patch size: 64x64, 128x128, and 256x256. By default, the patch size is 0 and it means that the patch size will be 128x128 if all images in the training and validation dataset have a higher resolution or the patch size will be 64x64.

ESupervisedSegmenter::GetSamplingDensity

ESupervisedSegmenter::SetSamplingDensity

Sampling density (default value: 1.25, its value must be equal or greater than 1).
The sampling density is the parameter of the sliding window algorithm used for processing a whole image using subwindows of size ESupervisedSegmenter::PatchSize. It indicates how much overlap there will be between the image patches: the stride between two consecutive patches is ESupervisedSegmenter::PatchSize / ESupervisedSegmenter::SamplingDensity.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSamplingDensity() const
void SetSamplingDensity(float density)
```

ESupervisedSegmenter::GetScale

ESupervisedSegmenter::SetScale

Down-scaling applied to images before processing them. Its value is between 0 and 1 (default value: 1).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetScale() const
void SetScale(float scale)
```

ESupervisedSegmenter::GetScaleDisabledAtInference

ESupervisedSegmenter::SetScaleDisabledAtInference

Whether to apply the scale parameter at inference or not.
If you disable the scale at inference, make sure your images are already scaled. For example, if you trained with images of 1024x1024 and a scale of 0.5, the inference images should be 512 x 512 images with the same field of view as the training images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool GetScaleDisabledAtInference() const
void SetScaleDisabledAtInference(bool disable)
```

ESupervisedSegmenter::SerializeSettings

Serializes the settings of the supervised segmenter.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer
Pointer to ESerializer

ESupervisedSegmenter::GetToolType

Type of the deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetToolType() const
```

4.241. ESupervisedSegmenterBlob Class

A blob detected by a supervised segmentation tool (see ESupervisedSegmenter and ESupervisedSegmenterResult).
A blob is a connected component from a label that is not the "Background" label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

ESupervisedSegmenterBlob	Copy constructor of an ESupervisedSegmenterBlob object.
GetArea	Area of the blob in pixels.
GetAverageBackgroundProbability	Average probability of the background label (ESupervisedSegmenterBlob::Label) in the blob.
GetAverageProbability	Average probability of the predicted label (ESupervisedSegmenterBlob::Label) in the blob.
GetLabel	Label of the blob.
GetMaxBackgroundProbability	Maximum probability of the background label (ESupervisedSegmenterBlob::Label) in the blob.
GetMaxProbability	Maximum probability of the predicted label (ESupervisedSegmenterBlob::Label) in the blob.
GetMinBackgroundProbability	Minimum probability of the background label (ESupervisedSegmenterBlob::Label) in the blob.
GetMinProbability	Minimum probability of the predicted label (ESupervisedSegmenterBlob::Label) in the blob.
GetRegion	Region of the blob.
operator=	Assignment operator

ESupervisedSegmenterBlob::GetArea

Area of the blob in pixels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetArea() const
```

ESupervisedSegmenterBlob::GetAverageBackgroundProbability

Average probability of the background label (ESupervisedSegmenterBlob::Label) in the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAverageBackgroundProbability() const
```

ESupervisedSegmenterBlob::GetAverageProbability

Average probability of the predicted label (ESupervisedSegmenterBlob::Label) in the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetAverageProbability() const
```

ESupervisedSegmenterBlob::ESupervisedSegmenterBlob

Copy constructor of an ESupervisedSegmenterBlob object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ESupervisedSegmenterBlob(
 const ESupervisedSegmenterBlob& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterBlob object that should be copied

ESupervisedSegmenterBlob::GetLabel

Label of the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetLabel() const
```

ESupervisedSegmenterBlob::GetMaxBackgroundProbability

Maximum probability of the background label (ESupervisedSegmenterBlob::Label) in the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxBackgroundProbability() const
```

ESupervisedSegmenterBlob::GetMaxProbability

Maximum probability of the predicted label (ESupervisedSegmenterBlob::Label) in the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMaxProbability() const
```

ESupervisedSegmenterBlob::GetMinBackgroundProbability

Minimum probability of the background label (ESupervisedSegmenterBlob::Label) in the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinBackgroundProbability() const
```

ESupervisedSegmenterBlob::GetMinProbability

Minimum probability of the predicted label (ESupervisedSegmenterBlob::Label) in the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetMinProbability() const
```

ESupervisedSegmenterBlob::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ESupervisedSegmenterBlob& operator=(
 const ESupervisedSegmenterBlob& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterBlob object used for the assignment

ESupervisedSegmenterBlob::GetRegion

Region of the blob.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
ERegion GetRegion() const
```

4.242. ESupervisedSegmenterMetrics Class

Collection of metrics used to evaluate the state of a ESupervisedSegmenter tool.
A metric is a value summarizing the quality of a collection of supervised segmentation results (see ESupervisedSegmenterResult) with respect to their ground truth.
New results can be added to the object individually with ESupervisedSegmenterMetrics.

The ESupervisedSegmenterMetrics contains three types of metrics:
- pixel based metrics that are related to the quality of the segmentation masks
- blob based metrics that are related to the ability of the supervised segmentation tool to detect foreground blobs
- defect detection metrics that are related to the ability of the supervised segmentation tool to differentiate between images that contains foreground pixels (defective images) and images that are entirely background (good images).

The pixel metrics are the error (see ESupervisedSegmenterMetrics::Error), the pixel accuracy (see ESupervisedSegmenterMetrics::PixelAccuracy), the pixel confusion (see ESupervisedSegmenterMetrics::GetPixelConfusion), the pixel per-label accuracy (see ESupervisedSegmenterMetrics::GetPixelLabelAccuracy) and the intersection over union (see ESupervisedSegmenterMetrics).
The blob based metrics are two confusion matrixes (ESupervisedSegmenterMetrics::GetGroundtruthBlobConfusion and ESupervisedSegmenterMetrics::GetPredictedBlobConfusion), and various defect detection metrics (ESupervisedSegmenterMetrics, ESupervisedSegmenterMetrics, and ESupervisedSegmenterMetrics). Note that the metrics for defective blob detection are different from the metrics for defective image detection because, in the case of blob detection, we have no "good" ground truth results.
See EDeepLearningDefectDetectionMetrics for a description of the defect detection metrics. These metrics are available when EDeepLearningDefectDetectionMetrics::IsDefectDetectionMetricsValid is true, i.e. when both images that are entirely background and images with non-background pixels have been added to the metrics.
Most metrics depends upon the ESupervisedSegmenterMetrics.

Base Class:EDeepLearningDefectDetectionMetrics

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

ESupervisedSegmenterMetrics	Constructs an ESupervisedSegmenterMetrics object.
GetBalancedError	Balanced error. The balanced error is the weighted error (ESupervisedSegmenterMetrics) where each label is given an equal weight.
GetBalancedIntersectionOverUnion	Balanced Intersection over Union (IoU). The balanced interesection over union is the weighted intersection over union (ESupervisedSegmenterMetrics::WeightedIntersectionOverUnion) where each label is given equal weight.

GetBalancedPixelAccuracy	Balanced accuracy. The balanced accuracy is the weighted accuracy (ESupervisedSegmenterMetrics::WeightedPixelAccuracy) where each label is given an equal weight.
GetBlobDetectionAveragePrecision	Average precision for blob detection. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label. The average precision for blob detection is the average of the precision (see ESupervisedSegmenterMetrics) over all the possible classification threshold values. As such, the average precision does not depend on a specific classification threshold.
GetBlobDetectionBestFScore	Best F1-Score of the segmenter for blob detection. See ESupervisedSegmenterMetrics::BlobDetectionBestFScoreThreshold for the corresponding threshold. See EDeepLearningDefectDetectionMetrics for a definition of the F1-Score. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.
GetBlobDetectionBestFScoreThreshold	Classification threshold that achieves the best F1-Score for blob detection. See ESupervisedSegmenterMetrics::BlobDetectionBestFScore for the corresponding F1-Score. See EDeepLearningDefectDetectionMetrics for a definition of the F1-Score. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.
GetBlobDetectionFScore	F1-Score for defective blob detection given the current classification threshold (see EDeepLearningDefectDetectionMetrics::ClassificationThreshold). See ESupervisedSegmenterMetrics for the corresponding F1-Score. See EDeepLearningDefectDetectionMetrics for a definition of the F1-Score. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.
GetBlobDetectionPrecision	Precision for blob detection given the current classification threshold (see EDeepLearningDefectDetectionMetrics::ClassificationThreshold). The precision is the proportion of detected defective blobs that match ground truth defective blobs. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.
GetBlobDetectionRecall	Recall for blob detection given the current classification threshold (see EDeepLearningDefectDetectionMetrics::ClassificationThreshold). The recall is the proportion of ground truth defective blobs that are matched to predicted defective blobs. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.
GetError	Error.
GetGroundtruthBlobConfusion	Number of ground truth blobs of the given ground truth label that best match with blobs of the given precited label. See ESupervisedSegmenterMetrics::GetPredictedBlobConfusion for the number of predicted blobs that match to these ground truth blobs.

GetIntersectionOverUnion	Intersection over union. The intersection over union if the ratio between the number of correctly classified pixels from the given label to the total number of pixels that belongs to that label or are predicted as being from that label. Assuming that the given label is the positive class, the intersection over union is expressed as IOU = TP / (TP + FP + FN) where TP is the number of true positive, FP the number of false positive (pixels that belongs to label but are not predicted as label) and FN the number of false negative (pixels predicted as label but that do not belong to label).
GetLabel	Label recognized by the segmenter that produced the results aggregated in the metrics.
GetLabelError	Label error.
GetNormalizedPixelConfusion	Pixel-wise normalized confusion between the given true and predicted labels. The normalized confusion is the ratio of the number of pixels belonging to the 'trueLabel' that are classified as 'predictedLabel' to the total number of pixels belonging to the 'trueLabel'.
GetNumLabels	Number of labels recognized by the segmenter that produced the results aggregated in the metrics.
GetPixelAccuracy	Accuracy. The accuracy is the ratio of the number of correctly classified pixels to the total number of pixels.
GetPixelConfusion	Pixel-wise confusion between the given true and predicted labels. The confusion is the number of pixels belonging to the 'trueLabel' that are classified as 'predictedLabel'.
GetPixelLabelAccuracy	Pixel label accuracy. The label accuracy is the ratio of the number of correctly classified pixels of the given class to the total number of ground truth pixels from that class. If there are no ground truth pixels from the requested label, the method returns '-1'.
GetPredictedBlobConfusion	Number of predicted blobs of the given precited label that match to blobs of the given ground truth label. See ESupervisedSegmenterMetrics::GetGroundtruthBlobConfusion for the number of ground truth blobs that match to these predicted blobs.
GetWeightedError	Weighted error. The weighted error is the weighted average of each label error (ESupervisedSegmenterMetrics::GetLabelError) with respect to the dataset label weights.
GetWeightedIntersectionOverUnion	Weighted Intersection over Union (IoU). The weighted interesection over union is the weighted averaged of each label intersection over union (see ESupervisedSegmenterMetrics::GetIntersectionOverUnion) with respect to the dataset label weights.

GetWeightedPixelAccuracy	Weighted accuracy. The weighted accuracy is the weighted average of each label accuracy (ESupervisedSegmenterMetrics::GetPixelLabelAccuracy) with respect to the dataset label weights.
IsValid	Indicates whether the object contains at least one result.
Load	Loads an supervised segmentation metric. The given ESerializer must have been created for reading.
operator=	Assignment operator.
operator==	Equality operator.
Save	Saves an supervised segmentation metric. The given ESerializer must have been created for writing.

ESupervisedSegmenterMetrics::GetBalancedError

Balanced error.
The balanced error is the weighted error (ESupervisedSegmenterMetrics) where each label is given an equal weight.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBalancedError() const
```

Remarks

The error is also called the cross-entropy loss.

ESupervisedSegmenterMetrics::GetBalancedIntersectionOverUnion

Balanced Intersection over Union (IoU).
The balanced interesection over union is the weighted intersection over union (ESupervisedSegmenterMetrics::WeightedIntersectionOverUnion) where each label is given equal weight.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBalancedIntersectionOverUnion() const
```

ESupervisedSegmenterMetrics::GetBalancedPixelAccuracy

Balanced accuracy.
The balanced accuracy is the weighted accuracy (ESupervisedSegmenterMetrics::WeightedPixelAccuracy) where each label is given an equal weight.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBalancedPixelAccuracy() const
```

ESupervisedSegmenterMetrics::GetBlobDetectionAveragePrecision

Average precision for blob detection.
Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label. The average precision for blob detection is the average of the precision (see ESupervisedSegmenterMetrics) over all the possible classification threshold values. As such, the average precision does not depend on a specific classification threshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBlobDetectionAveragePrecision() const
```

ESupervisedSegmenterMetrics::GetBlobDetectionBestFScore

Best F1-Score of the segmenter for blob detection.
See ESupervisedSegmenterMetrics::BlobDetectionBestFScoreThreshold for the corresponding threshold.
See EDeepLearningDefectDetectionMetrics for a definition of the F1-Score. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBlobDetectionBestFScore() const
```

ESupervisedSegmenterMetrics::GetBlobDetectionBestFScoreThreshold

Classification threshold that achieves the best F1-Score for blob detection.
See ESupervisedSegmenterMetrics::BlobDetectionBestFScore for the corresponding F1-Score.
See EDeepLearningDefectDetectionMetrics for a definition of the F1-Score. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBlobDetectionBestFScoreThreshold() const
```

ESupervisedSegmenterMetrics::GetBlobDetectionFScore

F1-Score for defective blob detection given the current classification threshold (see EDeepLearningDefectDetectionMetrics::ClassificationThreshold).
See ESupervisedSegmenterMetrics for the corresponding F1-Score.
See EDeepLearningDefectDetectionMetrics for a definition of the F1-Score. Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBlobDetectionFScore() const
```

ESupervisedSegmenterMetrics::GetBlobDetectionPrecision

Precision for blob detection given the current classification threshold (see EDeepLearningDefectDetectionMetrics::ClassificationThreshold).
The precision is the proportion of detected defective blobs that match ground truth defective blobs.
Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBlobDetectionPrecision() const
```

ESupervisedSegmenterMetrics::GetBlobDetectionRecall

Recall for blob detection given the current classification threshold (see EDeepLearningDefectDetectionMetrics::ClassificationThreshold).
The recall is the proportion of ground truth defective blobs that are matched to predicted defective blobs.
Blob detection is the ability of the segmenter to correctly detect foreground ground truth blobs, regardless of their specific label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBlobDetectionRecall() const
```

ESupervisedSegmenterMetrics::GetError

Error.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetError() const
```

Remarks

The error is also called the cross-entropy loss.

ESupervisedSegmenterMetrics::ESupervisedSegmenterMetrics

Constructs an ESupervisedSegmenterMetrics object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void ESupervisedSegmenterMetrics(
 )

void ESupervisedSegmenterMetrics(
 const ESupervisedSegmenterMetrics& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterMetrics object that should be copied

ESupervisedSegmenterMetrics::GetGroundtruthBlobConfusion

Number of ground truth blobs of the given ground truth label that best match with blobs of the given precited label.
See ESupervisedSegmenterMetrics::GetPredictedBlobConfusion for the number of predicted blobs that match to these ground truth blobs.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT64 GetGroundtruthBlobConfusion(
 const std::string& groundtruthLabel,
 const std::string& predictedLabel
 )

OEV_UINT64 GetGroundtruthBlobConfusion(
 int groundtruthLabelIndex,
 int predictedLabelIndex
 )
```

Parameters

groundtruthLabel
Ground truth label

predictedLabel
Predicted label

groundtruthLabelIndex
Ground truth label index

predictedLabelIndex
Predicted label index

Remarks

A ground truth blob is matched to the subset of predicted blobs from the same predicted label that has the best intersection over union with the ground truth blob. Otherwise, the ground truth blob is matched to background.

ESupervisedSegmenterMetrics::GetIntersectionOverUnion

Intersection over union.
The intersection over union if the ratio between the number of correctly classified pixels from the given label to the total number of pixels that belongs to that label or are predicted as being from that label.
Assuming that the given label is the positive class, the intersection over union is expressed as IOU = TP / (TP + FP + FN) where TP is the number of true positive, FP the number of false positive (pixels that belongs to label but are not predicted as label) and FN the number of false negative (pixels predicted as label but that do not belong to label).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetIntersectionOverUnion(
 const std::string& label
 )
```

Parameters

label
Label

Remarks

The intersection over union is also called the Jaccard index.

ESupervisedSegmenterMetrics::GetLabel

Label recognized by the segmenter that produced the results aggregated in the metrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::string GetLabel(
 int labelIdx
 )
```

Parameters

labelIdx

Index of the label between 0 and ESupervisedSegmenterMetrics::NumLabels - 1.

ESupervisedSegmenterMetrics::GetLabelError

Label error.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetLabelError(
 const std::string& label
 )
```

Parameters

label

Label

ESupervisedSegmenterMetrics::GetNormalizedPixelConfusion

Pixel-wise normalized confusion between the given true and predicted labels.
The normalized confusion is the ratio of the number of pixels belonging to the 'trueLabel' that are classified as 'predictedLabel' to the total number of pixels belonging to the 'trueLabel'.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetNormalizedPixelConfusion(
 const std::string& trueLabel,
 const std::string& predictedLabel
 )
```

Parameters

trueLabel

True label

predictedLabel

Predicted label

ESupervisedSegmenterMetrics::GetPixelConfusion

Pixel-wise confusion between the given true and predicted labels.
The confusion is the number of pixels belonging to the 'trueLabel' that are classified as 'predictedLabel'.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
OEV_UINT64 GetPixelConfusion(
 const std::string& trueLabel,
 const std::string& predictedLabel
 )
```

Parameters

trueLabel
True label

predictedLabel
Predicted label

ESupervisedSegmenterMetrics::GetPixelLabelAccuracy

Pixel label accuracy.
The label accuracy is the ratio of the number of correctly classified pixels of the given class to the total number of ground truth pixels from that class. If there are no ground truth pixels from the requested label, the method returns '-1'.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetPixelLabelAccuracy(
 const std::string& label
 )
```

Parameters

label
Label

ESupervisedSegmenterMetrics::GetPredictedBlobConfusion

Number of predicted blobs of the given precited label that match to blobs of the given ground truth label.
See ESupervisedSegmenterMetrics::GetGroundtruthBlobConfusion for the number of ground truth blobs that match to these predicted blobs.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
OEV_UINT64 GetPredictedBlobConfusion(
 const std::string& groundtruthLabel,
 const std::string& predictedLabel
 )

OEV_UINT64 GetPredictedBlobConfusion(
 int groundtruthLabelIndex,
 int predictedLabelIndex
 )
```

Parameters

groundtruthLabel

Ground truth label

predictedLabel

Predicted label

groundtruthLabelIndex

Ground truth label index

predictedLabelIndex

Predicted label index

Remarks

A predicted blob may be counted several times in the confusion matrix for predicted blobs:
- Once if the more than 50% of the blob intersects ground truth background.
- Once for each non-background label the predicted blob has an intersection with.

ESupervisedSegmenterMetrics::IsValid

Indicates whether the object contains at least one result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsValid(
 )
```

ESupervisedSegmenterMetrics::Load

Loads an supervised segmentation metric. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

ESupervisedSegmenterMetrics::GetNumLabels

Number of labels recognized by the segmenter that produced the results aggregated in the metrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumLabels() const
```

ESupervisedSegmenterMetrics::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterMetrics& operator=(
 const ESupervisedSegmenterMetrics& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterMetrics object used for the assignment

ESupervisedSegmenterMetrics::operator==

Equality operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool operator==(
 const ESupervisedSegmenterMetrics& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterMetrics object

ESupervisedSegmenterMetrics::GetPixelAccuracy

Accuracy.
The accuracy is the ratio of the number of correctly classified pixels to the total number of pixels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetPixelAccuracy() const
```

Remarks

The accuracy will be skewed towards the most represented labels in the dataset. For example, if your dataset has 1% of defective pixels, the accuracy will mostly reflect the results on the 99% of good pixels and will say nothing about the capability of the model to detect the defect labels.

ESupervisedSegmenterMetrics::Save

Saves an supervised segmentation metric. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

ESupervisedSegmenterMetrics::GetWeightedError

Weighted error.
The weighted error is the weighted average of each label error (ESupervisedSegmenterMetrics::GetLabelError) with respect to the dataset label weights.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetWeightedError() const
```

Remarks

The error is also called the cross-entropy loss.

ESupervisedSegmenterMetrics::GetWeightedIntersectionOverUnion

Weighted Intersection over Union (IoU).
The weighted interesection over union is the weighted averaged of each label intersection over union (see ESupervisedSegmenterMetrics::GetIntersectionOverUnion) with respect to the dataset label weights.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetWeightedIntersectionOverUnion() const
```

ESupervisedSegmenterMetrics::GetWeightedPixelAccuracy

Weighted accuracy.
The weighted accuracy is the weighted average of each label accuracy (ESupervisedSegmenterMetrics::GetPixelLabelAccuracy) with respect to the dataset label weights.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetWeightedPixelAccuracy() const
```

4.243. ESupervisedSegmenterResult Class

An ESupervisedSegmenterResult object represents the result of a supervised segmentater.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Draw	Draws the segmentation. The detected segment are drawn with the label color specified in the dataset used for training. The pixel classified as background are transparent.
ESupervisedSegmenterResult	Constructs a non-valid ESupervisedSegmenterResult.
GetBlobs	Detected blobs for all labels or for the specified label.
GetClassificationThreshold	Classification threshold for determining whether the image contains blobs with a label different than background. By default, its value is given by the supervised segmenter that produced this result.
GetColorizedSegmentation	Colorized version of the segmentation map.
GetColorizedSegmentationWithTransparency	Colorized version of the segmentation map with transparency.
GetGroundtruthSegmentationMap	Ground truth segmentation map. By default, the ground truth segmentation map is all background.
GetHeight	Height of the source image and of the segmentation result.
GetImageMetrics	Metrics for the image. You must set the ESupervisedSegmenterResult::GroundtruthSegmentationMap before accessing this field.
GetLabel	Label at the given index.
GetLabelColor	Color of a label.
GetNumBlobs	Number of detected blobs for all labels or for the specified label.
GetNumLabels	Number of segmentation labels of the supervised segmenter that produced this result.
GetProbabilityMap	Probability map for the given label. The probability values are mapped between 0 and 255.
GetRegionForLabel	Region corresponding to the given label.
GetScore	Score used to determine whether the result contains segments that are not background.
GetSegmentationMap	Segmentation result.
GetWidth	Width of the source image and of the segmentation result.
HasForegroundSegments	Whether the predicted segmnetation contains non-background pixels.
HasGroundtruthSegmentation	Whether the result is associated with a ground truth segmentation map.
IsValid	Indicates whether the result is valid and ready to be used. A default contructed ESupervisedSegmenterResult is not valid.

operator=	Assignment operator
RemoveGroundtruthSegmentation	Removes any ground truth associated with the result.
SetClassificationThreshold	Classification threshold for determining whether the image contains blobs with a label different than background. By default, its value is given by the supervised segmenter that produced this result.
SetGroundtruthSegmentationMap	Ground truth segmentation map. By default, the ground truth segmentation map is all background.

ESupervisedSegmenterResult::GetClassificationThreshold

ESupervisedSegmenterResult::SetClassificationThreshold

Classification threshold for determining whether the image contains blobs with a label different than background.
By default, its value is given by the supervised segmenter that produced this result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetClassificationThreshold() const

void SetClassificationThreshold(float threshold)
```

ESupervisedSegmenterResult::GetColorizedSegmentation

Colorized version of the segmentation map.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageC24 GetColorizedSegmentation() const
```

ESupervisedSegmenterResult::GetColorizedSegmentationWithTransparency

Colorized version of the segmentation map with transparency.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageC24A GetColorizedSegmentationWithTransparency() const
```

ESupervisedSegmenterResult::Draw

Draws the segmentation.
The detected segment are drawn with the label color specified in the dataset used for training. The pixel classified as background are transparent.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Draw(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

-

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

ESupervisedSegmenterResult::ESupervisedSegmenterResult

Constructs a non-valid ESupervisedSegmenterResult.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void ESupervisedSegmenterResult(
 )
void ESupervisedSegmenterResult(
 const ESupervisedSegmenterResult& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterResult object that should be copied

ESupervisedSegmenterResult::GetBlobs

Detected blobs for all labels or for the specified label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterBlob> GetBlobs
(
 )
std::vector<Euresys::Open_eVision::EasyDeepLearning::ESupervisedSegmenterBlob> GetBlobs
(
 const std::string& label
 )
```

Parameters

label

Label

Remarks

A blob is a contiguous set of pixels detected to be of the same non-background label. As such, the "background" label never has blobs. The set of detected blobs depends on the threshold.

ESupervisedSegmenterResult::GetLabel

Label at the given index.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetLabel(
 int labelIndex
 )
```

Parameters

labelIndex

Index of the label

Remarks

The index must be comprised between 0 and ESupervisedSegmenterResult::NumLabels - 1. The labels are the one from the supervised segmenter that produced this result.

ESupervisedSegmenterResult::GetLabelColor

Color of a label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ERGBColor GetLabelColor(
 int i
 )

ERGBColor GetLabelColor(
 const std::string& label
 )
```

Parameters

i

Index of the label for which to get the color (between 0 and ESupervisedSegmenterResult::NumLabels - 1)

label

Label for which to get the color

Remarks

The label color is controlled at the tool level. To change a color in a result, change the label color in the tool.

ESupervisedSegmenterResult::GetNumBlobs

Number of detected blobs for all labels or for the specified label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumBlobs(
 const std::string& label
 )
int GetNumBlobs(
 )
```

Parameters

label

Label

Remarks

A blob is a contiguous set of pixels detected to be of the same non-background label. As such, the "background" label never has blobs.

ESupervisedSegmenterResult::GetProbabilityMap

Probability map for the given label.
The probability values are mapped between 0 and 255.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageBW8 GetProbabilityMap(
 const std::string& label
 )
```

Parameters

label

Label

ESupervisedSegmenterResult::GetRegionForLabel

Region corresponding to the given label.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ERegion GetRegionForLabel(
 const std::string& label
 )
```

Parameters

label

Label

ESupervisedSegmenterResult::GetGroundtruthSegmentationMap

ESupervisedSegmenterResult::SetGroundtruthSegmentationMap

Ground truth segmentation map.
By default, the ground truth segmentation map is all background.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageBW16 GetGroundtruthSegmentationMap() const
void SetGroundtruthSegmentationMap(const EImageBW16& gt)
```

ESupervisedSegmenterResult::HasForegroundSegments

Whether the predicted segmnetation contains non-background pixels.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasForegroundSegments(
 )
```

ESupervisedSegmenterResult::HasGroundtruthSegmentation

Whether the result is associated with a ground truth segmentation map.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool HasGroundtruthSegmentation(
 )
```

ESupervisedSegmenterResult::GetHeight

Height of the source image and of the segmentation result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetHeight() const
```

ESupervisedSegmenterResult::GetImageMetrics

Metrics for the image.
You must set the ESupervisedSegmenterResult::GroundtruthSegmentationMap before accessing this field.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterMetrics GetImageMetrics() const
```

ESupervisedSegmenterResult::IsValid

Indicates whether the result is valid and ready to be used.
A default contructed ESupervisedSegmenterResult is not valid.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsValid(
 )
```

ESupervisedSegmenterResult::GetNumLabels

Number of segmentation labels of the supervised segmenter that produced this result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumLabels() const
```

ESupervisedSegmenterResult::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ESupervisedSegmenterResult& operator=(
 const ESupervisedSegmenterResult& other
 )
```

Parameters

other

Reference to the ESupervisedSegmenterResult object used for the assignment

ESupervisedSegmenterResult::RemoveGroundtruthSegmentation

Removes any ground truth associated with the result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveGroundtruthSegmentation(
 )
```

ESupervisedSegmenterResult::GetScore

Score used to determine whether the result contains segments that are not background.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetScore() const
```

ESupervisedSegmenterResult::GetSegmentationMap

Segmentation result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageBW16 GetSegmentationMap() const
```

ESupervisedSegmenterResult::GetWidth

Width of the source image and of the segmentation result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetWidth() const
```

4.244. EThreeLayersImageSegmenter Class

The base class from which all the segmenters that produce three layers derive.

Remarks

The Layer Encoding can be enabled/disabled for each layer individually. The index of the layers in the coded image can also be assigned individually.

Base Class:EImageSegmenter

Derived Class(es):EGrayscaleDoubleThresholdSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetBlackLayerEncoded	Black layer encoding status.
GetBlackLayerIndex	Index of the black layer in the destination coded image.
GetNeutralLayerEncoded	Neutral layer encoding status.
GetNeutralLayerIndex	Index of the neutral layer in the destination coded image.
GetWhiteLayerEncoded	White layer encoding status.
GetWhiteLayerIndex	Index of the white layer in the destination coded image.
Load	Load the EThreeLayersImageSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the EThreeLayersImageSegmenter configuration. The given ESerializer must have been created for writing.
SetBlackLayerEncoded	Black layer encoding status.
SetBlackLayerIndex	Index of the black layer in the destination coded image.
SetNeutralLayerEncoded	Neutral layer encoding status.
SetNeutralLayerIndex	Index of the neutral layer in the destination coded image.
SetWhiteLayerEncoded	White layer encoding status.
SetWhiteLayerIndex	Index of the white layer in the destination coded image.

EThreeLayersImageSegmenter::GetBlackLayerEncoded

EThreeLayersImageSegmenter::SetBlackLayerEncoded

Black layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool GetBlackLayerEncoded() const
void SetBlackLayerEncoded(bool encode)
```

EThreeLayersImageSegmenter::GetBlackLayerIndex

EThreeLayersImageSegmenter::SetBlackLayerIndex

Index of the black layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetBlackLayerIndex() const
void SetBlackLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the black layer.

EThreeLayersImageSegmenter::Load

Load the EThreeLayersImageSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EThreeLayersImageSegmenter::GetNeutralLayerEncoded

EThreeLayersImageSegmenter::SetNeutralLayerEncoded

Neutral layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool GetNeutralLayerEncoded() const
void SetNeutralLayerEncoded(bool encode)
```

EThreeLayersImageSegmenter::GetNeutralLayerIndex

EThreeLayersImageSegmenter::SetNeutralLayerIndex

Index of the neutral layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetNeutralLayerIndex() const
void SetNeutralLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the neutral layer.

EThreeLayersImageSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool operator==(
 const EThreeLayersImageSegmenter& other
 )
```

Parameters

other

Other segmenter to compare to.

EThreeLayersImageSegmenter::Save

Save the EThreeLayersImageSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EThreeLayersImageSegmenter::GetWhiteLayerEncoded

EThreeLayersImageSegmenter::SetWhiteLayerEncoded

White layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool GetWhiteLayerEncoded() const
void SetWhiteLayerEncoded(bool encode)
```

EThreeLayersImageSegmenter::GetWhiteLayerIndex

EThreeLayersImageSegmenter::SetWhiteLayerIndex

Index of the white layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetWhiteLayerIndex() const
```

```
void SetWhiteLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the white layer.

4.245. ETwoLayersImageSegmenter Class

The base class from which all the segmenters that produce two layers derive.

Remarks

The Layer Encoding can be enabled/disabled for each layer individually. The index of the layers in the coded image can also be assigned individually.

Base Class:EImageSegmenter

Derived Class (es):
EBinaryImageSegmenter
EColorRangeThresholdSegmenter
EColorSingleThresholdSegmenter
EGrayscaleSingleThresholdSegmenterEImageRangeSegmenterEReferenceImageSegmenter

Namespace: Euresys::Open_eVision::Segmenters

Methods

GetBlackLayerEncoded	Black layer encoding status.
GetBlackLayerIndex	Index of the black layer in the destination coded image.
GetWhiteLayerEncoded	White layer encoding status.
GetWhiteLayerIndex	Index of the white layer in the destination coded image.
Load	Load the ETwoLayersImageSegmenter configuration. The given ESerializer must have been created for reading.
operator==	Comparison operator.
Save	Save the ETwoLayersImageSegmenter configuration. The given ESerializer must have been created for writing.
SetBlackLayerEncoded	Black layer encoding status.
SetBlackLayerIndex	Index of the black layer in the destination coded image.
SetWhiteLayerEncoded	White layer encoding status.
SetWhiteLayerIndex	Index of the white layer in the destination coded image.

ETwoLayersImageSegmenter::GetBlackLayerEncoded

ETwoLayersImageSegmenter::SetBlackLayerEncoded

Black layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
bool GetBlackLayerEncoded() const
void SetBlackLayerEncoded(bool encode)
```

ETwoLayersImageSegmenter::GetBlackLayerIndex

ETwoLayersImageSegmenter::SetBlackLayerIndex

Index of the black layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
OEV_UINT32 GetBlackLayerIndex() const
void SetBlackLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the black layer.

ETwoLayersImageSegmenter::Load

Load the ETwoLayersImageSegmenter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Segmenters

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ETwoLayersImageSegmenter::operator==

Comparison operator.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool operator==(
 const ETwoLayersImageSegmenter& other
 )
```

Parameters

other
-

ETwoLayersImageSegmenter::Save

Save the ETwoLayersImageSegmenter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

ETwoLayersImageSegmenter::GetWhiteLayerEncoded

ETwoLayersImageSegmenter::SetWhiteLayerEncoded

White layer encoding status.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
bool GetWhiteLayerEncoded() const
void SetWhiteLayerEncoded(bool encode)
```

ETwoLayersImageSegmenter::GetWhiteLayerIndex

ETwoLayersImageSegmenter::SetWhiteLayerIndex

Index of the white layer in the destination coded image.

Namespace: Euresys::Open_eVision::Segmenters

```
[C++]
OEV_UINT32 GetWhiteLayerIndex() const
void SetWhiteLayerIndex(OEV_UINT32 index)
```

Remarks

Setting this property automatically switches on the encoding of the white layer.

4.246. EUnsupervisedSegmenter Class

Unsupervised segmentation tool.

The unsupervised segmentation tool learns a model of what is a good product and it can be used for classiyfing whether an image is from a good or defective product and for segmenting the defects within the image.
To learn a model of what is a good product, the tool is trained by considering only the images from the good label (EUnsupervisedSegmenter::GoodLabel). The tool labels (EDeepLearningTool::GetLabel) will always be empty.
The tool can work with images of any resolution higher than EUnsupervisedSegmenter::PatchSize by merging the results obtained by applying the deep neural network using a sliding window algorithm. The overlapping between the sliding windows is controled by EUnsupervisedSegmenter::SamplingDensity.
The unsupervised segmenter offers a tradeoff between a high good detection rate and a high bad detection rate through a classification threshold that can be configured after training (EUnsupervisedSegmenter::ClassificationThreshold).

Base Class:EDeepLearningTool

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Apply	Applies the unsupervised segmenter on the given image and its mask region. We recommend to use pointer-based versions of this method to avoid unnecessary image and result copies.
EUnsupervisedSegmenter	Constructs a EUnsupervisedSegmenter object.
GetCapacity	Capacity of the EUnsupervisedSegmenter. A higher capacity makes the unsupervised segmenter capable of learning more information at the cost of a slower processing speed.
GetClassificationThreshold	Classification threshold for the score of an image (See EUnsupervisedSegmenterResult::ClassificationScore). A image with a score smaller or equal to the classification threshold will be classified as good.
GetForceGrayscale	Forces the EUnsupervisedSegmenter to convert all images to grayscale (default: true). Setting this property to false will make the underlying neural network operates with the same number of channels as the images in the dataset used for training. Otherwise, color images will be converted to grayscale before using them.
GetGoodLabel	Name of the good label in the dataset that is used for training (default value: empty).
GetInferenceScale	The effective scale applied when performing inference. If EUnsupervisedSegmenter::ScaleDisabledAtInference is true, it is equal to 1.0. Otherwise, it is equal to EUnsupervisedSegmenter::Scale.
GetNumPatchesForImage	Number of patches that will be extracted from an input image to perform inference. For EasyClassify and EasyLocate, this will always be equal to 1. For EasySegment, the number of patches will depend on the scale, patch size and sampling density parameters.
GetPatchSize	Patch size (width and height of the patches processed by the neural network).
GetSamplingDensity	Sampling density (default value: 2.0, its value must be equal or greater than 1). The sampling density is the parameter of the sliding window algorithm used for processing a whole image using subwindows of size EUnsupervisedSegmenter::PatchSize. It indicates how much overlap there will be between the image patches: the stride between two consecutive patches is EUnsupervisedSegmenter::PatchSize / EUnsupervisedSegmenter::SamplingDensity.
GetScale	Down-scaling applied to images before processing them. Its value is between 0 and 1 (default value: 1).

GetScaleDisabledAtInference	Whether to apply the scale parameter at inference or not. If you disable the scale at inference, make sure your images are already scaled. For example, if you trained with images of 1024x1024 and a scale of 0.5, the inference images should be 512 x 512 images with the same field of view as the training images.
GetToolType	Type of the deep learning tool.
GetTrainingMetrics	Training metrics at the given iteration
GetValidationMetrics	Validation metrics at the given iteration
operator=	Assignment operator
SerializeSettings	Serializes the settings of the unsupervised segmenter.
SetCapacity	Capacity of the EUnsupervisedSegmenter. A higher capacity makes the unsupervised segmenter capable of learning more information at the cost of a slower processing speed.
SetClassificationThreshold	Classification threshold for the score of an image (See EUnsupervisedSegmenterResult::ClassificationScore). A image with a score smaller or equal to the classification threshold will be classified as good.
SetForceGrayscale	Forces the EUnsupervisedSegmenter to convert all images to grayscale (default: true). Setting this property to false will make the underlying neural network operates with the same number of channels as the images in the dataset used for training. Otherwise, color images will be converted to grayscale before using them.
SetGoodLabel	Name of the good label in the dataset that is used for training (default value: empty).
SetPatchSize	Patch size (width and height of the patches processed by the neural network).
SetSamplingDensity	Sampling density (default value: 2.0, its value must be equal or greater than 1). The sampling density is the parameter of the sliding window algorithm used for processing a whole image using subwindows of size EUnsupervisedSegmenter::PatchSize. It indicates how much overlap there will be between the image patches: the stride between two consecutive patches is EUnsupervisedSegmenter::PatchSize / EUnsupervisedSegmenter::SamplingDensity.
SetScale	Down-scaling applied to images before processing them. Its value is between 0 and 1 (default value: 1).
SetScaleDisabledAtInference	Whether to apply the scale parameter at inference or not. If you disable the scale at inference, make sure your images are already scaled. For example, if you trained with images of 1024x1024 and a scale of 0.5, the inference images should be 512 x 512 images with the same field of view as the training images.

EUnsupervisedSegmenter::Apply

Applies the unsupervised segmenter on the given image and its mask region. We recommend to use pointer-based versions of this method to avoid unnecessary image and result copies.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EUnsupervisedSegmenterResult Apply(
 const EBaseROI& img
 )

EUnsupervisedSegmenterResult Apply(
 const EBaseROI& img,
 const ERegion& mask
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterResult>
Apply(
 const std::vector<Euresys::Open_eVision::EImageBW8>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterResult>
Apply(
 const std::vector<Euresys::Open_eVision::EImageBW8>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterResult>
Apply(
 const std::vector<Euresys::Open_eVision::EImageBW16>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterResult>
Apply(
 const std::vector<Euresys::Open_eVision::EImageBW16>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterResult>
Apply(
 const std::vector<Euresys::Open_eVision::EImageC24>& imgs
 )

std::vector<Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterResult>
Apply(
 const std::vector<Euresys::Open_eVision::EImageC24>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion>& masks
 )

void Apply(
 const std::vector<Euresys::Open_eVision::EBaseROI*>& imgs,
 const std::vector<Euresys::Open_
eVision::EasyDeepLearning::EUnsupervisedSegmenterResult*>& results
 )
```

```
void Apply(
 const std::vector<Euresys::Open_eVision::EBaseROI*>& imgs,
 const std::vector<Euresys::Open_eVision::ERegion*>& masks,
 const std::vector<Euresys::Open_
eVision::EasyDeepLearning::EUnsupervisedSegmenterResult*>& results
 )
```

Parameters

img
Image to classify and segment defects in.

mask
Mask region of the image.

imgs
Vector of image to classify and segment defects in.

masks
Vector of mask regions for the images.

results
-

EUnsupervisedSegmenter::GetCapacity

EUnsupervisedSegmenter::SetCapacity

Capacity of the EUnsupervisedSegmenter.
A higher capacity makes the unsupervised segmenter capable of learning more information at the cost of a slower processing speed.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
Euresys::Open_eVision::EasyDeepLearning::EUnsupervisedSegmenterCapacity GetCapacity()
const

void SetCapacity(Euresys::Open_
eVision::EasyDeepLearning::EUnsupervisedSegmenterCapacity capacity)
```

EUnsupervisedSegmenter::GetClassificationThreshold

EUnsupervisedSegmenter::SetClassificationThreshold

Classification threshold for the score of an image (See EUnsupervisedSegmenterResult::ClassificationScore).
A image with a score smaller or equal to the classification threshold will be classified as good.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetClassificationThreshold() const
void SetClassificationThreshold(float threshold)
```

Remarks

The classification threshold is optmized during training to maximize the balanced accuracy of the unsupervised segmenter.
The classification threshold can be changed after training.

EUnsupervisedSegmenter::EUnsupervisedSegmenter

Constructs a EUnsupervisedSegmenter object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EUnsupervisedSegmenter(
 )
void EUnsupervisedSegmenter(
 const EUnsupervisedSegmenter& other
 )
```

Parameters

other

Reference to the EUnsupervisedSegmenter object that should be copied

EUnsupervisedSegmenter::GetForceGrayscale

EUnsupervisedSegmenter::SetForceGrayscale

Forces the EUnsupervisedSegmenter to convert all images to grayscale (default: true).
Setting this property to false will make the underlying neural network operates with the same number of channels as the images in the dataset used for training. Otherwise, color images will be converted to grayscale before using them.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetForceGrayscale() const
void SetForceGrayscale(bool forceGrayscale)
```

EUnsupervisedSegmenter::GetNumPatchesForImage

Number of patches that will be extracted from an input image to perform inference.
For EasyClassify and EasyLocate, this will always be equal to 1.
For EasySegment, the number of patches will depend on the scale, patch size and sampling density parameters.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetNumPatchesForImage(
 int imageWidth,
 int imageHeight
 )
```

Parameters

imageWidth

Width of the image for which to get the number of patch

imageHeight

Height of the image for which to get the number of patch

EUnsupervisedSegmenter::GetTrainingMetrics

Training metrics at the given iteration

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EUnsupervisedSegmenterMetrics GetTrainingMetrics(
 int iteration
 )
```

Parameters

iteration

Iteration at which to get the metrics

Remarks

At a given iteration, the metrics will always contain the error (see EUnsupervisedSegmenterMetrics::Error). Other metrics (scores, accuracy, etc.) will be available only at iterations where the validation error is a new minimum.

EUnsupervisedSegmenter::GetValidationMetrics

Validation metrics at the given iteration

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EUnsupervisedSegmenterMetrics GetValidationMetrics(
 int iteration
 )
```

Parameters

iteration

Iteration at which to get the metrics

Remarks

At a given iteration, the metrics will always contain the error (see EUnsupervisedSegmenterMetrics::Error). Other metrics (scores, accuracy, etc.) will be available only at iterations where the validation error is a new minimum.

EUnsupervisedSegmenter::GetGoodLabel

EUnsupervisedSegmenter::SetGoodLabel

Name of the good label in the dataset that is used for training (default value: empty).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
std::string GetGoodLabel() const
void SetGoodLabel(const std::string& label)
```

EUnsupervisedSegmenter::GetInferenceScale

The effective scale applied when performing inference.
If EUnsupervisedSegmenter::ScaleDisabledAtInference is true, it is equal to 1.0. Otherwise, it is equal to EUnsupervisedSegmenter::Scale.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetInferenceScale() const
```

EUnsupervisedSegmenter::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EUnsupervisedSegmenter& operator=(
 const EUnsupervisedSegmenter& other
 )
```

Parameters

other

Reference to the EUnsupervisedSegmenter object that should be copied

EUnsupervisedSegmenter::GetPatchSize

EUnsupervisedSegmenter::SetPatchSize

Patch size (width and height of the patches processed by the neural network).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int GetPatchSize() const

void SetPatchSize(int patchSize)
```

Remarks

There are three supported patch size: 64x64, 128x128, and 256x256. By default, the patch size is 0 and it means that the patch size will be 128x128 if all images in the training and validation dataset have a higher resolution or the patch size will be 64x64.

EUnsupervisedSegmenter::GetSamplingDensity

EUnsupervisedSegmenter::SetSamplingDensity

Sampling density (default value: 2.0, its value must be equal or greater than 1).
The sampling density is the parameter of the sliding window algorithm used for processing a whole image using subwindows of size EUnsupervisedSegmenter::PatchSize. It indicates how much overlap there will be between the image patches: the stride between two consecutive patches is EUnsupervisedSegmenter::PatchSize / EUnsupervisedSegmenter::SamplingDensity.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetSamplingDensity() const

void SetSamplingDensity(float density)
```

EUnsupervisedSegmenter::GetScale

EUnsupervisedSegmenter::SetScale

Down-scaling applied to images before processing them. Its value is between 0 and 1 (default value: 1).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetScale() const

void SetScale(float scale)
```

EUnsupervisedSegmenter::GetScaleDisabledAtInference

EUnsupervisedSegmenter::SetScaleDisabledAtInference

Whether to apply the scale parameter at inference or not.
If you disable the scale at inference, make sure your images are already scaled. For example, if you trained with images of 1024x1024 and a scale of 0.5, the inference images should be 512 x 512 images with the same field of view as the training images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool GetScaleDisabledAtInference() const

void SetScaleDisabledAtInference(bool disable)
```

EUnsupervisedSegmenter::SerializeSettings

Serializes the settings of the unsupervised segmenter.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void SerializeSettings(
 ESerializer* serializer
 )
```

Parameters

serializer

Pointer to ESerializer

EUnsupervisedSegmenter::GetToolType

Type of the deep learning tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
Euresys::Open_eVision::EasyDeepLearning::EDeepLearningToolType GetToolType() const
```

4.247. EUnsupervisedSegmenterMetrics Class

Collection of metrics used to evaluate the state of an EUnsupervisedSegmenter.
A metric is a value summarizing the quality of a collection of unsupervised segmentation results (see EUnsupervisedSegmenterResult) with respect to their ground truth.
New results can be added to the object individually with EUnsupervisedSegmenterMetrics::AddResult or collectively with EUnsupervisedSegmenterMetrics::AddMetrics.

EUnsupervisedSegmenterMetrics contains two types of metrics: unsupervised metrics that are computed only on good images and supervised/defect detection metrics that are computed on both good and bad images. The defect detection metrics are accessible only when results for bad images were added to the object. When supervised metrics are accessible, EUnsupervisedSegmenterMetrics::IsTotallyUnsupervised is false.
There is only one unsupervised metric: the error (see EUnsupervisedSegmenterMetrics::Error).
See EDeepLearningDefectDetectionMetrics for a description of the defect detection metrics.

Base Class:EDeepLearningDefectDetectionMetrics

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

AddErrorResult	Adds the given result to the metric for error computation. The result must be computed from a good image that was corrupted during training.
AddMetrics	Adds the other metrics to the current metrics of this object.
AddResult	Adds the given result with the corresponding ground truth label to the metrics.
EUnsupervisedSegmenterMetrics	Constructs an EUnsupervisedSegmenterMetrics object.
GetAverageScoreOnDefectiveImages	Gets the average score of defective images.
GetAverageScoreOnGoodImages	Gets the average score of good images.

GetBestWeightedAccuracy	Best achievable weighted accuracy. The weighted accuracy is the weighted average of the true positive rate and the true negative rate (which is equal to 1 minus the false positive rate). See EROCPoint. The classification threshold corresponding to this accuracy is given by EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracyClassificationThreshold.
GetBestWeightedAccuracyClassificationThreshold	Classification threshold giving the best achievable weighted accuracy (see EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracy).
GetError	The error of the segmenter. This metric is only available in the metrics computed during a training which are accessible through EUnsupervisedSegmenter::GetValidationMetrics. The error, which is also called the loss, is the quantity that is minimized during the training of the deep neural network.
IsTotallyUnsupervised	Whether this metrics has results from only good images (true) or from both good and defective images (false). Some metrics are accessible only if EUnsupervisedSegmenterMetrics::IsTotallyUnsupervised is false.
IsValid	Indicates whether the object contains at least one result.
Load	Loads an unsupervised segmentation metric. The given ESerializer must have been created for reading.
operator=	Assignment operator.
RemoveErrorResult	Removes the given result from the metric for error computation. The result must be computed from a good image that was corrupted during training.
RemoveResult	Removes the given result with the corresponding ground truth label to the metrics.
Save	Saves an unsupervised segmentation metric. The given ESerializer must have been created for writing.

EUnsupervisedSegmenterMetrics::AddErrorResult

Adds the given result to the metric for error computation. The result must be computed from a good image that was corrupted during training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void AddErrorResult(
 const EUnsupervisedSegmenterResult& result
 )
```

Parameters

result

A reference to an EUnsupervisedSegmenterResult object.

EUnsupervisedSegmenterMetrics::AddMetrics

Adds the other metrics to the current metrics of this object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddMetrics(
 const EUnsupervisedSegmenterMetrics& other
 )
```

Parameters

other

Unsupervised segmenter metrics

EUnsupervisedSegmenterMetrics::AddResult

Adds the given result with the corresponding ground truth label to the metrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void AddResult(
 const EUnsupervisedSegmenterResult& result,
 bool isGoodImage
 )
```

Parameters

result

A reference to an EUnsupervisedSegmenterResult object.

isGoodImage

True if the ground truth of this result is good, else false.

EUnsupervisedSegmenterMetrics::GetAverageScoreOnDefectiveImages

Gets the average score of defective images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetAverageScoreOnDefectiveImages() const
```

EUnsupervisedSegmenterMetrics::GetAverageScoreOnGoodImages

Gets the average score of good images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetAverageScoreOnGoodImages() const
```

EUnsupervisedSegmenterMetrics::GetError

The error of the segmenter.
This metric is only available in the metrics computed during a training which are accessible through EUnsupervisedSegmenter::GetValidationMetrics.
The error, which is also called the loss, is the quantity that is minimized during the training of the deep neural network.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetError() const
```

EUnsupervisedSegmenterMetrics::EUnsupervisedSegmenterMetrics

Constructs an EUnsupervisedSegmenterMetrics object.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EUnsupervisedSegmenterMetrics(
 )

void EUnsupervisedSegmenterMetrics(
 const EUnsupervisedSegmenterMetrics& other
 )
```

Parameters

other
Reference to the EUnsupervisedSegmenterMetrics object that should be copied

EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracy

Best achievable weighted accuracy.
The weighted accuracy is the weighted average of the true positive rate and the true negative rate (which is equal to 1 minus the false positive rate). See EROCPoint.
The classification threshold corresponding to this accuracy is given by EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracyClassificationThreshold.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedAccuracy(
 float goodWeight,
 float badWeight
 )

float GetBestWeightedAccuracy(
 const EClassificationDataset& dataset
 )
```

Parameters

goodWeight

-

badWeight

-

dataset

Dataset to get the label weight from.

Remarks

When using a dataset as the source for the label weights, the good weight is the weight of the "good" label and the bad weight is the sum of the weights of all the other labels.

EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracyClassificationThreshold

Classification threshold giving the best achievable weighted accuracy (see EUnsupervisedSegmenterMetrics::GetBestWeightedAccuracy).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetBestWeightedAccuracyClassificationThreshold(
 float goodWeight,
 float badWeight
 )

float GetBestWeightedAccuracyClassificationThreshold(
 const EClassificationDataset& dataset
 )
```

Parameters

goodWeight
Weight for the good label

badWeight
Weight for the bad label

dataset
Dataset to get the label weight from.

EUnsupervisedSegmenterMetrics::IsTotallyUnsupervised

Whether this metrics has results from only good images (true) or from both good and defective images (false).
Some metrics are accessible only if EUnsupervisedSegmenterMetrics::IsTotallyUnsupervised is false.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsTotallyUnsupervised(
 )
```

EUnsupervisedSegmenterMetrics::IsValid

Indicates whether the object contains at least one result.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
bool IsValid(
 )
```

EUnsupervisedSegmenterMetrics::Load

Loads an unsupervised segmentation metric. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& path
 )
```

```
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EUnsupervisedSegmenterMetrics::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EUnsupervisedSegmenterMetrics& operator=(
 const EUnsupervisedSegmenterMetrics& other
 )
```

Parameters

other
Reference to the EUnsupervisedSegmenterMetrics object used for the assignment

EUnsupervisedSegmenterMetrics::RemoveErrorResult

Removes the given result from the metric for error computation. The result must be computed from a good image that was corrupted during training.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveErrorResult(
 const EUnsupervisedSegmenterResult& result
 )
```

Parameters

result
A reference to an EUnsupervisedSegmenterResult object.

EUnsupervisedSegmenterMetrics::RemoveResult

Removes the given result with the corresponding ground truth label to the metrics.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void RemoveResult(
 const EUnsupervisedSegmenterResult& result,
 bool isGoodImage
 )
```

Parameters

result

A reference to an EUnsupervisedSegmenterResult object.

isGoodImage

True if the ground truth of this result is good, else false.

EUnsupervisedSegmenterMetrics::Save

Saves an unsupervised segmentation metric. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

4.248. EUnsupervisedSegmenterResult Class

An EUnsupervisedSegmenterResult object represents the result of a EUnsupervisedSegmenter tool.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Methods

Draw	Draws the segmentation (with transparency). To indicate the importance of the defect at a given pixel, the segmentation is drawn using a gradient going from yellow (least important) to red (maximum importance) (See EUnsupervisedSegmenterResult::SegmentationMap).
EUnsupervisedSegmenterResult	Constructs a non-valid EUnsupervisedSegmenterResult.
GetClassificationScore	Score that is used for classification of the image.
GetError	Error of the image.
GetRegion	Returns the segmented region. The segmented region corresponds to the pixels that have a value strictly higher than 0 in the segmentation map (see EUnsupervisedSegmenterResult::SegmentationMap).
GetSegmentationMap	Returns the segmentation map. The segmentation map is a grayscale image where all defective pixels have a value strictly higher than 0. The value of a pixel is proportional to the importance of the defect at that position.
IsComplete	Indicates whether the result is complete and ready to be used.
IsDefective	Indicates whether the result is defective/not good based on the threshold of the unsupervised segmentation tool that produced the result (see EUnsupervisedSegmenter::ClassificationThreshold).
IsGood	Indicates whether the result is good based on the threshold of the unsupervised segmentation tool that produced the result (see EUnsupervisedSegmenter::ClassificationThreshold).
IsValid	Indicates whether the result was produced by a EUnsupervisedSegmenter object. A default contructed EUnsupervisedSegmenterResult is not valid.
operator=	Assignment operator

EUnsupervisedSegmenterResult::GetClassificationScore

Score that is used for classification of the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float GetClassificationScore() const
```

Remarks

The classification score is the value which is compared to the classification threshold of the EUnsupervisedSegmenter to decide whether the corresponding image is good or defective.

EUnsupervisedSegmenterResult::Draw

Draws the segmentation (with transparency).
To indicate the importance of the defect at a given pixel, the segmentation is drawn using a gradient going from yellow (least important) to red (maximum importance) (See EUnsupervisedSegmenterResult::SegmentationMap).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Draw(
 EDrawAdapter* graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicsContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

Parameters

graphicsContext

-

zoomX

Horizontal zooming factor. A value greater than 1 means zoom in. By default, true scale is used.

zoomY

Vertical zooming factor. A value greater than 1 means zoom in. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EUnsupervisedSegmenterResult::GetError

Error of the image.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float GetError() const
```

Remarks

The error is the quantity that is minimized on good images during training.

EUnsupervisedSegmenterResult::EUnsupervisedSegmenterResult

Constructs a non-valid EUnsupervisedSegmenterResult.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
void EUnsupervisedSegmenterResult(
 )
void EUnsupervisedSegmenterResult(
 const EUnsupervisedSegmenterResult& other
 )
```

Parameters

other

Reference to the EUnsupervisedSegmenterResult object that should be copied

EUnsupervisedSegmenterResult::IsComplete

Indicates whether the result is complete and ready to be used.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsComplete(
 )
```

EUnsupervisedSegmenterResult::IsDefective

Indicates whether the result is defective/not good based on the threshold of the unsupervised segmentation tool that produced the result (see EUnsupervisedSegmenter::ClassificationThreshold).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsDefective(
 )
```

EUnsupervisedSegmenterResult::IsGood

Indicates whether the result is good based on the threshold of the unsupervised segmentation tool that produced the result (see EUnsupervisedSegmenter::ClassificationThreshold).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsGood(
 )
```

EUnsupervisedSegmenterResult::IsValid

Indicates whether the result was produced by a EUnsupervisedSegmenter object.
A default contructed EUnsupervisedSegmenterResult is not valid.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
bool IsValid(
 )
```

EUnsupervisedSegmenterResult::operator=

Assignment operator

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
EUnsupervisedSegmenterResult& operator=(
 const EUnsupervisedSegmenterResult& other
 )
```

Parameters

other

Reference to the EUnsupervisedSegmenterResult object used for the assignment

EUnsupervisedSegmenterResult::GetRegion

Returns the segmented region. The segmented region corresponds to the pixels that have a value strictly higher than 0 in the segmentation map (see EUnsupervisedSegmenterResult::SegmentationMap).

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
ERegion GetRegion() const
```

EUnsupervisedSegmenterResult::GetSegmentationMap

Returns the segmentation map. The segmentation map is a grayscale image where all defective pixels have a value strictly higher than 0. The value of a pixel is proportional to the importance of the defect at that position.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
EImageBW8 GetSegmentationMap() const
```

4.249. EUnwarpingLut Class

This class is used only as a lookup table in the EWorldShape::Unwarp and EWorldShape::SetupUnwarp methods. It has no other use of its own.

Namespace: Euresys::Open_eVision

Methods

EUnwarpingLut	Constructs a EUnwarpingLut object that is used to speed-up the unwarping process.

EUnwarpingLut::EUnwarpingLut

Constructs a EUnwarpingLut object that is used to speed-up the unwarping process.

Namespace: Euresys::Open_eVision

[C++]

```
void EUnwarpingLut(
 )
```

4.250. EUtils Class

3D Utilitarian Functions.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Copy	Copies a source map or a constant in a destination map.

EUtils::Copy

Copies a source map or a constant in a destination map.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Copy(
 const EZMap8& sourceImage,
 EZMap8& destinationImage
 )

void Copy(
 const EZMap16& sourceImage,
 EZMap16& destinationImage
 )

void Copy(
 const EZMap32f& sourceImage,
 EZMap32f& destinationImage
 )

void Copy(
 const EDepthMap8& sourceImage,
 EDepthMap8& destinationImage
 )

void Copy(
 const EDepthMap16& sourceImage,
 EDepthMap16& destinationImage
 )

void Copy(
 const EDepthMap32f& sourceImage,
 EDepthMap32f& destinationImage
 )

void Copy(
 const EZMap8& sourceImage,
 const ERegion& region,
 EZMap8& destinationImage
 )

void Copy(
 const EZMap16& sourceImage,
 const ERegion& region,
 EZMap16& destinationImage
 )
```

```
void Copy(
 const EZMap32f& sourceImage,
 const ERegion& region,
 EZMap32f& destinationImage
 )

void Copy(
 const EDepthMap8& sourceImage,
 const ERegion& region,
 EDepthMap8& destinationImage
 )

void Copy(
 const EDepthMap16& sourceImage,
 const ERegion& region,
 EDepthMap16& destinationImage
 )

void Copy(
 const EDepthMap32f& sourceImage,
 const ERegion& region,
 EDepthMap32f& destinationImage
 )

void Copy(
 EDepth8 constant,
 EZMap8& destinationImage
 )

void Copy(
 EDepth16 constant,
 EZMap16& destinationImage
 )

void Copy(
 EDepth32f constant,
 EZMap32f& destinationImage
 )

void Copy(
 EDepth8 constant,
 EDepthMap8& destinationImage
 )

void Copy(
 EDepth16 constant,
 EDepthMap16& destinationImage
 )

void Copy(
 EDepth32f constant,
 EDepthMap32f& destinationImage
 )
```

```
void Copy(
 EDepth8 constant,
 const ERegion& region,
 EZMap8& destinationImage
 )

void Copy(
 EDepth16 constant,
 const ERegion& region,
 EZMap16& destinationImage
 )

void Copy(
 EDepth32f constant,
 const ERegion& region,
 EZMap32f& destinationImage
 )

void Copy(
 EDepth8 constant,
 const ERegion& region,
 EDepthMap8& destinationImage
 )

void Copy(
 EDepth16 constant,
 const ERegion& region,
 EDepthMap16& destinationImage
 )

void Copy(
 EDepth32f constant,
 const ERegion& region,
 EDepthMap32f& destinationImage
 )
```

Parameters

sourceImage
: Source map.

destinationImage
: Destination map.

region
: Region on which to copy.

constant
: Depth constant.

4.251. EVector Class

Base class for all typed vectors.

Remarks

This class contains all methods that are not type specific. Mainly methods to handle elements count and serialization

Derived Class (es):
EBW16PathVector
EBW16Vector
EBW32Vector
EBW8PathVector
EBW8Vector
EBWHistogramVectorEC24PathVectorEC24VectorEColorVectorEPathVectorEPeakVector

Namespace: Euresys::Open_eVision

Methods

Empty	Resets the number of elements to 0.
GetNumElements	Number of elements in the vector.
Load	Loads the EVector object with all its attributes.
RemoveElement	Removes the element at the specified position
Save	Saves the EVector object with all its attributes.
SetNumElements	Number of elements in the vector.

EVector::Empty

Resets the number of elements to 0.

Namespace: Euresys::Open_eVision

[C++]

```
void Empty(
 )
```

EVector::Load

Loads the EVector object with all its attributes.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& file,
 OEV_UINT32 un32Version
 )
```

```
void Load(
 ESerializer* serializer,
 OEV_UINT32 un32Version
 )
```

Parameters

file
File path.

un32Version
The file version number.

serializer
Serializer. Must be in read mode.

EVector::GetNumElements

EVector::SetNumElements

Number of elements in the vector.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumElements() const
void SetNumElements(OEV_UINT32 un32NumElements)
```

EVector::RemoveElement

Removes the element at the specified position

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveElement(
 OEV_UINT32 index
 )
```

Parameters

index
Index, between 0 and EVector::NumElements (excluded) of the element to be accessed.

EVector::Save

Saves the EVector object with all its attributes.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& file,
 OEV_UINT32 un32Version
 )
void Save(
 ESerializer* serializer,
 OEV_UINT32 un32Version
 )
```

Parameters

file

File path.

un32Version

The file version number.

serializer

Serializer. Must be in write mode.

4.252. EVectorModel Class

Represents a EShape hierarchy with a single root. This hierarchy can be retrieved from files in different file formats.

Namespace: Euresys::Open_eVision

Methods

DisableDrawArea	Disables the custom drawing of the EVectorModel with a specific zoom and pan.
Draw	Draws a graphical representation of the hierarchy.
DrawWithCurrentPen	-
EnableDrawArea	Enables the drawing of the EVectorModel with a specific zoom and pan. Default: false
EVectorModel	Constructs an EVectorModel context.
GetCenter	Gets the center of the model.
GetRoot	Gets the root of the hierarchy.
GetScale	Gets the scale of the model.
GetVectorModelExtremas	Retrieves the coordinate extremas of the vector model
Load	Loads a Shape.The given ESerializer must have been created for reading.

LoadDXF	Loads an EVectorModel from an ascii DXF file. We do not support the whole possibilities offered by the dxf format. We are currently limited to straight polylines, lines, arcs and points. Blocks are ignored at the moment. Angles are assumed to be given in degrees.
operator=	Assignment operator, copies another EVectorModel instance to this one.
Save	Loads a Shape.The given ESerializer must have been created for writing.
SetCenter	Gets the center of the model.
SetScale	Gets the scale of the model.

EVectorModel::GetCenter

EVectorModel::SetCenter

Gets the center of the model.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter()

void SetCenter(const EPoint& center)
```

EVectorModel::DisableDrawArea

Disables the custom drawing of the EVectorModel with a specific zoom and pan.

Namespace: Euresys::Open_eVision

[C++]

```
void DisableDrawArea(
 )
```

EVectorModel::Draw

Draws a graphical representation of the hierarchy.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext
 )
void Draw(
 HDC graphicContext
 )
void Draw(
 HDC graphicContext,
 const ERGBColor& color
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EVectorModel::DrawWithCurrentPen

This method is deprecated.

-

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext
 )
```

Parameters

graphicContext

-

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EVectorModel::EnableDrawArea

Enables the drawing of the EVectorModel with a specific zoom and pan. Default: false

Namespace: Euresys::Open_eVision

```
[C++]
void EnableDrawArea(
 int drawWindowWidth,
 int drawWindowHeight
 )
```

Parameters

drawWindowWidth

The width of the window that the EVectorModel is drawn into

drawWindowHeight

The height of the window that the EVectorModel is drawn into

EVectorModel::EVectorModel

Constructs an EVectorModel context.

Namespace: Euresys::Open_eVision

```
[C++]
void EVectorModel(
 )

void EVectorModel(
 const EVectorModel& frameShape
 )
```

Parameters

frameShape

-

EVectorModel::GetVectorModelExtremas

Retrieves the coordinate extremas of the vector model

Namespace: Euresys::Open_eVision

```
[C++]
void GetVectorModelExtremas(
 float& xmin,
 float& ymin,
 float& xmax,
 float& ymax,
 bool inSensorCoordinates
 )
```

Parameters

xmin
The X minimum of EVectorModel

ymin
The Y minimum of EVectorModel

xmax
The X maximum of EVectorModel

ymax
The Y maximum of EVectorModel

inSensorCoordinates
Specifies wheather the extremas are retrieved in sensor coordinates. Default: false

EVectorModel::Load

Loads a Shape.The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& filePath
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

filePath
The file path.

serializer
Pointer to the ESerializer created for reading.

EVectorModel::LoadDXF

Loads an EVectorModel from an ascii DXF file. We do not support the whole possibilities offered by the dxf format. We are currently limited to straight polylines, lines, arcs and points. Blocks are ignored at the moment. Angles are assumed to be given in degrees.

Namespace: Euresys::Open_eVision

[C++]

```
void LoadDXF(
 const std::string& fileName,
 float scale
 )
```

```
void LoadDXF(
 const std::string& fileName,
 float scale,
 const EPoint& origin
 )
```

Parameters

fileName

The name of the dxf file.

scale

The scale of the vector model, defaults to 1.

origin

The origin of the EVectorModel in the dxf's coordinates. If not specified, we try to parse the origin from the UCSORG field in the dxf headers. Otherwise, origin is (0, 0).

EVectorModel::operator=

Assignment operator, copies another EVectorModel instance to this one.

Namespace: Euresys::Open_eVision

[C++]

```
EVectorModel& operator=(
 const EVectorModel& other
 )
```

Parameters

other

The EVectorModel instance to copy from.

EVectorModel::GetRoot

Gets the root of the hierarchy.

Namespace: Euresys::Open_eVision

[C++]

```
EFrameShape& GetRoot()
```

EVectorModel::Save

Loads a Shape.The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 ESerializer* serializer
 )
void Save(
 const std::string& filePath
 )
```

Parameters

serializer

Pointer to the ESerializer created for writing.

filePath

The file path.

EVectorModel::GetScale

EVectorModel::SetScale

Gets the scale of the model.

Namespace: Euresys::Open_eVision

[C++]

```
float GetScale()
void SetScale(float scale)
```

4.253. EWedge Class

Represents a model of a wedge (disk, ring, sector or curvilinear quadrilateral).

Base Class:EFrame

Namespace: Euresys::Open_eVision

Methods

CopyTo	Copies all the data of the current EWedge object into another EWedge object and returns it.
EWedge	Constructs a EWedge object.
GetAmplitude	Angular amplitude of the EWedge.
GetApexAngle	Angular position at the apex of the EWedge.
GetBreadth	Breadth of the EWedge.
GetCorners	Retrieves the coordinates of each corner of the EWedge.

GetDirect	Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.
GetEdges	Retrieves each edge of the EWedge.
GetEndAngle	Ending angular position of the EWedge.
GetFullBreadth	Flag indicating if the EWedge has a full breadth (breadth = radius).
GetFullCircle	Flag indicating if the EWedge is a full circle (amplitude = 2 PI).
GetInnerApex	Inner apex point coordinates of the EWedge.
GetInnerArcLength	Inner circle arc length of the EWedge.
GetInnerDiameter	Inner diameter of the EWedge.
GetInnerEnd	Inner end point coordinates of the EWedge.
GetInnerOrg	Inner origin point coordinates of the EWedge.
GetInnerPoint	Returns the coordinates of a particular point specified by its location along the inner circle arc.
GetInnerRadius	Inner radius of the EWedge
GetMidEdges	Retrieves the center coordinates of each edge of the wedge fitting gauge.
GetOrgAngle	Angular position from where the EWedge extents.
GetOuterApex	Outer apex point coordinates of the EWedge.
GetOuterArcLength	Outer circle arc length of the EWedge.
GetOuterDiameter	Outer diameter of the EWedge.
GetOuterEnd	Outer end point coordinates of the EWedge.
GetOuterOrg	Outer origin point coordinates of the EWedge.
GetOuterPoint	Returns the coordinates of a particular point specified by its location along the outter circle arc.
GetOuterRadius	Outer radius of the EWedge.
GetPoint	Returns the coordinates of a particular point specified by its location along the wedge perimeter.
operator=	Copies all the data from another EWedge object into the current EWedge object
SetAmplitude	Angular amplitude of the EWedge.
SetDiameters	Sets the nominal inner/outer diameter and breadth of the EWedge.
SetFromCenterAndOrigin	Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.
SetFromOriginMiddleEnd	Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.
SetFromTwoPoints	DEPRECATED (you should use EWedge::SetFromCenterAndOrigin): Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.
SetRadii	Sets the nominal radius and breadth of the EWedge.

EWedge::GetAmplitude

EWedge::SetAmplitude

Angular amplitude of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAmplitude() const
void SetAmplitude(float ampl)
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge::GetApexAngle

Angular position at the apex of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetApexAngle() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge::GetBreadth

Breadth of the EWedge.

Namespace: Euresys::Open_eVision

```
[C++]
float GetBreadth() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::CopyTo

Copies all the data of the current EWedge object into another EWedge object and returns it.

Namespace: Euresys::Open_eVision

```
[C++]
void CopyTo(
 EWedge& other
 )

EWedge* CopyTo(
 EWedge* destinationImage
 )
```

Parameters

other

-

destinationImage

Pointer to the EWedge object in which the current EWedge object data have to be copied.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EWedge object will be created and returned.

EWedge::GetDirect

EWedge::SetDirect

Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetDirect() const
void SetDirect(bool direct)
```

Remarks

true (default) means that angles increase anticlockwisely in a direct coordinate system, and clockwisely in an inverse coordinate system.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards.
* When the field of view is not calibrated, the coordinate system is said to be inverse the abscissa extends rightwards and the ordinate extends downwards.

EWedge::GetEndAngle

Ending angular position of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEndAngle() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge::EWedge

Constructs a EWedge object.

Namespace: Euresys::Open_eVision

[C++]

```
void EWedge(
 )
```

```
void EWedge(
 const EPoint& center,
 float diameter,
 float breadth,
 float originAngle,
 bool direct
 )

void EWedge(
 const EPoint& center,
 const EPoint& origin,
 float breadth,
 bool direct
 )

void EWedge(
 const EPoint& center,
 float diameter,
 float breadth,
 float originAngle,
 float amplitude
 )

void EWedge(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 float breadth,
 bool fullCircle
 )

void EWedge(
 const EWedge& other
 )
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is (0,0).

diameter

Nominal diameter of the wedge. The default value is 100.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

originAngle

Origin point coordinates of the wedge.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

origin

Origin point coordinates of the wedge.

amplitude

Nominal angular amplitude of the wedge. The default value is 360.

middle
Middle point coordinates of the wedge.
end
End point coordinates of the wedge.
fullCircle
true (default) in case of a full turn wedge. If fullCircle is false, origin and end give the wedge's amplitude.
other
Another EWedge object to be copied in the new EWedge object.

EWedge::GetFullBreadth

Flag indicating if the EWedge has a full breadth (breadth = radius).

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFullBreadth() const
```

EWedge::GetFullCircle

Flag indicating if the EWedge is a full circle (amplitude = 2 PI).

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFullCircle() const
```

EWedge::GetCorners

Retrieves the coordinates of each corner of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
void GetCorners(
 EPoint& ar,
 EPoint& AAr,
 EPoint& aRR,
 EPoint& AARR
 )
```

Parameters

ar
Coordinates of the inner org corner of the EWedge.

AAr
Coordinates of the inner end corner of the EWedge.

aRR
Coordinates of the outer org corner of the EWedge.

AARR
Coordinates of the outer end corner of the EWedge.

EWedge::GetEdges

Retrieves each edge of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
void GetEdges(
 ELine& a,
 ELine& AA,
 ECircle& r,
 ECircle& RR
 )
```

Parameters

a
Org edge of the EWedge.

AA
End edge of the EWedge.

r
Inner edge of the EWedge.

RR
Outer edge of the EWedge.

EWedge::GetInnerPoint

Returns the coordinates of a particular point specified by its location along the inner circle arc.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetInnerPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range [-1, +1]).

EWedge::GetMidEdges

Retrieves the center coordinates of each edge of the wedge fitting gauge.

Namespace: Euresys::Open_eVision

```
[C++]
void GetMidEdges(
 EPoint& a,
 EPoint& AA,
 EPoint& r,
 EPoint& RR
 )
```

Parameters

a

Center coordinates of the org edge of the EWedge.

AA

Center coordinates of the end edge of the EWedge.

r

Center coordinates of the inner edge of the EWedge.

RR

Center coordinates of the outer edge of the EWedge.

EWedge::GetOuterPoint

Returns the coordinates of a particular point specified by its location along the outter circle arc.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetOuterPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range [-1, +1]).

EWedge::GetPoint

Returns the coordinates of a particular point specified by its location along the wedge perimeter.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetPoint(
 float breadthFraction,
 float angleFraction
 )
```

Parameters

breadthFraction

Point location expressed as a fraction of the wedge breadth (range -1, +1).

angleFraction

Point location expressed as a fraction of the wedge amplitude (range -1, +1).

EWedge::GetInnerApex

Inner apex point coordinates of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetInnerApex() const
```

EWedge::GetInnerArcLength

Inner circle arc length of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetInnerArcLength() const
```

EWedge::GetInnerDiameter

Inner diameter of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetInnerDiameter() const
```

EWedge::GetInnerEnd

Inner end point coordinates of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetInnerEnd() const
```

EWedge::GetInnerOrg

Inner origin point coordinates of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetInnerOrg() const
```

EWedge::GetInnerRadius

Inner radius of the EWedge

Namespace: Euresys::Open_eVision

[C++]

```
float GetInnerRadius() const
```

EWedge::operator=

Copies all the data from another EWedge object into the current EWedge object

Namespace: Euresys::Open_eVision

[C++]

```
EWedge& operator=(
 const EWedge& other
 )
```

Parameters

other

EWedge object to be copied

EWedge::GetOrgAngle

Angular position from where the EWedge extents.

Namespace: Euresys::Open_eVision

```
[C++]
float GetOrgAngle() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedge::GetOuterApex

Outer apex point coordinates of the EWedge.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetOuterApex() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::GetOuterArcLength

Outer circle arc length of the EWedge.

Namespace: Euresys::Open_eVision

```
[C++]
float GetOuterArcLength() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::GetOuterDiameter

Outer diameter of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOuterDiameter() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::GetOuterEnd

Outer end point coordinates of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetOuterEnd() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::GetOuterOrg

Outer origin point coordinates of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetOuterOrg() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal outer radius (diameter), its breadth (must be negative), the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::GetOuterRadius

Outer radius of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOuterRadius() const
```

Remarks

A EWedge is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedge::SetDiameters

Sets the nominal inner/outer diameter and breadth of the EWedge.

Namespace: Euresys::Open_eVision

[C++]

```
void SetDiameters(
 float diameter,
 float breadth
 )
```

Parameters

diameter

Outer diameter of the EWedge. The default value is 100.

breadth

Breadth of the EWedge. It must be negative or zero. Its default value is -50.

Remarks

A EWedge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal outer radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance.
By default, the EWedge diameter value is 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWedge::SetFromCenterAndOrigin

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromCenterAndOrigin(
 const EPoint& center,
 const EPoint& origin,
 float breadth,
 bool direct
 )
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedge::SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 float breadth,
 bool fullCircle
 )
```

Parameters

origin

Origin point coordinates of the wedge.

middle

Middle point coordinates of the wedge.

end

End point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

fullCircle

true (default) in case of a full turn wedge. If fullCircle is false, origin and end give the wedge's amplitude.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedge::SetFromTwoPoints

This method is deprecated.

DEPRECATED (you should use EWedge::SetFromCenterAndOrigin): Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedge object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromTwoPoints(
 const EPoint& center,
 const EPoint& origin,
 float breadth,
 bool direct
 )
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedge::SetRadii

Sets the nominal radius and breadth of the EWedge.

Namespace: Euresys::Open_eVision

```
[C++]
void SetRadii(
 float radius,
 float breadth
 )
```

Parameters

radius

Outer radius of the EWedge. The default value is 50.

breadth

Breadth of the EWedge, which must be negative or zero. Its default value is -50.

Remarks

A EWedge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance.
By default, the EWedge radius value is 50, which means 50 pixels when the field of view is not calibrated and 50 "units" in case of a calibrated field of view.

4.254. EWedgeGauge Class

Manages a wedge fitting gauge.

Base Class:EWedgeShape

Namespace: Euresys::Open_eVision

Methods

AddSkipRange	Adds an item to the set of skip ranges and returns the index of the newly added range.
CopyTo	Copies all the data of the current EWedgeGauge object into another EWedgeGauge object, and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the gauge.
Draw	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.
EWedgeGauge	Constructs a wedge measurement context.
GetActiveEdges	Active edges as defined in EDragHandle.
GetAverageDistance	Average distance between the sampled points and the fitted model.
GetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

GetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
GetMeasuredPoint	Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.
GetMeasuredWedge	Information pertaining to the fitted wedge.
GetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
GetMinArea	Minimum area value.
GetMinNumFitSamples	Returns the minimum number of samples required for fitting on each side of the shape.
GetNumFilteringPasses	Number of filtering passes for a model fitting operation.
GetNumSamples	Number of sampled points during the model fitting operation.
GetNumSamplesa	Number of sampled points found on edge a during the measure operation.
GetNumSamplesA	Number of sampled points found on edge A during the measure operation.
GetNumSamplesInnerEdge	Number of sampled points found on inner edge during the measure operation.
GetNumSamplesLeftEdge	Number of sampled points found on leftmost edge during the measure operation.
GetNumSamplesOuterEdge	Number of sampled points found on outer edge during the measure operation.
GetNumSamplesr	Number of sampled points found on edge r during the measure operation.
GetNumSamplesR	Number of sampled points found on edge R during the measure operation.
GetNumSamplesRightEdge	Number of sampled points found on rightmost edge during the measure operation.
GetNumSkipRanges	Number of skip ranges in the gauge after a call to EWedgeGauge::AddSkipRange.
GetNumValidSamples	Number of valid sample points remaining after a model fitting operation.
GetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
GetSamplea	Allows to retrieve information on the samples found along the a edge.
GetSampleA	Allows to retrieve information on the samples found along the A edge.
GetSampleInnerEdge	Allows to retrieve information on the samples found along the inner edge.
GetSampleLeftEdge	Allows to retrieve information on the samples found along the left edge.
GetSampleOuterEdge	Allows to retrieve information on the samples found along the outer edge.

GetSampler	Allows to retrieve information on the samples found along the r edge.
GetSampleR	Allows to retrieve information on the samples found along the R edge.
GetSampleRightEdge	Allows to retrieve information on the samples found along the rightmost edge.
GetSamplingStep	Approximate distance between sampled points during a model fitting operation.
GetSkipRange	Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the EWedgeGauge::AddSkipRange method.
GetSmoothing	Number of pixels used for the low-pass filtering operation.
GetThickness	Number of parallel segments used to extract the data profile.
GetThreshold	Threshold level used to delimit significant peaks in the data profile.
GetTolerance	Searching area half thickness of the wedge fitting gauge.
GetTransitionChoice	Transition choice.
GetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
GetTransitionType	Transition type.
GetType	Shape type.
GetValid	Flag indicating if at least one valid transition has been found.
HitTest	Checks whether the cursor is positioned over a handle (true) or not (false).
Measure	Triggers the point location or the model fitting operation.
MeasureSample	Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.
MeasureWithoutFitting	Triggers the point location without wedge fitting operation.
operator=	Copies all the data from another EWedgeGauge object into the current EWedgeGauge object
Plot	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
PlotWithCurrentPen	Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.
Process	Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.
RemoveAllSkipRanges	Removes all the skip ranges previously created by a call to EWedgeGauge::AddSkipRange.
RemoveSkipRange	After a call to EWedgeGauge::AddSkipRange, removes the skip range with the given index.
SetActive	Sets the flag indicating whether the gauge is active or not.
SetActiveEdges	Active edges as defined in EDragHandle.

SetDiameters	Sets the nominal inner/outer diameter and breadth of the EWedgeGauge.
SetFilteringThreshold	Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.
SetFromOriginMiddleEnd	Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeGauge object.
SetFromTwoPoints	DEPRECATED (you should use EWedgeGauge): Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeGauge object.
SetHVConstraint	Status of the restriction on the orientation of the point location gauge or model fitting sample paths.
SetMinAmplitude	Offset added to the Threshold when a peak is to be detected.
SetMinArea	Minimum area value.
SetMinNumFitSamples	Sets the minimum number of samples required for fitting on each side of the shape.
SetNumFilteringPasses	Number of filtering passes for a model fitting operation.
SetRadii	Sets the nominal radius and breadth of the EWedgeGauge.
SetRectangularSamplingArea	Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.
SetSamplingStep	Approximate distance between sampled points during a model fitting operation.
SetSmoothing	Number of pixels used for the low-pass filtering operation.
SetThickness	Number of parallel segments used to extract the data profile.
SetThreshold	Threshold level used to delimit significant peaks in the data profile.
SetTolerance	Searching area half thickness of the wedge fitting gauge.
SetTransitionChoice	Transition choice.
SetTransitionIndex	Index (from 0 on) of the transition to be retained when the transition choice parameter is set to ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd.
SetTransitionType	Transition type.
SetWedge	Sets the nominal position (center coordinates), diameter, breadth, angular origin and amplitude of the wedge fitting gauge according to a known wedge.

EWedgeGauge::SetActive

Sets the flag indicating whether the gauge is active or not.

Namespace: Euresys::Open_eVision

[C++]

```
void SetActive(bool active)
```

Remarks

When complex gauging is required, several gauges can be grouped together. Applying EWedgeGauge::Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process. By default, the gauge is active (true).

EWedgeGauge::GetActiveEdges

EWedgeGauge::SetActiveEdges

Active edges as defined in EDragHandle.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetActiveEdges()

void SetActiveEdges(OEV_UINT32 un32ActiveEdges)
```

Remarks

In the case of a wedge fitting gauge, each edge can have its own transition detection parameters. Updating the transition parameters only affect the current active edges. By default, all edges are active.

EWedgeGauge::AddSkipRange

Adds an item to the set of skip ranges and returns the index of the newly added range.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 AddSkipRange(
 const OEV_UINT32 start,
 const OEV_UINT32 end
 )
```

Parameters

start

Beginning of the skip range.

end

End of the skip range.

Remarks

The samples indices between start and end (including the boundaries) will be discarded during the measurement process.
The EWedgeGauge::AddSkipRange method allows to define skip ranges in an EWedgeGauge. This means that, at measure time, samples belonging to these ranges will not be taken into account.
A sample may belong to more than one skip range; to be discarded, a sample has to pertain to at least one range. Moreover, the skip ranges are allowed to overlap one another.
The range is allowed to be reversed (i.e. end is not required to be greater than start).
Also, start and end are not required to reference valid indices at the time of the call (i.e. the range may lie outside of the current return value for EWedgeGauge::NumSamples).

EWedgeGauge::GetAverageDistance

Average distance between the sampled points and the fitted model.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAverageDistance()
```

Remarks

Irrelevant in case of a point location operation.

EWedgeGauge::CopyTo

Copies all the data of the current EWedgeGauge object into another EWedgeGauge object, and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EWedgeGauge& other,
 bool recursive
 )

EWedgeGauge* CopyTo(
 EWedgeGauge* other,
 bool recursive
 )
```

Parameters

other

Pointer to the EWedgeGauge object in which the current EWedgeGauge object data have to be copied.

recursive

true if the children gauges have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EWedgeGauge object will be created and returned.

EWedgeGauge::Drag

Moves a handle to a new position and updates the position parameters of the gauge.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int x,
 int y
 )
```

Parameters

x

Cursor current X coordinate.

y

Cursor current Y coordinate.

EWedgeGauge::Draw

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

```
void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWedgeGauge::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a point location or model fitting gauge, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWedgeGauge::EWedgeGauge

Constructs a wedge measurement context.

Namespace: Euresys::Open_eVision

```
[C++]
void EWedgeGauge(
 )
void EWedgeGauge(
 const EWedgeGauge& other
 )
```

Parameters

other

Another EWedgeGauge object to be copied in the new EWedgeGauge object.

Remarks

With the default constructor, all the parameters are initialized to their respective default values. With the copy constructor, the constructed wedge measurement context is based on a pre-existing EWedgeGauge object. The gauge children are also copied. Hierarchy copying through a copy constructor is always recursive. To disable this recursion, use instead the EWedgeGauge::CopyTo method.

EWedgeGauge::GetFilteringThreshold

EWedgeGauge::SetFilteringThreshold

Relative filtering threshold, that is the fraction of the average distance between the sampled points and the fitted model above which a point is filtered out.

Namespace: Euresys::Open_eVision

```
[C++]
float GetFilteringThreshold()
void SetFilteringThreshold(float f32FilteringThreshold)
```

Remarks

Irrelevant in case of a point location operation. During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious).

EWedgeGauge::GetMeasuredPoint

Returns the coordinates of a sample point, measured along one of the sample paths of the gauge.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetMeasuredPoint(
 OEV_UINT32 index
 )
```

Parameters

index

This argument must be left unchanged from its default value, i.e. ~0 (= 0xFFFFFFFF).

Remarks

These coordinates pertain to the world space; they are expressed in the reference frame to which the current EWedgeGauge object belongs. The gauging process uses a list of sample points to find the shape position and size that best fit a given image. These sample points are measured along the sample paths defined by the gauge geometry. EWedgeGauge::GetMeasuredPoint returns the coordinates of the sample point that meets the following two requirements: 1. It lies on the sample path inspected with the last call to EWedgeGauge::MeasureSample, and 1. Among all the sample points along the latter sample path, it is the one selected by the EWedgeGauge::TransitionChoice property.

Note. For this method to succeed, it is necessary to previously call EWedgeGauge::MeasureSample.

EWedgeGauge::GetMinNumFitSamples

Returns the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

```
[C++]
void GetMinNumFitSamples(
 int& side0,
 int& side1,
 int& side2,
 int& side3
 )
```

Parameters

side0

Minimum number of samples on the top side of the rectangle.

side1

Minimum number of samples on the left side of the rectangle.

side2

Minimum number of samples on the bottom side of the rectangle.

side3

Minimum number of samples on the right side of the rectangle.

Remarks

Irrelevant in case of a point location operation.

EWedgeGauge::GetSampleA

This method is deprecated.

Allows to retrieve information on the samples found along the A edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleA(
 EPoint& pt,
 OEV_UINT32 index
 )
void GetSampleA(
 ESamplePoint& sp,
 OEV_UINT32 index
 )
bool GetSampleA(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use EWedgeGauge::GetSampleRightEdge instead.

EWedgeGauge::GetSampleA

This method is deprecated.

Allows to retrieve information on the samples found along the A edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleA(
 EPoint& pt,
 OEV_UINT32 index
 )
void GetSampleA(
 ESamplePoint& sp,
 OEV_UINT32 index
 )
bool GetSampleA(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use EWedgeGauge::GetSampleRightEdge instead.

EWedgeGauge::GetSampleInnerEdge

Allows to retrieve information on the samples found along the inner edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleInnerEdge(
 EPoint& pt,
 OEV_UINT32 index
 )
```

```
void GetSampleInnerEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleInnerEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge::GetSampleLeftEdge

Allows to retrieve information on the samples found along the left edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleLeftEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleLeftEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleLeftEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge::GetSampleOuterEdge

Allows to retrieve information on the samples found along the outer edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleOuterEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleOuterEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleOuterEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge::GetSampleR

This method is deprecated.

Allows to retrieve information on the samples found along the R edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleR(
 EPoint& pt,
 OEV_UINT32 index
 )
void GetSampleR(
 ESamplePoint& sp,
 OEV_UINT32 index
 )
bool GetSampleR(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use EWedgeGauge::GetSampleInnerEdge instead.

EWedgeGauge::GetSampleR

This method is deprecated.

Allows to retrieve information on the samples found along the R edge.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetSampleR(
 EPoint& pt,
 OEV_UINT32 index
 )
```

```
void GetSampleR(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleR(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

Deprecation notice: Use EWedgeGauge::GetSampleInnerEdge instead.

EWedgeGauge::GetSampleRightEdge

Allows to retrieve information on the samples found along the rightmost edge.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetSampleRightEdge(
 EPoint& pt,
 OEV_UINT32 index
 )

void GetSampleRightEdge(
 ESamplePoint& sp,
 OEV_UINT32 index
 )

bool GetSampleRightEdge(
 EPeak& pk,
 OEV_UINT32 index
 )
```

Parameters

pt

EPoint structure that will receive the sample position.

index

The sample index

sp

ESamplePoint structure that will receive the sample position.

pk

EPeak structure that will contain the sample peak properties.

Remarks

The method provides the sample point corresponding to the supplied index.
The returned value is true when the sample is valid, and false otherwise.

EWedgeGauge::GetSkipRange

Allows to retrieve the start and end values of the skip range corresponding to the given index, if it is valid (i.e. if it has previously been created by a call to the EWedgeGauge::AddSkipRange method.

Namespace: Euresys::Open_eVision

```
[C++]
void GetSkipRange(
 const OEV_UINT32 index,
 OEV_UINT32& start,
 OEV_UINT32& end
 )
```

Parameters

index

Index of the skip range.

start

Beginning of the skip range.

end

End of the skip range.

Remarks

Start is guaranteed to be smaller or equal to end.

EWedgeGauge::HitTest

Checks whether the cursor is positioned over a handle (true) or not (false).

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool daughters
 )
```

Parameters

daughters

true if the daughters gauges handles have to be considered as well.

EWedgeGauge::GetHVConstraint

EWedgeGauge::SetHVConstraint

Status of the restriction on the orientation of the point location gauge or model fitting sample paths.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetHVConstraint()
void SetHVConstraint(bool bHVConstraint)
```

Remarks

Sample paths are the point location gauges placed along the model to be fitted.

EWedgeGauge::Measure

Triggers the point location or the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
void Measure(
 EROIBW8* sourceImage
 )
void Measure(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Pointer to the source image.

region

Region to use with the source image.

Remarks

When this method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EWedgeGauge::GetMeasuredWedge

Information pertaining to the fitted wedge.

Namespace: Euresys::Open_eVision

[C++]

```
EWedge GetMeasuredWedge()
```

Remarks

Use method EShape::GetFound to get the status of the measurement.
EWedgeGauge::MeasuredWedge returns a successful fitted wedge if EShape::GetFound is true, otherwise it returns the orginal (nominal) wedge.

EWedgeGauge::MeasureSample

Computes the sample points along the sample path whose index in the list is given by the pathIndex parameter.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureSample(
 EROIBW8* sourceImage,
 OEV_UINT32 pathIndex
 )

void MeasureSample(
 EROIBW8* sourceImage,
 const ERegion& region,
 OEV_UINT32 pathIndex
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

pathIndex

Sample path index.

region

Region on which to measure.

Remarks

This method stores its results into a temporary variable inside the EWedgeGauge object.

EWedgeGauge::MeasureWithoutFitting

Triggers the point location without wedge fitting operation.

Namespace: Euresys::Open_eVision

[C++]

```
void MeasureWithoutFitting(
 EROIBW8* sourceImage
 )

void MeasureWithoutFitting(
 EROIBW8* sourceImage,
 const ERegion& region
 )
```

Parameters

sourceImage

Source image.

region

-

Remarks

This method performs the actual measurement for each transition, but does not perform the wedge fitting. This means that individual samples will be available for each edges through the EWedgeGauge::GetSamplea (Edge a), EWedgeGauge::GetSampler (Edge r), EWedgeGauge::GetSampleA (Edge A), EWedgeGauge::GetSampleR (Edge R) methods, but the gauge position will not be changed.
Please note that the filtering will not be performed in this method, since it relies upon the fitting process. The filtering parameters will thus be unused.

EWedgeGauge::GetMinAmplitude

EWedgeGauge::SetMinAmplitude

Offset added to the Threshold when a peak is to be detected.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMinAmplitude()
void SetMinAmplitude(OEV_UINT32 un32MinAmplitude)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value.
To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected.
When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

EWedgeGauge::GetMinArea

EWedgeGauge::SetMinArea

Minimum area value.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetMinArea()
void SetMinArea(OEV_UINT32 un32MinArea)
```

Remarks

A transition is detected if its derivative peak reaches Threshold + MinAmplitude value, and then declared valid if the area between the peak curve and the horizontal at level Threshold reaches the MinArea value.

EWedgeGauge::GetNumFilteringPasses

EWedgeGauge::SetNumFilteringPasses

Number of filtering passes for a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumFilteringPasses()
void SetNumFilteringPasses(OEV_UINT32 un32NumFilteringPasses)
```

Remarks

Irrelevant in case of a point location operation. During a filtering pass, the points that are too distant from the model are discarded. During a model fitting operation, the "filtering" process can be invoked to remove outliers, i.e. points that were located significantly far away from the fitted model (so that their position is dubious). By default (the number of filtering passes is 0), the outliers rejection process is disabled.

EWedgeGauge::GetNumSamples

Number of sampled points during the model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamples() const
```

Remarks

Irrelevant in case of a point location operation.
After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

EWedgeGauge::GetNumSamplesA

This property is deprecated.

Number of sampled points found on edge A during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesA() const
```

Remarks

Deprecation notice: Use EWedgeGauge instead.

EWedgeGauge::GetNumSamplesA

This property is deprecated.

Number of sampled points found on edge A during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesA() const
```

Remarks

Deprecation notice: Use EWedgeGauge instead.

EWedgeGauge::GetNumSamplesInnerEdge

Number of sampled points found on inner edge during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesInnerEdge() const
```

EWedgeGauge::GetNumSamplesLeftEdge

Number of sampled points found on leftmost edge during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesLeftEdge() const
```

EWedgeGauge::GetNumSamplesOuterEdge

Number of sampled points found on outer edge during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesOuterEdge() const
```

EWedgeGauge::GetNumSamplesR

This property is deprecated.

Number of sampled points found on edge R during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesR() const
```

Remarks

Deprecation notice: Use EWedgeGauge instead.

EWedgeGauge::GetNumSamplesR

This property is deprecated.

Number of sampled points found on edge R during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesR() const
```

Remarks

Deprecation notice: Use EWedgeGauge instead.

EWedgeGauge::GetNumSamplesRightEdge

Number of sampled points found on rightmost edge during the measure operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSamplesRightEdge() const
```

EWedgeGauge::GetNumSkipRanges

Number of skip ranges in the gauge after a call to EWedgeGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumSkipRanges() const
```

EWedgeGauge::GetNumValidSamples

Number of valid sample points remaining after a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetNumValidSamples()
```

Remarks

Irrelevant in case of a point location operation. After a model fitting operation, a number of points have been fitted along the model. Among them, some are not reliable because of their Area value. Among the remaining ones, some are filtered out (NumFilteringPasses, FilteringThreshold).

EWedgeGauge::operator=

Copies all the data from another EWedgeGauge object into the current EWedgeGauge object

Namespace: Euresys::Open_eVision

[C++]

```
EWedgeGauge& operator=(
 const EWedgeGauge& other
 )
```

Parameters

other

EWedgeGauge object to be copied

EWedgeGauge::Plot

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void Plot(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )

void Plot(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.

height
Height of the plot.

originX
Origin point coordinates of the plot along the X axis.

originY
Origin point coordinates of the plot along the Y axis.

color
The color in which to draw the overlay.

Remarks

The sample path that is taken into considered is the one inspected with the last call to EWedgeGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call EWedgeGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWedgeGauge::PlotWithCurrentPen

This method is deprecated.

Draws the profile that is crossed by one of the sample paths of the gauge, as defined by EPlotItem.

Namespace: Euresys::Open_eVision

[C++]

```
void PlotWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EPlotItem drawItems,
 float width,
 float height,
 float originX,
 float originY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

drawItems
Boolean combination of EPlotItem members, that indicates which items must be displayed.

width
Width of the plot.
height
Height of the plot.
originX
Origin point coordinates of the plot along the X axis.
originY
Origin point coordinates of the plot along the Y axis.

Remarks

The sample path that is taken into considered is the one inspected with the last call to EWedgeGauge::MeasureSample.

Note. For this method to succeed, it is necessary to previously call EWedgeGauge::MeasureSample.
Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWedgeGauge::Process

Triggers the process pertaining to a shape or gauge and all the daughter gauges attach to it.

Namespace: Euresys::Open_eVision

[C++]

```
void Process(
 EROIBW8* sourceImage,
 bool daughters
 )

void Process(
 EROIBW8* sourceImage,
 const ERegion& region,
 bool daughters
 )
```

Parameters

sourceImage
Pointer to the source image.
daughters
Flag indicating whether the daughters shapes inherit of the same behavior.
region
Region to use with the source image.

Remarks

When complex gauging is required, several gauges can be grouped together. Applying Process to the mother gauge or shape triggers the measurement of the whole. Only the active gauges will participate in the process.

EWedgeGauge::GetRectangularSamplingArea

EWedgeGauge::SetRectangularSamplingArea

Flag indicating whether the sampling area remains a rectangle when rotated, instead of becoming a parallelogram.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetRectangularSamplingArea()

void SetRectangularSamplingArea(bool bRectangularSamplingArea)
```

Remarks

By default, this flag is set to true: the sampling area always remains a rectangle. Setting this property is only useful when the thickness transition parameter is greater than 1. In fact, when thickness transition parameter is equal to 1, rectangle and parallelogram reduce to the same segment.

EWedgeGauge::RemoveAllSkipRanges

Removes all the skip ranges previously created by a call to EWedgeGauge::AddSkipRange.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveAllSkipRanges(
 )
```

EWedgeGauge::RemoveSkipRange

After a call to EWedgeGauge::AddSkipRange, removes the skip range with the given index.

Namespace: Euresys::Open_eVision

[C++]

```
void RemoveSkipRange(
 const OEV_UINT32 index
 )
```

Parameters

index

Index of the skip range to remove, as returned by EWedgeGauge::AddSkipRange.

EWedgeGauge::GetSamplingStep

EWedgeGauge::SetSamplingStep

Approximate distance between sampled points during a model fitting operation.

Namespace: Euresys::Open_eVision

```
[C++]
float GetSamplingStep()
void SetSamplingStep(float f32SamplingStep)
```

Remarks

Irrelevant in case of a point location operation.
To fit a model, a series of point location operations are performed along the model. The point location gauges spacing is given by the sampling step.
By default, the sampling step is set to 5, which means 5 pixels when the field of view is not calibrated, and 5 physical units in case of a calibrated field of view.
Be aware that if the sampling step is too large, the number of sampled points along the model will not be sufficient enough to accurately locate it.

EWedgeGauge::SetDiameters

Sets the nominal inner/outer diameter and breadth of the EWedgeGauge.

Namespace: Euresys::Open_eVision

```
[C++]
void SetDiameters(
 float diameter,
 float breadth
 )
```

Parameters

diameter

Outer diameter of the EWedgeGauge. The default value is 100.

breadth

Breadth of the EWedgeGauge. It must be negative or zero. Its default value is -50.

Remarks

A EWedgeGauge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal outer radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance.
By default, the EWedgeGauge diameter value is 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWedgeGauge::SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeGauge object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 float breadth,
 bool fullCircle
 )
```

Parameters

origin

Origin point coordinates of the wedge.

middle

Middle point coordinates of the wedge.

end

End point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

fullCircle

true (default) in case of a full turn wedge. If fullCircle is false, origin and end give the wedge's amplitude.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedgeGauge::SetFromTwoPoints

DEPRECATED (you should use EWedgeGauge): Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeGauge object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromTwoPoints(
 const EPoint& center,
 const EPoint& origin,
 float breadth,
 bool direct
 )
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedgeGauge::SetMinNumFitSamples

Sets the minimum number of samples required for fitting on each side of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void SetMinNumFitSamples(
 int side0,
 int side1,
 int side2,
 int side3
 )
```

Parameters

side0

Minimum number of samples on the *outer circle* of the wedge. The default value is 3.

side1

Minimum number of samples on the *original border* of the wedge. If this value is not specified, it is equal to n32Side0. The default value is 2.

side2

Minimum number of samples on the *inner circle* of the wedge. If this value is not specified, it is equal to n32Side0. The default value is 3.

side3

Minimum number of samples on the *end border* of the wedge. If this value is not specified, it is equal to n32Side1. The default value is 2.

Remarks

Irrelevant in case of a point location operation. When the EWedgeGauge::Measure method is called, and if not enough valid points are detected, then the method returns directly, and the measured gauge is set to the nominal parameters.

EWedgeGauge::SetRadii

Sets the nominal radius and breadth of the EWedgeGauge.

Namespace: Euresys::Open_eVision

```
[C++]
void SetRadii(
 float radius,
 float breadth
 )
```

Parameters

radius

Outer radius of the EWedgeGauge. The default value is 50.

breadth

Breadth of the EWedgeGauge, which must be negative or zero. Its default value is -50.

Remarks

A EWedgeGauge is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance.
By default, the EWedgeGauge radius value is 50, which means 50 pixels when the field of view is not calibrated and 50 "units" in case of a calibrated field of view.

EWedgeGauge::GetSmoothing

EWedgeGauge::SetSmoothing

Number of pixels used for the low-pass filtering operation.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetSmoothing()
void SetSmoothing(OEV_UINT32 un32Smoothing)
```

Remarks

To reduce the effect of noise, the profile data can be low-pass filtered along the point location gauge direction.

EWedgeGauge::GetThickness

EWedgeGauge::SetThickness

Number of parallel segments used to extract the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThickness()
void SetThickness(OEV_UINT32 un32Thickness)
```

Remarks

To reduce the effect of noise and/or strengthen a transition, several parallel profiles can be accumulated.

EWedgeGauge::GetThreshold

EWedgeGauge::SetThreshold

Threshold level used to delimit significant peaks in the data profile.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 GetThreshold()
void SetThreshold(OEV_UINT32 un32Threshold)
```

Remarks

When analyzing a derivative profile, a peak is made up of consecutive pixel values above Threshold. To detect weak [strong] transitions, lower [raise] the Threshold value.
To avoid interference of noise, an additional parameter is provided. The MinAmplitude parameter is an offset added to Threshold when a peak is to be detected.
When the pixel values of the derivative profile do not reach Threshold + MinAmplitude, the peak is not taken into account. Anyway, when a peak is taken into account, all the pixels with values above Threshold are considered (for more accuracy). Setting the MinAmplitude value to 0 merely cancels its effect.

EWedgeGauge::GetTolerance

EWedgeGauge::SetTolerance

Searching area half thickness of the wedge fitting gauge.

Namespace: Euresys::Open_eVision

[C++]

```
float GetTolerance()
void SetTolerance(float f32Tolerance)
```

Remarks

A wedge fitting gauge is fully defined knowing its nominal position (its center coordinates), its outer nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
By default, the searching area thickness of the wedge fitting gauge is 20 (2x10), which means 20 pixels when the field of view is not calibrated, and 20 physical units in case of a calibrated field of view.

EWedgeGauge::GetTransitionChoice

EWedgeGauge::SetTransitionChoice

Transition choice.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::ETransitionChoice GetTransitionChoice()
void SetTransitionChoice(Euresys::Open_eVision::ETransitionChoice transitionChoice)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition.
In case of ETransitionChoice_NthFromBegin or ETransitionChoice_NthFromEnd transition choice, set EWedgeGauge::TransitionIndex to specify the desired transition.
By default, the selected transition corresponds to the one with the largest amplitude (ETransitionChoice_LargestAmplitude).

EWedgeGauge::GetTransitionIndex

EWedgeGauge::SetTransitionIndex

Index (from 0 on) of the transition to be retained when the transition choice parameter is set to NthFromBegin or NthFromEnd.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetTransitionIndex()
void SetTransitionIndex(OEV_UINT32 un32TransitionIndex)
```

Remarks

Several peaks may be detected along a point location gauge. This parameter helps to select the desired transition.
By default, the first transition is retained (the index value is 0).

EWedgeGauge::GetTransitionType

EWedgeGauge::SetTransitionType

Transition type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::ETransitionType GetTransitionType()
void SetTransitionType(Euresys::Open_eVision::ETransitionType transitionType)
```

Remarks

The type of a transition tells whether it crosses increasing or decreasing gray-level values. This helps discriminate between nearby edges of an object.
By default, the searched transition type is indifferently a black to white or a white to black transition (ETransitionType_BwOrWb).

EWedgeGauge::GetType

Shape type.

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EShapeType GetType()
```

EWedgeGauge::GetValid

Flag indicating if at least one valid transition has been found.

Namespace: Euresys::Open_eVision

```
[C++]
bool GetValid()
```

Remarks

A false value means that no measurement has been performed. A true value means that a transition was found along the sample path inspected with the last call to EWedgeGauge::MeasureSample, and thus a point has been measured.

EWedgeGauge::SetWedge

Sets the nominal position (center coordinates), diameter, breadth, angular origin and amplitude of the wedge fitting gauge according to a known wedge.

Namespace: Euresys::Open_eVision

[C++]

```
void SetWedge(const EWedge& wedge)
```

4.255. EWedgeShape Class

Manages a wedge shape.

Base Class:EShape
Derived Class(es):EWedgeGauge
Namespace: Euresys::Open_eVision

Methods

Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
CopyTo	Copies all the data of the current EWedgeShape object into another EWedgeShape object and returns it.
Drag	Moves a handle to a new position and updates the position parameters of the shape.
Draw	Draws a graphical representation of a shape, as defined by EDrawingMode.
DrawWithCurrentPen	Draws a graphical representation of a shape, as defined by EDrawingMode.
GetAmplitude	Angular amplitude of the EWedgeShape.
GetAngle	Orientation of the shape.
GetApexAngle	Angular position at the apex of the EWedgeShape.
GetBreadth	Breadth of the EWedgeShape.
GetCenter	Center point of the shape.
GetCenterX	Abscissa of the origin point of the shape.
GetCenterY	Ordinate of the origin point of the shape.
GetCorners	Retrieves the coordinates of each corner of the EWedgeShape.
GetDirect	Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.
GetEdges	Retrieves each edge of the EWedgeShape.

GetEndAngle	Ending angular position of the EWedgeShape.
GetFullBreadth	Flag indicating if the EWedgeShape has a full breadth (breadth = radius).
GetFullCircle	Flag indicating if the EWedgeShape is a full circle (amplitude = 2 PI).
GetInnerApex	Inner apex point coordinates of the EWedgeShape.
GetInnerArcLength	Inner circle arc length of the EWedgeShape.
GetInnerDiameter	Inner diameter of the EWedgeShape.
GetInnerEnd	Inner end point coordinates of the EWedgeShape.
GetInnerOrg	Inner origin point coordinates of the EWedgeShape.
GetInnerPoint	Returns the coordinates of a particular point specified by its location along the inner circle arc.
GetInnerRadius	Inner radius of the EWedgeShape
GetMidEdges	Retrieves the center coordinates of each edge of the wedge fitting gauge.
GetOrgAngle	Angular position from where the EWedgeShape extents.
GetOuterApex	Outer apex point coordinates of the EWedgeShape.
GetOuterArcLength	Outer circle arc length of the EWedgeShape.
GetOuterDiameter	Outer diameter of the EWedgeShape.
GetOuterEnd	Outer end point coordinates of the EWedgeShape.
GetOuterOrg	Outer origin point coordinates of the EWedgeShape.
GetOuterPoint	Returns the coordinates of a particular point specified by its location along the outter circle arc.
GetOuterRadius	Outer radius of the EWedgeShape.
GetPoint	Returns the coordinates of a particular point specified by its location along the wedge perimeter.
GetScale	Horizontal sensor resolution, in pixels per unit.
GetType	Shape type.
HitTest	Checks if there is a handle under the cursor.
operator=	Copies all the data from another EWedgeShape object into the current EWedgeShape object
SetAmplitude	Angular amplitude of the EWedgeShape.
SetAngle	Orientation of the shape.
SetCenter	Center point of the shape.
SetCenterXY	Sets the center coordinates of a EWedgeShape object.
SetDiameters	Sets the nominal inner/outer diameter and breadth of the EWedgeShape.
SetFromCenterAndOrigin	Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeShape object.

SetFromOriginMiddleEnd	Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeShape object.
SetFromTwoPoints	DEPRECATED (you should use EWedgeShape::SetFromCenterAndOrigin): Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeShape object.
SetRadii	Sets the nominal radius and breadth of the EWedgeShape.
SetScale	Horizontal sensor resolution, in pixels per unit.
SetWedge	Sets the nominal position, length and rotation angle of the wedge according to a known EWedge object.

EWedgeShape::GetAmplitude

EWedgeShape::SetAmplitude

Angular amplitude of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAmplitude() const

void SetAmplitude(float amplitude)
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedgeShape::GetAngle

EWedgeShape::SetAngle

Orientation of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const
void SetAngle(float f32Angle)
```

EWedgeShape::GetApexAngle

Angular position at the apex of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetApexAngle() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedgeShape::GetBreadth

Breadth of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetBreadth() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetCenter

EWedgeShape::SetCenter

Center point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const
void SetCenter(const EPoint& center)
```

EWedgeShape::GetCenterX

Abscissa of the origin point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

EWedgeShape::GetCenterY

Ordinate of the origin point of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

EWedgeShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

EWedgeShape::CopyTo

Copies all the data of the current EWedgeShape object into another EWedgeShape object and returns it.

Namespace: Euresys::Open_eVision

[C++]

```
void CopyTo(
 EWedgeShape& dest,
 bool bRecursive
 )

EWedgeShape* CopyTo(
 EWedgeShape* dest,
 bool bRecursive
 )
```

Parameters

dest

Pointer to the EWedgeShape object in which the current EWedgeShape object data have to be copied.

bRecursive

true if the children shapes have to be copied as well, false otherwise.

Remarks

Deprecation notice: the overload taking and returning a pointer is deprecated. In that overload, in case of a NULL pointer, a new EWedgeShape object will be created and returned.

EWedgeShape::GetDirect

EWedgeShape::SetDirect

Flag indicating whether positive angles correspond to a clockwise or an anticlockwise rotation.

Namespace: Euresys::Open_eVision

[C++]

```
bool GetDirect()

void SetDirect(bool direct)
```

Remarks

true (default) means that angles increase anticlockwisely in a direct coordinate system, and clockwisely in an inverse coordinate system.
 * When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards.
 * When the field of view is not calibrated, the coordinate system is said to be inverse the abscissa extends rightwards and the ordinate extends downwards.

EWedgeShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

EWedgeShape::Draw

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

color

The color to draw with.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWedgeShape::DrawWithCurrentPen

This method is deprecated.

Draws a graphical representation of a shape, as defined by EDrawingMode.

Namespace: Euresys::Open_eVision

[C++]

```
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingMode,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingMode

Indicates how the point location or model fitting gauge must be displayed, as defined by EDrawingMode.

daughters

true if the daughters gauges are to be displayed also.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWedgeShape::GetEndAngle

Ending angular position of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetEndAngle() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedgeShape::GetFullBreadth

Flag indicating if the EWedgeShape has a full breadth (breadth = radius).

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFullBreadth() const
```

EWedgeShape::GetFullCircle

Flag indicating if the EWedgeShape is a full circle (amplitude = 2 PI).

Namespace: Euresys::Open_eVision

[C++]

```
bool GetFullCircle() const
```

EWedgeShape::GetCorners

Retrieves the coordinates of each corner of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
void GetCorners(
 EPoint& ar,
 EPoint& AAr,
 EPoint& aRR,
 EPoint& AARR
 )
```

Parameters

ar
Coordinates of the inner org corner of the EWedgeShape.

AAr
Coordinates of the inner end corner of the EWedgeShape.

aRR
Coordinates of the outer org corner of the EWedgeShape.

AARR
Coordinates of the outer end corner of the EWedgeShape.

EWedgeShape::GetEdges

Retrieves each edge of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
void GetEdges(
 ELine& a,
 ELine& AA,
 ECircle& r,
 ECircle& RR
 )
```

Parameters

a
Org edge of the EWedgeShape.

AA
End edge of the EWedgeShape.

r
Inner edge of the EWedgeShape.

RR
Outer edge of the EWedgeShape.

EWedgeShape::GetInnerPoint

Returns the coordinates of a particular point specified by its location along the inner circle arc.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetInnerPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range [-1, +1]).

EWedgeShape::GetMidEdges

Retrieves the center coordinates of each edge of the wedge fitting gauge.

Namespace: Euresys::Open_eVision

```
[C++]
void GetMidEdges(
 EPoint& a,
 EPoint& AA,
 EPoint& r,
 EPoint& RR
 )
```

Parameters

a

Center coordinates of the org edge of the EWedgeShape.

AA

Center coordinates of the end edge of the EWedgeShape.

r

Center coordinates of the inner edge of the EWedgeShape.

RR

Center coordinates of the outer edge of the EWedgeShape.

EWedgeShape::GetOuterPoint

Returns the coordinates of a particular point specified by its location along the outter circle arc.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetOuterPoint(
 float fraction
 )
```

Parameters

fraction

Point location expressed as a fraction of the circle arc (range [-1, +1]).

EWedgeShape::GetPoint

Returns the coordinates of a particular point specified by its location along the wedge perimeter.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetPoint(
 float breadthFraction,
 float angleFraction
 )
```

Parameters

breadthFraction

Point location expressed as a fraction of the wedge breadth (range -1, +1).

angleFraction

Point location expressed as a fraction of the wedge amplitude (range -1, +1).

EWedgeShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

```
[C++]
bool HitTest(
 bool daughters
 )
```

Parameters

daughters

-

EWedgeShape::GetInnerApex

Inner apex point coordinates of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetInnerApex() const
```

EWedgeShape::GetInnerArcLength

Inner circle arc length of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
float GetInnerArcLength() const
```

EWedgeShape::GetInnerDiameter

Inner diameter of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
float GetInnerDiameter() const
```

EWedgeShape::GetInnerEnd

Inner end point coordinates of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetInnerEnd() const
```

EWedgeShape::GetInnerOrg

Inner origin point coordinates of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint GetInnerOrg() const
```

EWedgeShape::GetInnerRadius

Inner radius of the EWedgeShape

Namespace: Euresys::Open_eVision

[C++]

```
float GetInnerRadius() const
```

EWedgeShape::operator=

Copies all the data from another EWedgeShape object into the current EWedgeShape object

Namespace: Euresys::Open_eVision

[C++]

```
EWedgeShape& operator=(
 const EWedgeShape& other
 )
```

Parameters

other

EWedgeShape object to be copied

EWedgeShape::GetOrgAngle

Angular position from where the EWedgeShape extents.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOrgAngle() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.
The sign of the rotation angle depends whether the field of view is calibrated or not.
* When the field of view is calibrated, the coordinate system is said to be direct, the abscissa extends rightwards and the ordinate extends upwards. In this case, an anticlockwise rotation leads to a positive angle value.
* When the field of view is not calibrated, the coordinate system is said to be inverse, the abscissa extends rightwards and the ordinate extends downwards. In this case, a clockwise rotation leads to a positive angle value.

EWedgeShape::GetOuterApex

Outer apex point coordinates of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetOuterApex() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetOuterArcLength

Outer circle arc length of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOuterArcLength() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetOuterDiameter

Outer diameter of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOuterDiameter() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetOuterEnd

Outer end point coordinates of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetOuterEnd() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetOuterOrg

Outer origin point coordinates of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetOuterOrg() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal outer radius (diameter), its breadth (must be negative), the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetOuterRadius

Outer radius of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
float GetOuterRadius() const
```

Remarks

A EWedgeShape is fully defined knowing its nominal position (its center coordinates), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude, and its outline tolerance.

EWedgeShape::GetScale

EWedgeShape::SetScale

Horizontal sensor resolution, in pixels per unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetScale() const

void SetScale(float f32Scale)
```

EWedgeShape::SetCenterXY

Sets the center coordinates of a EWedgeShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Center coordinates of the EWedgeShape object.

centerY

Center coordinates of the EWedgeShape object.

EWedgeShape::SetDiameters

Sets the nominal inner/outer diameter and breadth of the EWedgeShape.

Namespace: Euresys::Open_eVision

```
[C++]
void SetDiameters(
 float diameter,
 float breadth
 )
```

Parameters

diameter

Outer diameter of the EWedgeShape. The default value is 100.

breadth

Breadth of the EWedgeShape. It must be negative or zero. Its default value is -50.

Remarks

A EWedgeShape is fully defined knowing its nominal position (given by the coordinates of its center), its nominal outer radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance.
By default, the EWedgeShape diameter value is 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWedgeShape::SetFromCenterAndOrigin

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromCenterAndOrigin(
 const EPoint& center,
 const EPoint& origin,
 float breadth,
 bool direct
 )
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedgeShape::SetFromOriginMiddleEnd

Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeShape object.

Namespace: Euresys::Open_eVision

```
[C++]
void SetFromOriginMiddleEnd(
 const EPoint& origin,
 const EPoint& middle,
 const EPoint& end,
 float breadth,
 bool fullCircle
 )
```

Parameters

origin

Origin point coordinates of the wedge.

middle

Middle point coordinates of the wedge.

end

End point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

fullCircle

true (default) in case of a full turn wedge. If fullCircle is false, origin and end give the wedge's amplitude.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedgeShape::SetFromTwoPoints

This method is deprecated.

DEPRECATED (you should use EWedgeShape::SetFromCenterAndOrigin): Sets the geometric parameters (nominal position and diameter, breadth, angular position and amplitude) of an EWedgeShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFromTwoPoints(
 const EPoint& center,
 const EPoint& origin,
 float breadth,
 bool direct
 )
```

Parameters

center

Center coordinates of the wedge at its nominal position. The default value is (0,0).

origin

Origin point coordinates of the wedge.

breadth

Nominal breadth of the wedge. It must be negative or zero. The default value is -50.

direct

true (default) means that angles increase anticlockwise in a direct coordinate system and clockwise in an inverse coordinate system.

Remarks

In a direct coordinate system, the abscissa extends rightwards and the ordinate extends upwards. The coordinate system is said to be inverse if the abscissa extends rightwards and the ordinate extends downwards.

EWedgeShape::SetRadii

Sets the nominal radius and breadth of the EWedgeShape.

Namespace: Euresys::Open_eVision

[C++]

```
void SetRadii(
 float outerRadius,
 float breadth
 )
```

Parameters

outerRadius

Outer radius of the EWedgeShape. The default value is 50.

breadth

Breadth of the EWedgeShape, which must be negative or zero. Its default value is -50.

Remarks

A EWedgeShape is fully defined knowing its nominal position (given by the coordinates of its center), its nominal radius (diameter), its breadth, the angular position from where it extents, its angular amplitude and its outline tolerance.
By default, the EWedgeShape radius value is 50, which means 50 pixels when the field of view is not calibrated and 50 "units" in case of a calibrated field of view.

EWedgeShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

EWedgeShape::SetWedge

Sets the nominal position, length and rotation angle of the wedge according to a known EWedge object.

Namespace: Euresys::Open_eVision

[C++]

```
void SetWedge(const EWedge& wedge)
```

4.256. EWindowsDrawAdapter Class

A draw adapter using the GDI+ native API on Windows. This class is only usable on Windows and all its methods will throw NotImplemented errors if used on another platform.

Base Class:EDrawAdapter
Derived Class(es):EGDIPlusDrawAdapter
Namespace: Euresys::Open_eVision

Methods

Arc	Draws an arc defined by a rectangle, angle, and amplitude.
BackedText	Draws a text with a background.
Close	Closes the attached drawing context, if the current instance was initialized from an image.
DrawPoint	Draws a point at the given coordinate.
Ellipse	Draws an ellipse.
EWindowsDrawAdapter	Creates an EWindowsDrawAdapter either by providing a drawing context or an image. If a drawing context is provided EWindowsDrawAdapter will forward all drawing commands to that drawing context. If an image is provided a drawing context will be initialized on that image and that drawing context will be closed either on destruction of the EWindowsDrawAdapter or when the EWindowsDrawAdapter::Close method is called.
FilledEllipse	Fills an ellipse.
FilledPolygon	Fills a polygon.
FilledRectangle	Fills a rectangle.
GetHDC	Gets the HDC associated with this instance of EWindowsDrawAdapter.
GetTextSize	Size of the given text using the default font (EWindowsDrawAdapter::Font).
Image	Draws an image.
Line	Draws a line between two points.
Polygon	Draws a polygon.
Rectangle	Draws a rectangle.
SetBrush	Default brush.
SetFont	Default font.
SetPen	Default pen.

Text	Draws a text.
UseCurrentBrush	Use the current pen set in the drawing framework.
UseCurrentPen	Uses the current GDI pen.

EWindowsDrawAdapter::Arc

Draws an arc defined by a rectangle, angle, and amplitude.

Namespace: Euresys::Open_eVision

```
[C++]
void Arc(
 int orgX,
 int orgY,
 int width,
 int height,
 float startAngle,
 float amplitude,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

startAngle
Starting angle of the arc in radians

amplitude
Amplitude of the arc in radians

pen
Optional pen to use

EWindowsDrawAdapter::BackedText

Draws a text with a background.

Namespace: Euresys::Open_eVision

[C++]

```
void BackedText(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush,
 EBrush backgroundBrush
 )
```

Parameters

text
Text to draw.

x
X position of the text.

y
Y position of the text.

orientation
Orientation of the text.

textBrush
Optional brush to use for the color of the text.

backgroundBrush
Optional brush to use for the background.

EWindowsDrawAdapter::SetBrush

Default brush.

Namespace: Euresys::Open_eVision

[C++]

```
void SetBrush(const EBrush& brush)
```

EWindowsDrawAdapter::Close

Closes the attached drawing context, if the current instance was initialized from an image.

Namespace: Euresys::Open_eVision

[C++]

```
void Close(
 )
```

EWindowsDrawAdapter::DrawPoint

Draws a point at the given coordinate.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawPoint(
 int x1,
 int y1,
 EPen pen
 )
```

Parameters

x1
X coordinate

y1
Y coordinate

pen
Optional pen to use

EWindowsDrawAdapter::Ellipse

Draws an ellipse.

Namespace: Euresys::Open_eVision

```
[C++]
void Ellipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle containing the ellipse

orgY
Y origin of the rectangle containing the ellipse

width
Width of the rectangle containing the ellipse

height
Height of the rectangle containing the ellipse

pen
Optional pen to use

EWindowsDrawAdapter::EWindowsDrawAdapter

Creates an EWindowsDrawAdapter either by providing a drawing context or an image. If a drawing context is provided EWindowsDrawAdapter will forward all drawing commands to that drawing context. If an image is provided a drawing context will be initialized on that image and that drawing context will be closed either on destruction of the EWindowsDrawAdapter or when the EWindowsDrawAdapter::Close method is called.

Namespace: Euresys::Open_eVision

[C++]

```
void EWindowsDrawAdapter(
 HDC dc
 )

void EWindowsDrawAdapter(
 EImageBW8* pImage
 )

void EWindowsDrawAdapter(
 EImageC24* pImage
 )
```

Parameters

dc

The Windows API drawning context.

pImage

-

EWindowsDrawAdapter::FilledEllipse

Fills an ellipse.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledEllipse(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle containing the ellipse

orgY
Y origin of the rectangle containing the ellipse

width
Width of the rectangle containing the ellipse

height
Height of the rectangle containing the ellipse

pen
Optional pen to use for drawing the contour of the ellipse

brush
Optional pen to use for drawing the inside of the ellipse

EWindowsDrawAdapter::FilledPolygon

Fills a polygon.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledPolygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen,
 EBrush brush
 )
```

Parameters

points
Points of the polygon

pen
Optional pen to use for drawing the contour of the polygon

brush
Optional pen to use for drawing the inside of the polygon

EWindowsDrawAdapter::FilledRectangle

Fills a rectangle.

Namespace: Euresys::Open_eVision

[C++]

```
void FilledRectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen,
 EBrush brush
 )
```

Parameters

orgX
X origin of the rectangle

orgY
Y origin of the rectangle

width
Width of the rectangle

height
Height of the rectangle

pen
Optional pen to use for drawing the contour of the rectangle

brush
Optional brush to use for filling the inside of the rectangle

EWindowsDrawAdapter::SetFont

Default font.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFont(const EFont& font)
```

EWindowsDrawAdapter::GetTextSize

Size of the given text using the default font (EWindowsDrawAdapter::Font).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetTextSize(
 const std::string& text
 )
```

Parameters

text

Text

EWindowsDrawAdapter::GetHDC

Gets the HDC associated with this instance of EWindowsDrawAdapter.

Namespace: Euresys::Open_eVision

```
[C++]
HDC GetHDC()
```

EWindowsDrawAdapter::Image

Draws an image.

Namespace: Euresys::Open_eVision

```
[C++]
void Image(
 const EBaseROI& image,
 float orgX,
 float orgY,
 float width,
 float height
 )
void Image(
 const EROIBW8& image,
 EC24Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
void Image(
 const EROIBW8& image,
 EBW8Vector* pColorScale,
 float orgX,
 float orgY,
 float width,
 float height
 )
```

Parameters

image
Image.
orgX
X coordinate of the point where to draw the image.
orgY
Y coordinate of the point where to draw the image.
width
Width of the destination rectangle in which to draw the image.
height
Height of the destination rectangle in which to draw the image.
pColorScale
Color scale to draw a grayscale image with.

EWindowsDrawAdapter::Line

Draws a line between two points.

Namespace: Euresys::Open_eVision

[C++]

```
void Line(
 int x1,
 int y1,
 int x2,
 int y2,
 EPen pen
 )
```

Parameters

x1
X coordinate of line origin point
y1
Y coordinate of line origin point
x2
X coordinate of line end point
y2
Y coordinate of line end point
pen
Optional pen to use

EWindowsDrawAdapter::SetPen

Default pen.

Namespace: Euresys::Open_eVision

```
[C++]
void SetPen(const EPen& pen)
```

EWindowsDrawAdapter::Polygon

Draws a polygon.

Namespace: Euresys::Open_eVision

```
[C++]
void Polygon(
 const std::vector<Euresys::Open_eVision::EPoint>& points,
 EPen pen
 )
```

Parameters

points

Points of the polygon

pen

Optional pen to use

EWindowsDrawAdapter::Rectangle

Draws a rectangle.

Namespace: Euresys::Open_eVision

```
[C++]
void Rectangle(
 int orgX,
 int orgY,
 int width,
 int height,
 EPen pen
 )
```

Parameters

orgX
X origin of the rectangle
orgY
Y origin of the rectangle
width
Width of the rectangle
height
Height of the rectangle
pen
Optional pen to use

EWindowsDrawAdapter::Text

Draws a text.

Namespace: Euresys::Open_eVision

[C++]

```
void Text(
 const std::string& text,
 int x,
 int y,
 float orientation,
 EBrush textBrush
 )
```

Parameters

text
Text to draw.
x
X position of the text.
y
Y position of the text.
orientation
Orientation of the text in radians.
textBrush
Optional brush to use for the color of the text.

EWindowsDrawAdapter::UseCurrentBrush

Use the current pen set in the drawing framework.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentBrush(
 )
```

EWindowsDrawAdapter::UseCurrentPen

Uses the current GDI pen.

Namespace: Euresys::Open_eVision

[C++]

```
void UseCurrentPen(
 )
```

4.257. EWorldShape Class

Manages a complete context for calibrating a field of view.

Base Class:EShape

Namespace: Euresys::Open_eVision

Methods

AddLandmark	Adds a new pair of points coordinates (in Sensor and World spaces) to the set of landmarks used for calibration.
AddPoint	Adds a new point coordinates (in Sensor space) to the set of grid points used for calibration.
AutoCalibrate	Returns the best calibration modes for the current calibration grid and calibrates the field of view accordingly.
AutoCalibrateDotGrid	Performs an automatic calibration based on a dot grid image.
AutoCalibrateLandmarks	Returns the best calibration modes for the current landmark set and calibrates the field of view accordingly.
Calibrate	Performs a calibration according to the specified combination of calibration modes.
CalibrationSucceeded	Getter method for the CalibrationSucceeded property. This property is the flag indicating if the calibration has succeeded (true), that is whether the mean variation of grid points (distance between computed grid points and ideal grid points in world space) and the maximum variation of grid points are between given tolerances.
Closest	Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.
DisableTypeFilter	Enables all shape types

Drag	Moves a handle to a new position and updates the position parameters of the shape.
DragLandmark	Moves the landmark to a new position.
Draw	Draws the world coordinate axis.
DrawCrossGrid	Draws a regular grid of crosses in world coordinates.
DrawCrossGridWithCurrentPen	Draws a regular grid of crosses in world coordinates.
DrawGrid	Draws the reconstructed grid to be used for grid calibration.
DrawGridWithCurrentPen	Draws the reconstructed grid to be used for grid calibration.
DrawLandmarks	Draws the landmarks to be used for landmark calibration.
DrawWithCurrentPen	Draws the world coordinate axis.
EmptyLandmarks	Resets the landmark specification sequence.
EnableTypeFilter	Enables the filter of the specified shape type
EWorldShape	Constructs a EWorldShape object.
GetAngle	Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.
GetCalibrationModes	Current calibration mode, made from a combination of values.
GetCenter	Position of the origin point of the reference frame as projected onto the image, i.e. the Sensor position of the point at World coordinates (0,0).
GetCenterX	Horizontal position of the origin point of the reference frame as projected onto the image, i.e. the Sensor abscissa of the point at World coordinates (0,0).
GetCenterY	Vertical position of the origin point of the reference frame as projected onto the image, i.e. the Sensor ordinate of the point at World coordinates (0,0).
GetDistortionStrength	Sets the optical distortion parameters
GetFieldHeight	Field-of-view height, in physical units.
GetFieldWidth	Field-of-view width, in physical units.
GetGridPointsMaxVariation	Maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to EWorldShape::CalibrationSucceeded
GetGridPointsMaxVariationThreshold	Grid points maximum variation threshold, that is the value above which the maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded.

GetGridPointsMeanVariation	Mean variation of grid points (mean distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to EWorldShape::CalibrationSucceeded.
GetGridPointsMeanVariationThreshold	Grid points mean variation threshold, that is the value above which the mean variation of grid points (mean distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded.
GetHitLandmark	Returns the landmark selected by EWorldShape::HitLandmarks or ~0 if no landmark is selected.
GetLandmarkElement	Returns the landmark element corresponding to the given index.
GetNumLandmarkElements	Returns the number of landmark elements.
GetPanX	Current horizontal panning value for drawing operations, expressed in pixels. By default, no panning occurs.
GetPanY	Current vertical panning value for drawing operations, expressed in pixels. By default, no panning occurs.
GetPerspectiveStrength	Perspective effect coefficient, that is the inverse of the observation distance.
GetRatio	XResolution/YResolution ratio.
GetScale	The scale of the reference frame.
GetSensorHeight	Logical image height, that is the number of pixels vertically.
GetSensorWidth	Logical image width, that is the number of pixels horizontally.
GetTiltXAngle	Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.
GetTiltYAngle	Tilt Y angle, that is the amplitude of the rotation applied around the Y-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.
GetType	Shape type.
GetXResolution	Horizontal sensor resolution, in pixels per unit.
GetYResolution	Vertical sensor resolution, in pixels per unit.
GetZoomX	Current horizontal zooming factor for drawing operations.
GetZoomY	Current vertical zooming factor for drawing operations.
HitLandmarks	Checks if the cursor is placed over a landmark point.
HitTest	Checks if there is a handle under the cursor.
operator=	Copies all the data from another EWorldShape object into the current EWorldShape object
RebuildGrid	Reconstructs the grid of points from the given dot centers, to compute the World coordinates of the points.
RemoveLandmark	Removes a landmark.

SensorToWorld	Performs coordinate transform for arbitrary points from Sensor space to World space.
SetAngle	Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.
SetCalibrationModes	Current calibration mode, made from a combination of values.
SetCenter	Position of the origin point of the reference frame as projected onto the image, i.e. the Sensor position of the point at World coordinates (0,0).
SetCenterXY	Position of the origin point of the reference frame as projected onto the image, i.e. the Sensor position of the point at World coordinates (0,0).
SetDistortion	Sets the optical distortion parameters
SetDistortionStrength	Sets the optical distortion parameters
SetFieldSize	Sets the field of view size in physical units.
SetGridPointsMaxVariationThreshold	Grid points maximum variation threshold, that is the value above which the maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded.
SetGridPointsMeanVariationThreshold	Grid points mean variation threshold, that is the value above which the mean variation of grid points (mean distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded.
SetPan	Sets the horizontal and vertical panning values for drawing operations, expressed in pixels.
SetPerspective	Sets the perspective effect coefficient, i.e. the inverse of the observation distance.
SetRatio	XResolution/YResolution ratio.
SetResolution	Sets the sensor resolution in pixels per unit in both directions.
SetScale	The scale of the reference frame.
SetSensor	Initializes the calibration object using all given parameters.
SetSensorSize	Sets the logical image size, i.e. the number of pixels horizontally and vertically.
SetSize	Sets the frame size.
SetupUnwarp	Prepares a lookup table for fast image unwarping.
SetZoom	Sets the horizontal and vertical zooming factors for drawing operations.
Unwarp	Unwarps a distorted image using the current calibration model.
WorldToSensor	Performs coordinate transform for arbitrary points from World space to Sensor space.

EWorldShape::AddLandmark

Adds a new pair of points coordinates (in Sensor and World spaces) to the set of landmarks used for calibration.

Namespace: Euresys::Open_eVision

```
[C++]
void AddLandmark(
 const EPoint& sensorPoint,
 const EPoint& worldPoint
 )
```

Parameters

sensorPoint

Sensor point coordinates.

worldPoint

Corresponding World point coordinates.

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known.

EWorldShape::AddPoint

Adds a new point coordinates (in Sensor space) to the set of grid points used for calibration.

Namespace: Euresys::Open_eVision

```
[C++]
void AddPoint(
 const EPoint& sensorPoint
 )
```

Parameters

sensorPoint

Sensor point coordinates.

Remarks

Grid calibration is the process of computing the calibration parameters by means of a set of points known to lie on a rectangular grid. If the grid pitch is known and one of the points is chosen as the origin point, the points can be used as landmarks. By contrast with the landmark calibration functions, the World coordinates of the grid points need not be specified, nor do they have to be given in any specific order. The calibration algorithm is capable of sorting out the points to reconstruct the grid topology. Typically, this function is used in conjunction with blob analysis to extract the dot centers from a grid of dots. Anyway, any other scheme can be used. The grid of points need not be complete, i.e. some of the nodes may be missing, and the points need not completely fill a rectangular area. Landmark calibration is simply achieved by providing a series of point coordinates (in Sensor space only) and then calling the grid reconstruction function followed by the calibration function.

EWorldShape::GetAngle

EWorldShape::SetAngle

Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

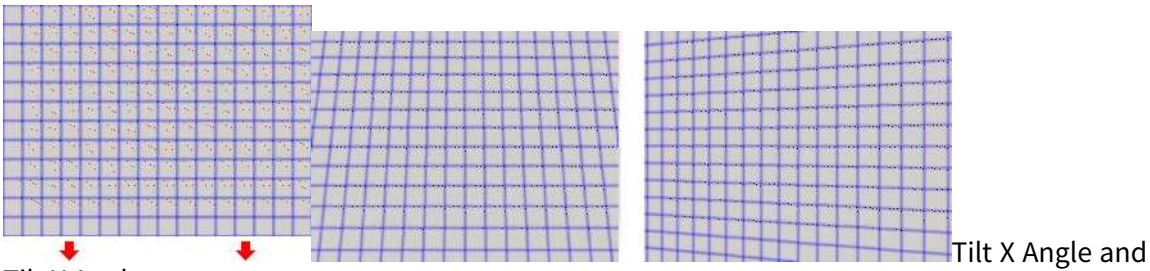
Namespace: Euresys::Open_eVision

[C++]

```
float GetAngle() const

void SetAngle(float f32Angle)
```

Remarks



Tilt Y Angle

Tilt X Angle and

EWorldShape::AutoCalibrate

Returns the best calibration modes for the current calibration grid and calibrates the field of view accordingly.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 AutoCalibrate(
 bool testEmpiricalModes
 )
```

Parameters

testEmpiricalModes

Boolean indicating whether empirical calibration modes (ECalibrationMode_Quadratic and ECalibrationMode_Bilinear) should be considered when determining the best calibration modes.

Remarks

To ensure a successful calibration, the perspective angle of the view should not exceed 45 degrees.

EWorldShape::AutoCalibrateDotGrid

Performs an automatic calibration based on a dot grid image.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 AutoCalibrateDotGrid(
 EROIBW8* sourceImage,
 float columnPitch,
 float rowPitch,
 bool testEmpiricalModes
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

columnPitch

Actual pitches of the grid, i.e. distances between vertical and horizontal rows of the grid.

rowPitch

Actual pitches of the grid, i.e. distances between vertical and horizontal rows of the grid.

testEmpiricalModes

Boolean indicating whether empirical calibration modes (ECalibrationMode_Quadratic and ECalibrationMode_Bilinear) should be considered when determining the best calibration modes. Default value is false.

Remarks

Returns the best calibration mode for the current dot grid. The EWorldShape::AutoCalibrateDotGrid method will first do an automatic blob analysis in order to extract all dots (all blobs whose area is smaller than 5 pixels will be considered as noise and rejected). The dot gravity centers are used as the grid reference points. Then, the EWorldShape::AutoCalibrateDotGrid method will select and compute the best calibration mode by reducing the fitting error. To ensure a successful calibration, the perspective angle of the dot grid should not exceed 45 degrees.

EWorldShape::AutoCalibrateLandmarks

Returns the best calibration modes for the current landmark set and calibrates the field of view accordingly.

Namespace: Euresys::Open_eVision

```
[C++]

OEV_UINT32 AutoCalibrateLandmarks(
 bool testEmpiricalModes
 )
```

Parameters

testEmpiricalModes

Boolean indicating whether empirical calibration modes (ECalibrationMode_Quadratic and ECalibrationMode_Bilinear) should be considered when determining the best calibration modes. Default value is false.

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known. To ensure a successful calibration, the perspective angle of the view should not exceed 45 degrees. The EWorldShape::AutoCalibrateLandmarks method is meant to be used with landmark calibration only. To calibrate automatically your field of view using a dot grid, use the EWorldShape::AutoCalibrate method instead.

EWorldShape::Calibrate

Performs a calibration according to the specified combination of calibration modes.

Namespace: Euresys::Open_eVision

```
[C++]

void Calibrate(
 OEV_UINT32 calibrationModes
 )
```

Parameters

calibrationModes

Calibration modes, as defined by a combination of values from ECalibrationMode.

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known. In some cases, not all requested calibration modes are honored. After calibration, EWorldShape::CalibrationModes returns the actual combination of modes in effect. To ensure a successful calibration, the perspective angle of the view should not exceed 45 degrees.

EWorldShape::GetCalibrationModes

EWorldShape::SetCalibrationModes

Current calibration mode, made from a combination of values.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetCalibrationModes()

void SetCalibrationModes(OEV_UINT32 calibrationModes)
```

Remarks

The supported calibration modes can be set to ECalibrationMode_Raw, meaning that no calibration at all is performed (the World coordinates are pixel indices), or to the logical sum of other values from ECalibrationMode.

EWorldShape::CalibrationSucceeded

Getter method for the CalibrationSucceeded property. This property is the flag indicating if the calibration has succeeded (true), that is whether the mean variation of grid points (distance between computed grid points and ideal grid points in world space) and the maximum variation of grid points are between given tolerances.

Namespace: Euresys::Open_eVision

[C++]

```
bool CalibrationSucceeded(
 )
```

Remarks

The mean and maximum grid point variations are normalized using the pitch values. By default, tolerances are set to 0.05 (5 %) for the mean, and 0.1 (10 %) for the maximum grid point variation. You can get and set these tolerances using the EWorldShape::GridPointsMeanVariationThreshold and EWorldShape::GridPointsMaxVariationThreshold properties.

EWorldShape::GetCenter

EWorldShape::SetCenter

Position of the origin point of the reference frame as projected onto the image, i.e. the Sensor position of the point at World coordinates (0,0).

Namespace: Euresys::Open_eVision

[C++]

```
EPoint GetCenter() const

void SetCenter(const EPoint& center)
```

EWorldShape::GetCenterX

Horizontal position of the origin point of the reference frame as projected onto the image, i.e. the Sensor abscissa of the point at World coordinates (0,0).

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterX() const
```

EWorldShape::GetCenterY

Vertical position of the origin point of the reference frame as projected onto the image, i.e. the Sensor ordinate of the point at World coordinates (0,0).

Namespace: Euresys::Open_eVision

[C++]

```
float GetCenterY() const
```

EWorldShape::Closest

Find the daughter shape that is the closest to this shape. To retrieve the closest shape, use EShape::ClosestShape.

Namespace: Euresys::Open_eVision

[C++]

```
void Closest(
 )
```

EWorldShape::DisableTypeFilter

Enables all shape types

Namespace: Euresys::Open_eVision

[C++]

```
void DisableTypeFilter(
 )
```

EWorldShape::SetDistortion

This property is deprecated.

Sets the optical distortion parameters

Namespace: Euresys::Open_eVision

[C++]

```
void SetDistortion(float distortionStrength)
```

Remarks

Deprecated in favor of EWorldShape::DistortionStrength.

EWorldShape::GetDistortionStrength

EWorldShape::SetDistortionStrength

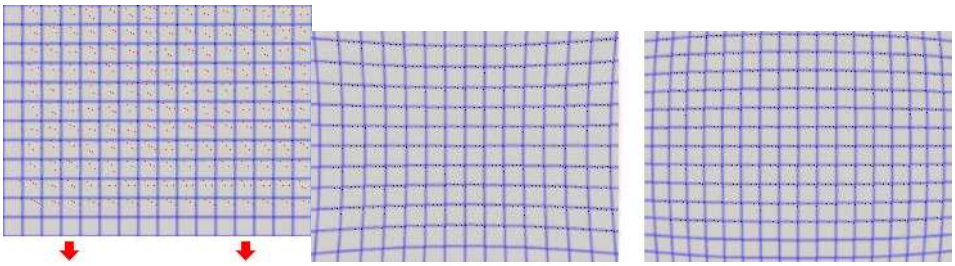
Sets the optical distortion parameters

Namespace: Euresys::Open_eVision

[C++]

```
float GetDistortionStrength()
void SetDistortionStrength(float distortionStrength)
```

Remarks



<caption>Positive distortion and negative distortion</caption>

EWorldShape::Drag

Moves a handle to a new position and updates the position parameters of the shape.

Namespace: Euresys::Open_eVision

[C++]

```
void Drag(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

EWorldShape::DragLandmark

Moves the landmark to a new position.

Namespace: Euresys::Open_eVision

[C++]

```
void DragLandmark(
 int n32CursorX,
 int n32CursorY
 )
```

Parameters

n32CursorX

Current cursor coordinates.

n32CursorY

Current cursor coordinates.

EWorldShape::Draw

Draws the world coordinate axis.

Namespace: Euresys::Open_eVision

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingModes,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingModes,
 bool daughters
 )

void Draw(
 HDC graphicContext,
 const ERGBColor& color,
 Euresys::Open_eVision::EDrawingMode drawingModes,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingModes

Indicates how the world coordinate axis must be displayed, as defined by EDrawingMode.

daughters

Indicates whether the daughter shapes are to be displayed as well.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::DrawCrossGrid

Draws a regular grid of crosses in world coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawCrossGrid(
 EDrawAdapter* graphicContext,
 float minimumX,
 float maximumX,
 float minimumY,
 float maximumY,
 OEV_UINT32 numberOfIntervalsX,
 OEV_UINT32 numberOfIntervalsY
 )
void DrawCrossGrid(
 HDC graphicContext,
 float minimumX,
 float maximumX,
 float minimumY,
 float maximumY,
 OEV_UINT32 numberOfIntervalsX,
 OEV_UINT32 numberOfIntervalsY
 )
void DrawCrossGrid(
 HDC graphicContext,
 const ERGBColor& color,
 float minimumX,
 float maximumX,
 float minimumY,
 float maximumY,
 OEV_UINT32 numberOfIntervalsX,
 OEV_UINT32 numberOfIntervalsY
 )
```

Parameters

graphicContext
Handle of the device context on which to draw.

minimumX
Abscissa of the leftmost crosses, in world coordinates.

maximumX
Abscissa of the rightmost crosses, in world coordinates.

minimumY
Ordinate of the leftmost crosses, in world coordinates.

maximumY
Ordinate of the rightmost crosses, in world coordinates.

numberOfIntervalsX

Number of intervals between crosses along the horizontal direction.

numberOfIntervalsY

Number of intervals between crosses along the vertical direction.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::DrawCrossGridWithCurrentPen

This method is deprecated.

Draws a regular grid of crosses in world coordinates.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawCrossGridWithCurrentPen(
 HDC graphicContext,
 float minimumX,
 float maximumX,
 float minimumY,
 float maximumY,
 OEV_UINT32 numberOfIntervalsX,
 OEV_UINT32 numberOfIntervalsY
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

minimumX

Abscissa of the leftmost crosses, in world coordinates.

maximumX

Abscissa of the rightmost crosses, in world coordinates.

minimumY

Ordinate of the leftmost crosses, in world coordinates.

maximumY

Ordinate of the rightmost crosses, in world coordinates.

numberOfIntervalsX

Number of intervals between crosses along the horizontal direction.

numberOfIntervalsY

Number of intervals between crosses along the vertical direction.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::DrawGrid

Draws the reconstructed grid to be used for grid calibration.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawGrid(
 EDrawAdapter* graphicContext
 )
void DrawGrid(
 HDC graphicContext
 )
void DrawGrid(
 HDC graphicContext,
 const ERGBColor& color
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

color

The color in which to draw the overlay.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::DrawGridWithCurrentPen

This method is deprecated.

Draws the reconstructed grid to be used for grid calibration.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawGridWithCurrentPen(
 HDC graphicContext
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::DrawLandmarks

Draws the landmarks to be used for landmark calibration.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawLandmarks(
 EDrawAdapter* graphicContext
 )
void DrawLandmarks(
 HDC graphicContext
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::DrawWithCurrentPen

This method is deprecated.

Draws the world coordinate axis.

Namespace: Euresys::Open_eVision

```
[C++]
void DrawWithCurrentPen(
 HDC graphicContext,
 Euresys::Open_eVision::EDrawingMode drawingModes,
 bool daughters
 )
```

Parameters

graphicContext

Handle of the device context on which to draw.

drawingModes

Indicates how the world coordinate axis must be displayed, as defined by EDrawingMode.

daughters

Indicates whether the daughter shapes are to be displayed as well.

Remarks

Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EWorldShape::EmptyLandmarks

Resets the landmark specification sequence.

Namespace: Euresys::Open_eVision

[C++]

```
void EmptyLandmarks(
 )
```

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known.

EWorldShape::EnableTypeFilter

Enables the filter of the specified shape type

Namespace: Euresys::Open_eVision

[C++]

```
void EnableTypeFilter(
 OEV_UINT32 un32Types
 )
```

Parameters

un32Types

The type of the shape to filter from EShapeType.

EWorldShape::EWorldShape

Constructs a EWorldShape object.

Namespace: Euresys::Open_eVision

[C++]

```
void EWorldShape(
 const EWorldShape& other
 )
```

```
void EWorldShape(
 )
```

Parameters

other

Another EWorldShape object to be copied in the new EWorldShape object.

EWorldShape::GetFieldHeight

Field-of-view height, in physical units.

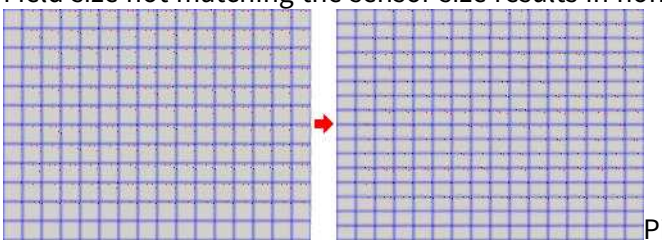
Namespace: Euresys::Open_eVision

[C++]

```
float GetFieldHeight() const
```

Remarks

Field size not matching the sensor size results in non-square pixels.

 Pixels having non-square aspect ratioBeware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (XResolution/YResolution) should be in the range [-4/3, -3/4] (or [3/4, 4/3]), otherwise the calibration process could fail.

EWorldShape::GetFieldWidth

Field-of-view width, in physical units.

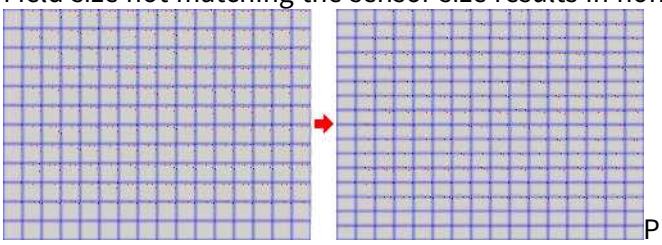
Namespace: Euresys::Open_eVision

[C++]

```
float GetFieldWidth() const
```

Remarks

Field size not matching the sensor size results in non-square pixels.

 Pixels having non-square aspect ratioBeware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (XResolution/YResolution) should be in the range [-4/3, -3/4] (or [3/4, 4/3]), otherwise the calibration process could fail.

EWorldShape::GetLandmarkElement

Returns the landmark element corresponding to the given index.

Namespace: Euresys::Open_eVision

```
[C++]

ELandmark& GetLandmarkElement(
 OEV_UINT32 i
 )

const ELandmark& GetLandmarkElement(
 OEV_UINT32 i
 )
```

Parameters

i

Landmark index.

EWorldShape::GetGridPointsMaxVariation

Maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to EWorldShape::CalibrationSucceeded

Namespace: Euresys::Open_eVision

```
[C++]

float GetGridPointsMaxVariation()
```

Remarks

The maximum grid point variation is normalized using the pitch values.

EWorldShape::GetGridPointsMaxVariationThreshold

EWorldShape::SetGridPointsMaxVariationThreshold

Grid points maximum variation threshold, that is the value above which the maximum variation of grid points (maximum distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded.

Namespace: Euresys::Open_eVision

[C++]

```
float GetGridPointsMaxVariationThreshold()
void SetGridPointsMaxVariationThreshold(float maxVariationThreshold)
```

Remarks

The maximum grid point variation is normalized using the pitch values.

EWorldShape::GetGridPointsMeanVariation

Mean variation of grid points (mean distance between computed grid points and ideal grid points in world space), normalized using the pitch values, after a call to EWorldShape::CalibrationSucceeded.

Namespace: Euresys::Open_eVision

[C++]

```
float GetGridPointsMeanVariation()
```

Remarks

The mean grid point variation is normalized using the pitch values.

EWorldShape::GetGridPointsMeanVariationThreshold

EWorldShape::SetGridPointsMeanVariationThreshold

Grid points mean variation threshold, that is the value above which the mean variation of grid points (mean distance between computed grid points and ideal grid points in world space) is considered too high, and thus marks the calibration as a failure when using EWorldShape::CalibrationSucceeded.

Namespace: Euresys::Open_eVision

[C++]

```
float GetGridPointsMeanVariationThreshold()
```

```
void SetGridPointsMeanVariationThreshold(float meanVariationThreshold)
```

Remarks

The mean grid point variation is normalized using the pitch values.

EWorldShape::GetHitLandmark

Returns the landmark selected by EWorldShape::HitLandmarks or ~0 if no landmark is selected.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetHitLandmark()
```

EWorldShape::HitLandmarks

Checks if the cursor is placed over a landmark point.

Namespace: Euresys::Open_eVision

[C++]

```
void HitLandmarks(
 )
```

Remarks

Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known.

EWorldShape::HitTest

Checks if there is a handle under the cursor.

Namespace: Euresys::Open_eVision

[C++]

```
bool HitTest(
 bool bDaughters
 )
```

Parameters

bDaughters

Indicates if the check must be done in the whole hierarchy or just this object.

EWorldShape::GetNumLandmarkElements

Returns the number of landmark elements.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 GetNumLandmarkElements() const
```

EWorldShape::operator=

Copies all the data from another EWorldShape object into the current EWorldShape object

Namespace: Euresys::Open_eVision

[C++]

```
EWorldShape& operator=(
 const EWorldShape& other
 )
```

Parameters

other

EWorldShape object to be copied

EWorldShape::GetPanX

Current horizontal panning value for drawing operations, expressed in pixels. By default, no panning occurs.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPanX()
```

EWorldShape::GetPanY

Current vertical panning value for drawing operations, expressed in pixels. By default, no panning occurs.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPanY()
```

EWorldShape::GetPerspectiveStrength

Perspective effect coefficient, that is the inverse of the observation distance.

Namespace: Euresys::Open_eVision

[C++]

```
float GetPerspectiveStrength()
```

Remarks

The larger this parameter, the more perceivable the perspective distortion will be. A NULL value corresponds to a telecentric lens.

EWorldShape::GetRatio

EWorldShape::SetRatio

XResolution/YResolution ratio.

Namespace: Euresys::Open_eVision

[C++]

```
float GetRatio()

void SetRatio(float ratio)
```

Remarks

If Ratio equals -1 (or 1), pixels are square. Beware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (XResolution/YResolution) should be in the range [-4/3, -3/4] (or [3/4, 4/3]), otherwise the calibration process could fail.

EWorldShape::RebuildGrid

Reconstructs the grid of points from the given dot centers, to compute the World coordinates of the points.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 RebuildGrid(
 float colPitch,
 float rowPitch,
 OEV_UINT32 centerIndex,
 bool direct
 )
OEV_UINT32 RebuildGrid(
 float colPitch,
 float rowPitch,
 const EPoint& worldCenter,
 OEV_UINT32 centerIndex,
 bool direct
 )
```

Parameters

colPitch

Actual pitches of the grid, that are distances between vertical and horizontal rows of the grid.

rowPitch

Actual pitches of the grid, that are distances between vertical and horizontal rows of the grid.

centerIndex

Index of the grid point chosen as coordinate origin point. By default, the most central grid point.

direct

true if the world reference frame points upwards.

worldCenter

World coordinates of the starting grid point.

Remarks

This member function also returns the number of grid points that were connected. This prepares the calibration using landmarks (for use by member EWorldShape::Calibrate). Landmark calibration is the process of computing the calibration parameters by means of a set of known points for which the coordinates are available in both World and Sensor spaces. Usually, such points are chosen as salient features on the part or target in view. They must be such that appropriate image processing techniques allow measuring their positions from the image (directly or indirectly by geometric constructions), while at the same time their coordinates in a reference frame are known. See also Dot-Grid-Based Calibration for the grid construction algorithm.

EWorldShape::RemoveLandmark

Removes a landmark.

Namespace: Euresys::Open_eVision

```
[C++]
void RemoveLandmark(
 OEV_UINT32 index
 )
```

Parameters

index

Index of the landmark to be removed.

EWorldShape::GetScale

EWorldShape::SetScale

The scale of the reference frame.

Namespace: Euresys::Open_eVision

```
[C++]
float GetScale() const
void SetScale(float f32Scale)
```

EWorldShape::GetSensorHeight

Logical image height, that is the number of pixels vertically.

Namespace: Euresys::Open_eVision

```
[C++]
int GetSensorHeight() const
```

EWorldShape::SensorToWorld

Performs coordinate transform for arbitrary points from Sensor space to World space.

Namespace: Euresys::Open_eVision

```
[C++]
EPoint SensorToWorld(
 const EPoint& sensorPoint
 )
```

Parameters

sensorPoint

Sensor point.

EWorldShape::GetSensorWidth

Logical image width, that is the number of pixels horizontally.

Namespace: Euresys::Open_eVision

[C++]

```
int GetSensorWidth() const
```

EWorldShape::SetCenterXY

Position of the origin point of the reference frame as projected onto the image, i.e. the Sensor position of the point at World coordinates (0,0).

Namespace: Euresys::Open_eVision

[C++]

```
void SetCenterXY(
 float centerX,
 float centerY
 )
```

Parameters

centerX

Horizontal position (abscissa)

centerY

Vertical position (ordinate)

EWorldShape::SetFieldSize

Sets the field of view size in physical units.

Namespace: Euresys::Open_eVision

[C++]

```
void SetFieldSize(
 float width,
 float height
 )
```

Parameters

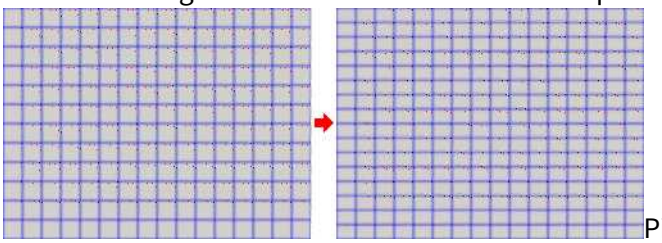
width

Full image physical width, in length units.

height

Full image physical height, in length units. If not specified, same as physical width.

Remarks

Field size not matching the sensor size results in non-square pixels. By default, the pixels are square.  Pixels having non-square aspect ratioBeware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio (XResolution/YResolution) should be in the range [-4/3, -3/4] (or [3/4, 4/3]), otherwise the calibration process could fail.

EWorldShape::SetPan

Sets the horizontal and vertical panning values for drawing operations, expressed in pixels.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPan(
 float panX,
 float panY
 )
```

Parameters

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

Remarks

All objects attached to an EWorldShape object inherit of the same panning factor.

EWorldShape::SetPerspective

Sets the perspective effect coefficient, i.e. the inverse of the observation distance.

Namespace: Euresys::Open_eVision

[C++]

```
void SetPerspective(
 float tiltXAngle,
 float tiltYAngle,
 float perspectiveStrength
 )
```

Parameters

tiltXAngle

Tilt angles, i.e. the amplitudes of the rotations applied around the X and Y axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

tiltYAngle

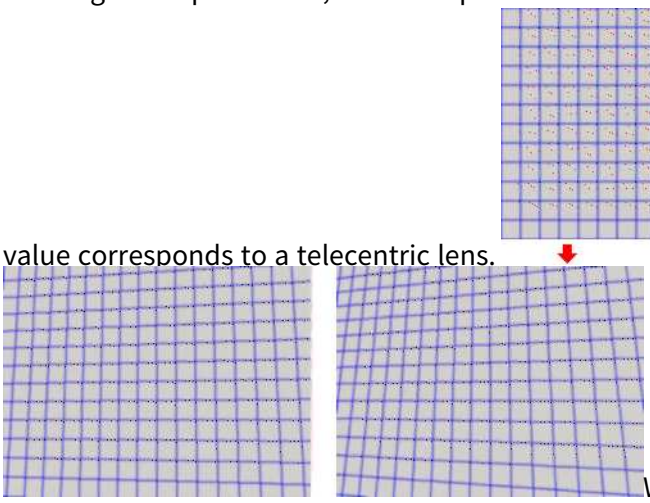
Tilt angles, i.e. the amplitudes of the rotations applied around the X and Y axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

perspectiveStrength

Perspective effect coefficient.

Remarks

The larger this parameter, the more perceivable the perspective distortion will be. A NULL value corresponds to a telecentric lens.



Weak Perspective and Strong Perspective

EWorldShape::SetResolution

Sets the sensor resolution in pixels per unit in both directions.

Namespace: Euresys::Open_eVision

[C++]

```
void SetResolution(
 float resolutionX,
 float resolutionY
 )
```

Parameters

resolutionX

Horizontal resolution in pixels per units

resolutionY

Vertical resolution in pixels per units. If not specified, same as horizontal resolution.

Remarks

By default, the pixels are square.

EWorldShape::SetSensor

Initializes the calibration object using all given parameters.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSensor(
 int sensorWidth,
 int sensorHeight,
 float fieldWidth,
 float fieldHeight,
 float centerX,
 float centerY,
 float angle,
 float tiltXAngle,
 float tiltYAngle,
 float perspectiveStrength,
 float distortionStrength,
 float opticalCenterX,
 float opticalCenterY,
 OEV_UINT32 calibrationModes
 )
```

Parameters

sensorWidth

Logical size of the field of view, i.e. image size, in pixels.

sensorHeight

Logical size of the field of view, i.e. image size, in pixels.

fieldWidth

Physical size of the field of view. By default (argument omitted), the pixels are square.

fieldHeight

Physical size of the field of view. By default (argument omitted), the pixels are square.

centerX

Position of the "intersection" between the optical axis and the field of view in the image. By default, if the calibration modes contain ECalibrationMode_Raw, it is set to 0. Otherwise, it is set to the image center.

centerY

Position of the "intersection" between the optical axis and the field of view in the image. By default, if the calibration modes contain ECalibrationMode_Raw, it is set to 0 (or to the bottommost pixel index if the calibration modes also contain ECalibrationMode_Inverse). Otherwise, it is set to the image center.

angle

Skew angle, i.e. angle formed by the axis of reference and the image edges. By default (argument omitted), no skewing effect is assumed.

tiltXAngle

Rotation angles on the X axis to bring the optical axis perpendicular to the image plane. By default (argument omitted), no perspective effect is assumed.

tiltYAngle

Rotation angles on the Y axis to bring the optical axis perpendicular to the image plane. By default (argument omitted), no perspective effect is assumed.

perspectiveStrength

Relative importance of the perspective effect. By default, no perspective effect is assumed, as if the lens was telecentric.

distortionStrength

Relative importance of the lens radial distortion. Positive for barrel, negative for cushion. By default (argument omitted), no optical distortion is assumed.

opticalCenterX

X Position of the "intersection" between the optical axis and the field of view in the image. By default (argument omitted) the image center.

opticalCenterY

Y Position of the "intersection" between the optical axis and the field of view in the image. By default (argument omitted) the image center.

calibrationModes

Desired calibration mode effects to be combined, as defined by ECalibrationMode. By default (argument omitted), the simplest model compatible with the given parameters is chosen.

Remarks

The function automatically selects the appropriate calibration model by checking the parameters. The use of a more complex calibration mode can be enforced by means of parameter EWorldShape::CalibrationModes, not a simpler one.

EWorldShape::SetSensorSize

Sets the logical image size, i.e. the number of pixels horizontally and vertically.

Namespace: Euresys::Open_eVision

[C++]

```
void SetSensorSize(
 int width,
 int height
 )
```

Parameters

width

Full image logical sizes, in pixels.

height

Full image logical sizes, in pixels.

EWorldShape::SetSize

Sets the frame size.

Namespace: Euresys::Open_eVision

```
[C++]
void SetSize(
 float sizeX,
 float sizeY
 )
```

Parameters

sizeX

Frame X-axis length. The default value is 100.

sizeY

Frame Y-axis length. By default, both axes have the same length.

Remarks

By default, both frame axis value are set to 100, which means 100 pixels when the field of view is not calibrated and 100 "units" in case of a calibrated field of view.

EWorldShape::SetupUnwarp

Prepares a lookup table for fast image unwarping.

Namespace: Euresys::Open_eVision

```
[C++]
void SetupUnwarp(
 EUnwarpingLut* lookupTable,
 EROIBW8* sourceImage,
 bool interpolate
 )
void SetupUnwarp(
 EUnwarpingLut* lookupTable,
 EROIC24* sourceImage,
 bool interpolate
 )
```

Parameters

lookupTable

Pointer to the lookup table.

sourceImage

Pointer to the source image/ROI.

interpolate

Interpolation mode. Default value is false.

Remarks

The function should be called each time the system is re-calibrated (after the optical setup has been changed, for instance). A sample source image has to be supplied to EWorldShape::SetupUnwarp, and its row pitch is recorded in order to speedup the unwarping process. This implies that the following calls to EWorldShape::Unwarp are not allowed to use images with row pitches different from the source image initially supplied to EWorldShape::SetupUnwarp.

EWorldShape::SetZoom

Sets the horizontal and vertical zooming factors for drawing operations.

Namespace: Euresys::Open_eVision

[C++]

```
void SetZoom(
 float zoomX,
 float zoomY
 )
```

Parameters

zoomX

Horizontal zooming factor. By default, true scale is used.

zoomY

Vertical zooming factor. If set to 0, the default value, the horizontal zooming factor is used instead, so as to provide isotropic zooming.

Remarks

All objects attached to an EWorldShape inherit of the same zooming factor.

EWorldShape::GetTiltXAngle

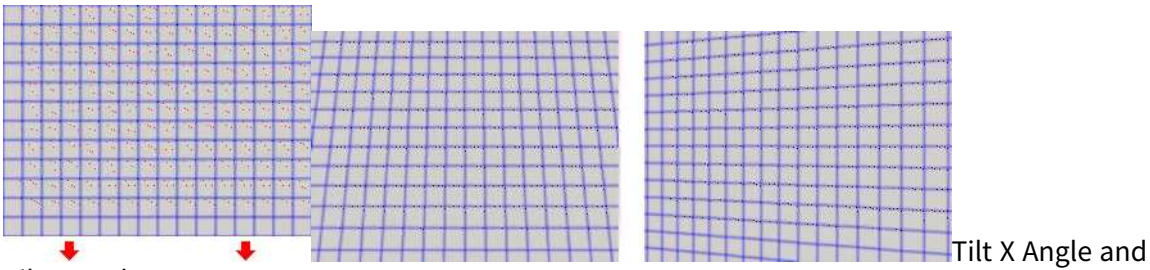
Tilt X angle, that is the amplitude of the rotation applied around the X-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

Namespace: Euresys::Open_eVision

[C++]

```
float GetTiltXAngle()
```

Remarks



Tilt Y Angle

Tilt X Angle and

EWorldShape::GetTiltYAngle

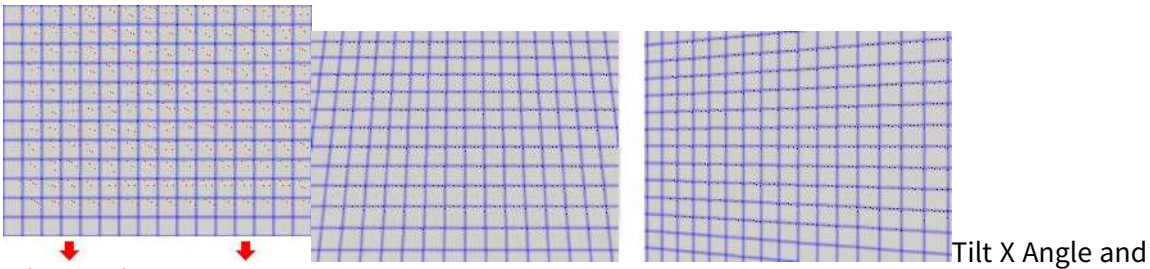
Tilt Y angle, that is the amplitude of the rotation applied around the Y-axis of the sensor to bring the optical (Z) axis perpendicular to the field of view.

Namespace: Euresys::Open_eVision

[C++]

```
float GetTiltYAngle()
```

Remarks



Tilt Y Angle

Tilt X Angle and

EWorldShape::GetType

Shape type.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EShapeType GetType()
```

EWorldShape::Unwarp

Unwarps a distorted image using the current calibration model.

Namespace: Euresys::Open_eVision

[C++]

```
void Unwarp(
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 bool interpolate
 )

void Unwarp(
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 bool interpolate
 )

void Unwarp(
 EUnwarpingLut* lookupTable,
 EROIBW8* sourceImage,
 EROIBW8* destinationImage,
 bool interpolate
 )

void Unwarp(
 EUnwarpingLut* lookupTable,
 EROIC24* sourceImage,
 EROIC24* destinationImage,
 bool interpolate
 )
```

Parameters

sourceImage

Pointer to the source image/ROI.

destinationImage

Pointer to the destination unwarped image.

interpolate

Interpolation mode. Default value is false.

lookupTable

Pointer to the lookup table.

Remarks

Using a precomputed lookup table allows speeding up the unwarping process. The lookup table is initialized by means of the EWorldShape::SetupUnwarp function.

EWorldShape::WorldToSensor

Performs coordinate transform for arbitrary points from World space to Sensor space.

Namespace: Euresys::Open_eVision

[C++]

```
EPoint WorldToSensor(
 const EPoint& worldPoint
 )
```

Parameters

worldPoint

World point.

EWorldShape::GetXResolution

Horizontal sensor resolution, in pixels per unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetXResolution()
```

EWorldShape::GetYResolution

Vertical sensor resolution, in pixels per unit.

Namespace: Euresys::Open_eVision

[C++]

```
float GetYResolution()
```

EWorldShape::GetZoomX

Current horizontal zooming factor for drawing operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetZoomX()
```

EWorldShape::GetZoomY

Current vertical zooming factor for drawing operations.

Namespace: Euresys::Open_eVision

[C++]

```
float GetZoomY()
```

4.258. EZMap Class

Represents a generic ZMap type interface.

Derived Class(es):EZMap16EZMap32fEZMap8

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. If the metadata key already exists, its value will be overwritten.
Clear	Clears the ZMap: replaces all pixels with the undefined value.
Create	Factory method to allocates and reads a ZMap from a file. Depending of the serialized EZMap type, it returns the corresponding EZMap8, EZMap16 or EZMap32f object. The allocated EZMap must be released after use.
Draw	Draws an EZMap in a device context.
DrawImage	Displays the internal image buffer
GetBufferPtr	Retrieves the pointer to the internal pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer to the pixel buffer.
GetHeight	Access ZMap Height.
GetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
GetMetadata	Returns the string value of the given metadata. Throws an exception if it does not exist.
GetResolution	Gets the resolution of the EZMap along the X, Y and Z axis. On the Z axis, the resolution is the number of metric units per grey value. On the X and Y axis, the resolution is the number of metric units per pixel.
GetRowPitch	Returns the buffer row pitch.
GetSizeInWorld	Returns the dimensions of the EZMap in real world space (e.g metric unit).
GetType	Pixel accessor type.
GetWidth	Access ZMap Width.

GetWorldPositionFromMapPosition	Returns the 3D world position corresponding to a EZMap 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap, otherwise an exception will be thrown.
GetWorldPositionFromPixelPosition	Returns the 3D world position corresponding to a EZMap pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.
GetWorldShape	Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap in real space coordinates (e.g mm).
GetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
GetXResolution	Resolution of the EZMap along the X axis The resolution is the number of metric units per pixel.
GetYResolution	Resolution of the EZMap along the Y axis The resolution is the number of metric units per pixel.
GetZMapPositionFromPixelPosition	Returns the corresponding EZMap 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.
GetZResolution	Resolution of the EZMap along the Z axis The resolution is the number of metric units per grey value.
ImageToWorld	Transforms a floating point (sub)pixel image position to a 3D world position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The pixel Z value is in grey scale values (its range depends on the ZMap type).
ImageToZMap	Converts a 2D image (sub)pixel coordinate to the EZMap space. (u,v) is the pixel position (with its origin in the upper left corner of the image). (x,y) is the corresponding ZMap position (which has the same scale as the world space). All values are expressed in floating point numbers.
IsVoid	Tests if the EZMap object size is zero.
Load	Restores the EZMap stored in the given Open eVision file.
LoadImage	Restores the EZMap image stored in the given image file.
LoadImageAndMetadata	Loads image format and Metadata in JSON format.

LoadMetadata	Loads Metadata in JSON format.
ResetWorldTransformation	Reset the world transformation, Map to World and World to Map matrices become identity matrix.
Save	Saves the EZMap object to the given Open eVision file.
SaveImage	Saves the EZMap image to the given image file.
SaveImageAndMetadata	Saves image format and Metadata JSON format.
SaveMetadata	Saves Metadata in JSON format.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.
SetHeight	Access ZMap Height.
SetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
SetResolution	Sets the resolution of the EZMap along the X, Y and Z axis. On the Z axis, the resolution is the number of metric units per grey value. On the X and Y axis, the resolution is the number of metric units per pixel.
SetSize	Sets the width and height of the EZMap.
SetWidth	Access ZMap Width.
SetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
SetXResolution	Resolution of the EZMap along the X axis The resolution is the number of metric units per pixel.
SetYResolution	Resolution of the EZMap along the Y axis The resolution is the number of metric units per pixel.
SetZResolution	Resolution of the EZMap along the Z axis The resolution is the number of metric units per grey value.
WorldToImage	Transforms a 3D world position to a floating point (sub)pixel image position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The Z value is given in grey scale value. Returns true if the pixel position is inside the image limits and the Z value is positive. The parameter pixelPt is filled even if the point is outside the image.
WorldToZMap	Transforms a 3D world position to a 3D EZMap position. The ZMap space origin is at the lower left corner of the image. The scales of the world space and ZMap space are the same.

ZMapToImage	Converts a 2D coordinate in the EZMap space to image (sub)pixel space. (x,y) is the ZMap position (which has the same scale as the world space). (u,v) is the corresponding pixel position (with its origin in the upper left corner of the image). All values are expressed in floating point numbers. Returns true if the pixel position is inside the image limits.
ZMapToWorld	Transforms a 3D EZMap position to a 3D world space position. The ZMap space origin is at the lower left corner of the image.

EZMap::AddMetadata

Adds a metadata key (name) and value.
If the metadata key already exists, its value will be overwritten.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EZMap::Clear

Clears the ZMap: replaces all pixels with the undefined value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Clear(
 )
```

EZMap::Create

Factory method to allocates and reads a ZMap from a file. Depending of the serialized EZMap type, it returns the corresponding EZMap8, EZMap16 or EZMap32f object. The allocated EZMap must be released after use.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
std::unique_ptr<EZMap> Create(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EZMap::Draw

Draws an EZMap in a device context.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
```

```
void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the ZMap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A ZMap can be drawn (its pixels rendered) using a device context. EZMap::Draw and EZMap::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context. EZMap::Draw and EZMap::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap::GetBufferPtr

Retrieves the pointer to the internal pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )

void* GetBufferPtr(
 int x,
 int y
 )

const void* GetBufferPtr(
 )

const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel which we want the address.

y

Row of the pixel which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EZMap::GetCheckedBufferPtr

Retrieves the pointer to the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )

const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

EZMap::GetMetadata

Returns the string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EZMap::GetResolution

Gets the resolution of the EZMap along the X, Y and Z axis.
On the Z axis, the resolution is the number of metric units per grey value.
On the X and Y axis, the resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetResolution(
 )
```

```
void GetResolution(
 float& sx,
 float& sy,
 float& sz
 )
```

Parameters

sx
Contains the resolution along the X axis.

sy
Contains the resolution along the Y axis.

sz
Contains the resolution along the Z axis.

EZMap::GetSizeInWorld

Returns the dimensions of the EZMap in real world space (e.g metric unit).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetSizeInWorld(
 float& worldWidth,
 float& worldHeight
 )
```

Parameters

worldWidth
Contains the size of the ZMap along the X axis (column).

worldHeight
Contains the size of the ZMap along the Y axis (row).

EZMap::GetWorldPositionFromMapPosition

Returns the 3D world position corresponding to a EZMap 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromMapPosition(
 float x,
 float y
 )
```

Parameters

x

The X coordinate.

y

The Y coordinate.

EZMap::GetWorldPositionFromPixelPosition

Returns the 3D world position corresponding to a EZMap pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap::GetZMapPositionFromPixelPosition

Returns the corresponding EZMap 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetZMapPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap::GetHeight

EZMap::SetHeight

Access ZMap Height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetHeight() const
void SetHeight(int height)
```

EZMap::ImageToWorld

Transforms a floating point (sub)pixel image position to a 3D world position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The pixel Z value is in grey scale values (its range depends on the ZMap type).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToWorld(
 const E3DPoint& pixelPt,
 E3DPoint& worldPt
 )
```

Parameters

pixelPt

Position in the image space.

worldPt

Position in the 3D world space.

EZMap::ImageToZMap

Converts a 2D image (sub)pixel coordinate to the EZMap space.
(u,v) is the pixel position (with its origin in the upper left corner of the image).
(x,y) is the corresponding ZMap position (which has the same scale as the world space).
All values are expressed in floating point numbers.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToZMap(
 float u,
 float v,
 float& x,
 float& y
 )
```

Parameters

u

X Coordinate of the pixel as a floating point value.

v

Y Coordinate of the pixel as a floating point value.

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

EZMap::IsVoid

Tests if the EZMap object size is zero.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsVoid(
 )
```

Remarks

Returns true if the ZMap size is zero.

EZMap::Load

Restores the EZMap stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path to the file.

serializer

-

Remarks

When loading, the ZMap is resized if needed. This function restores all the ZMap attributes.

EZMap::LoadImage

Restores the EZMap image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the ZMap is resized if needed. This function does not restore the ZMap attributes, only the image associated with the EZMap is updated.

EZMap::LoadImageAndMetadata

Loads image format and Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata
 )
```

Parameters

pathImage

Full path to the file.

pathMetadata

Full path to the file.

EZMap::LoadMetadata

Loads Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EZMap::GetMapToWorldMatrix

EZMap::SetMapToWorldMatrix

E3DTransformMatrix that transforms positions from the EZMap space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetMapToWorldMatrix() const

void SetMapToWorldMatrix(const E3DTransformMatrix& matrix)
```

EZMap::ResetWorldTransformation

Reset the world transformation, Map to World and World to Map matrices become identity matrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ResetWorldTransformation(
 )
```

EZMap::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetRowPitch() const
```

EZMap::Save

Saves the EZMap object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This format save the EZMap in a Open eVision file. This function stores all the ZMap attributes.

EZMap::SaveImage

Saves the EZMap image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This format save the image associated to EZMap in a standard image file and thus does not store ZMap attributes.

EZMap::SaveImageAndMetadata

Saves image format and Metadata JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

pathImage

The full path to the destination file.

pathMetadata

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

EZMap::SaveMetadata

Saves Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EZMap::SetBufferPtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the data.

bitsPerRow

The total number of bits contained in a row, padding included.
Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EZMap::SetBufferPtr.

EZMap::SetResolution

Sets the resolution of the EZMap along the X, Y and Z axis.
On the Z axis, the resolution is the number of metric units per grey value.
On the X and Y axis, the resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetResolution(
 E3DPoint resolution
 )

void SetResolution(
 float rx,
 float ry,
 float rz
 )
```

Parameters

resolution
Contains the resolution along the X,Y and Z axis.

rx
Contains the resolution along the X axis.

ry
Contains the resolution along the Y axis.

rz
Contains the resolution along the Z axis.

EZMap::SetSize

Sets the width and height of the EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetSize(
 int width,
 int height
 )

void SetSize(
 const EZMap& other
 )
```

Parameters

width

The new requested width.

height

The new requested height.

other

The other ZMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an ZMap* is specified as a number of columns (width) and rows (height).
The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries.

EZMap::GetType

Pixel accessor type.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::EImageType GetType() const
```

EZMap::GetWidth

EZMap::SetWidth

Access ZMap Width.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetWidth() const
void SetWidth(int width)
```

EZMap::GetWorldShape

Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap in real space coordinates (e.g mm).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EWorldShape& GetWorldShape() const
```

EZMap::WorldToImage

Transforms a 3D world position to a floating point (sub)pixel image position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The Z value is given in grey scale value.
Returns true if the pixel position is inside the image limits and the Z value is positive.
The parameter pixelPt is filled even if the point is outside the image.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool WorldToImage(
 const E3DPoint& worldPt,
 E3DPoint& pixelPt
 )
```

Parameters

worldPt

Position in the 3D world space.

pixelPt

Position in the image space.

EZMap::GetWorldToMapMatrix

EZMap::SetWorldToMapMatrix

Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const E3DTransformMatrix& GetWorldToMapMatrix() const

void SetWorldToMapMatrix(const E3DTransformMatrix& matrix)
```

EZMap::WorldToZMap

Transforms a 3D world position to a 3D EZMap position.
The ZMap space origin is at the lower left corner of the image.
The scales of the world space and ZMap space are the same.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void WorldToZMap(
 const E3DPoint& worldPt,
 E3DPoint& zmapPt
 )
```

Parameters

worldPt

Position in the 3D world space.

zmapPt

Position in the ZMap space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap::GetXResolution

EZMap::SetXResolution

Resolution of the EZMap along the X axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetXResolution() const
void SetXResolution(float resolution)
```

EZMap::GetYResolution

EZMap::SetYResolution

Resolution of the EZMap along the Y axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetYResolution() const
void SetYResolution(float resolution)
```

EZMap::ZMapToImage

Converts a 2D coordinate in the EZMap space to image (sub)pixel space.
(x,y) is the ZMap position (which has the same scale as the world space).
(u,v) is the corresponding pixel position (with its origin in the upper left corner of the image).
All values are expressed in floating point numbers.
Returns true if the pixel position is inside the image limits.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ZMapToImage(
 float x,
 float y,
 float& u,
 float& v
 )
```

Parameters

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

u

Column of the pixel as a floating point value.

v

Row of the pixel as a floating point value.

EZMap::ZMapToWorld

Transforms a 3D EZMap position to a 3D world space position.
The ZMap space origin is at the lower left corner of the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ZMapToWorld(
 const E3DPoint& zmapPt,
 E3DPoint& worldPt
 )
```

Parameters

zmapPt

Position in the ZMap space.

worldPt

Position in the 3D world space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap::GetZResolution

EZMap::SetZResolution

Resolution of the EZMap along the Z axis
The resolution is the number of metric units per grey value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.259. EZMap16 Class

A ZMap16 is a 16bits corrected 2.5D image.
ZMap Pixel values (16 bits integers) represent distances from a 3D reference plane.
Distances are positive, during the ZMap generation all points below the reference plane are discarded.
The EZMap class also stores the transformation from the pixel coordinates to the real world coordinate system.
There could be undefined pixels in the ZMap.

Base Class:EZMap

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. If the metadata key already exists, its value will be overwritten.
AsEImage	Returns the EZMap16 as an EImageBW16 (16 bits gray scale) to use with existing eVision 2D tools.
Clear	Clears the ZMap: replaces all pixels with the undefined value.
ClearMetadata	Deletes all metadata.

ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Buffer coordinates
ConvertCoordinatesPixelToMap	Converts Buffer coordinates to 3D Map coordinates
CopyMetadataTo	Copies all metadata.
DeleteMetadata	Deletes value of this existing metadata key . Throws an exception if it does not exist.
Draw	Draws an EZMap16 in a device context.
DrawImage	Displays the internal image buffer
EZMap16	Creates a 16 bits EZMap.
FillUndefinedPixels	Fills undefined pixels, used to fill the "holes" in the ZMap.
FillUndefinedPixelsWithMedian	Fills undefined pixels, used to fill the "holes" in the ZMap using a median rectangular kernel of odd size.
GetBufferPtr	Retrieves the pointer to the internal pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer to the pixel buffer.
GetHeight	Access ZMap Height.
GetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap16 space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
GetMetadata	Returns the string value of the given metadata. Throws an exception if it does not exist.
GetPixel	Gets the value of a pixel .
GetPixelPositionFromWorldPosition	Returns in the u, v and value parameters the EZMap16 values corresponding to a 3D world position. The world position is projected on the ZMap reference plane to get a position in the ZMap. If the projected position is outside the ZMap, the method returns false.
GetResolution	Gets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.
GetRowPitch	Returns the buffer row pitch.
GetSizeInWorld	Returns the dimensions of the EZMap16 in real world space (e.g metric unit).
GetType	Pixel accessor type.
GetUndefinedValue	Returns the Undefined value. That value is used to set to mark pixels with no valid depth value.
GetWidth	Access ZMap Width.

GetWorldPositionFromMapPosition	Returns the 3D world position corresponding to a EZMap16 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap16, otherwise an exception will be thrown.
GetWorldPositionFromPixelPosition	Returns the 3D world position corresponding to a EZMap16 pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.
GetWorldShape	Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap16 in real space coordinates (e.g mm).
GetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap16 space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
GetXResolution	Resolution of the EZMap16 along the X axis The resolution is the number of metric units per pixel.
GetYResolution	Resolution of the EZMap16 along the Y axis The resolution is the number of metric units per pixel.
GetZMapPositionFromPixelPosition	Returns the corresponding EZMap16 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.
GetZRange	Compute the minimum and maximum pixel values, excluding the undefined pixels.
GetZResolution	Resolution of the EZMap16 along the Z axis The resolution is the number of metric units per grey value.
GetZValue	Gets Z value (in metric coordinate) at pixel coordinates.
ImageToWorld	Transforms a floating point (sub)pixel image position to a 3D world position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The pixel Z value is in grey scale values (its range depends on the ZMap type).
ImageToZMap	Converts a 2D image (sub)pixel coordinate to the EZMap16 space. (u,v) is the pixel position (with its origin in the upper left corner of the image). (x,y) is the corresponding ZMap position (which has the same scale as the world space). All values are expressed in floating point numbers.
IsVoid	Tests if the EZMap16 object size is zero.
Load	Restores the EZMap16 stored in the given Open eVision file.

LoadImage	Restores the EZMap16 image stored in the given image file.
LoadImageAndMetadata	Loads image format and Metadata in JSON format.
LoadMetadata	Loads Metadata in JSON format.
ModifyMetadata	Changes an existing metadata key and value. Throws an exception if it does not exist.
operator=	Assignment operator.
ResetWorldTransformation	Reset the world transformation, Map to World and World to Map matrices become identity matrix.
Save	Saves the EZMap16 object to the given Open eVision file.
SaveImage	Saves the EZMap16 image to the given image file.
SaveImageAndMetadata	Saves image format and Metadata JSON format.
SaveMetadata	Saves Metadata in JSON format.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.
SetHeight	Access ZMap Height.
SetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap16 space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
SetPixel	Sets the value of a pixel .
SetResolution	Sets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.
SetSize	Sets the width and height of the EZMap16.
SetWidth	Access ZMap Width.
SetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap16 space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
SetXResolution	Resolution of the EZMap16 along the X axis The resolution is the number of metric units per pixel.
SetYResolution	Resolution of the EZMap16 along the Y axis The resolution is the number of metric units per pixel.
SetZResolution	Resolution of the EZMap16 along the Z axis The resolution is the number of metric units per grey value.
SetZValue	Sets Z value (in metric coordinate) at pixel coordinates.

WorldToImage	Transforms a 3D world position to a floating point (sub)pixel image position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The Z value is given in grey scale value. Returns true if the pixel position is inside the image limits and the Z value is positive. The parameter pixelPt is filled even if the point is outside the image.
WorldToZMap	Transforms a 3D world position to a 3D EZMap16 position. The ZMap space origin is at the lower left corner of the image. The scales of the world space and ZMap space are the same.
ZMapToImage	Converts a 2D coordinate in the EZMap16 space to image (sub)pixel space. (x,y) is the ZMap position (which has the same scale as the world space). (u,v) is the corresponding pixel position (with its origin in the upper left corner of the image). All values are expressed in floating point numbers. Returns true if the pixel position is inside the image limits.
ZMapToWorld	Transforms a 3D EZMap16 position to a 3D world space position. The ZMap space origin is at the lower left corner of the image.

EZMap16::AddMetadata

Adds a metadata key (name) and value.
If the metadata key already exists, its value will be overwritten.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EZMap16::AsEImage

Returns the EZMap16 as an EImageBW16 (16 bits gray scale) to use with existing eVision 2D tools.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EImageBW16& AsEImage(
 )

const EImageBW16& AsEImage(
 )
```

EZMap16::Clear

Clears the ZMap: replaces all pixels with the undefined value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Clear(
 )
```

EZMap16::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ClearMetadata(
 )
```

EZMap16::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Buffer coordinates

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D
The Map X coordinate.

y3D
The Map Y coordinate.

xBuffer
The returned Pixel X coordinate.

yBuffer
The returned Pixel Y coordinate.

EZMap16::ConvertCoordinatesPixelToMap

Converts Buffer coordinates to 3D Map coordinates

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EZMap16::CopyMetadataTo

Copies all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void CopyMetadataTo(
 EZMap16& other
 )
```

Parameters

other

An other EZMap16.

EZMap16::DeleteMetadata

Deletes value of this existing metadata key . Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key

-

EZMap16::Draw

Draws an EZMap16 in a device context.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the ZMap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A ZMap can be drawn (its pixels rendered) using a device context. EZMap16::Draw and EZMap16::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap16::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context. EZMap16::Draw and EZMap16::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap16::EZMap16

Creates a 16 bits EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EZMap16(
 )

void EZMap16(
 int width,
 int height
 )

void EZMap16(
 const EZMap16& other
 )
```

Parameters

width

The width of the new ZMap.

height

The height of the new ZMap.

other

Another ZMap.

EZMap16::FillUndefinedPixels

Fills undefined pixels, used to fill the "holes" in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillUndefinedPixels(
 EZMap16& outMap,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

outMap

The destination ZMap.

direction

Direction in which the undefined pixels are filled in a ZMap from EFillUndefinedPixelsDirection.

method

Which values used to fill the undefined pixels in a ZMap from EFillUndefinedPixelsMethod.

EZMap16::FillUndefinedPixelsWithMedian

Fills undefined pixels, used to fill the "holes" in the ZMap using a median rectangular kernel of odd size.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillUndefinedPixelsWithMedian(
 EZMap16& outMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

outMap

The destination ZMap.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 2).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth).

EZMap16::GetBufferPtr

Retrieves the pointer to the internal pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )

void* GetBufferPtr(
 int x,
 int y
 )

const void* GetBufferPtr(
 )

const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel which we want the address.

y

Row of the pixel which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EZMap16::GetCheckedBufferPtr

Retrieves the pointer to the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )

const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

EZMap16::GetMetadata

Returns the string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EZMap16::GetPixel

Gets the value of a pixel .

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepth16 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EZMap16::GetPixelPositionFromWorldPosition

Returns in the u, v and value parameters the EZMap16 values corresponding to a 3D world position.
The world position is projected on the ZMap reference plane to get a position in the ZMap.
If the projected position is outside the ZMap, the method returns false.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetPixelPositionFromWorldPosition(
 const E3DPoint& world_position,
 int& u,
 int& v,
 EDepth16& value
 )
```

Parameters

world_position
The 3D coordinates of a world position.

u
Column of the ZMap pixel in [0,width[.

v
Row of the ZMap pixel in [0,height[.

value
Value of the pixel.

EZMap16::GetResolution

Gets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetResolution(
 float& sx,
 float& sy,
 float& sz
 )
E3DPoint GetResolution(
 )
```

Parameters

sx
Resolution along the X axis.

sy
Resolution along the Y axis.

sz
Resolution along the Z axis.

EZMap16::GetSizeInWorld

Returns the dimensions of the EZMap16 in real world space (e.g metric unit).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetSizeInWorld(
 float& worldWidth,
 float& worldHeight
 )
```

Parameters

worldWidth

Contains the size of the ZMap along the X axis (column).

worldHeight

Contains the size of the ZMap along the Y axis (row).

EZMap16::GetWorldPositionFromMapPosition

Returns the 3D world position corresponding to a EZMap16 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap16, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetWorldPositionFromMapPosition(
 float x,
 float y
 )
```

Parameters

x

The X coordinate.

y

The Y coordinate.

EZMap16::GetWorldPositionFromPixelPosition

Returns the 3D world position corresponding to a EZMap16 pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap16::GetZMapPositionFromPixelPosition

Returns the corresponding EZMap16 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetZMapPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap16::GetZRange

Compute the minimum and maximum pixel values, excluding the undefined pixels.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetZRange(
 EBW16& min,
 EBW16& max
 )
```

Parameters

min
The lowest pixel value.

max
The highest pixel value.

EZMap16::GetZValue

Gets Z value (in metric coordinate) at pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x
X Coordinate.

y
Y Coordinate.

EZMap16::GetHeight

EZMap16::SetHeight

Access ZMap Height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetHeight() const

void SetHeight(int height)
```

EZMap16::ImageToWorld

Transforms a floating point (sub)pixel image position to a 3D world position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The pixel Z value is in grey scale values (its range depends on the ZMap type).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToWorld(
 const E3DPoint& pixelPt,
 E3DPoint& worldPt
 )
```

Parameters

pixelPt

Position in the image space.

worldPt

Position in the 3D world space.

EZMap16::ImageToZMap

Converts a 2D image (sub)pixel coordinate to the EZMap16 space.
(u,v) is the pixel position (with its origin in the upper left corner of the image).
(x,y) is the corresponding ZMap position (which has the same scale as the world space).
All values are expressed in floating point numbers.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToZMap(
 float u,
 float v,
 float& x,
 float& y
 )
```

Parameters

u

X Coordinate of the pixel as a floating point value.

v

Y Coordinate of the pixel as a floating point value.

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

EZMap16::IsVoid

Tests if the EZMap16 object size is zero.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsVoid(
 )
```

Remarks

Returns true if the ZMap size is zero.

EZMap16::Load

Restores the EZMap16 stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path to the file.

serializer

-

Remarks

When loading, the ZMap is resized if needed. This function restores all the ZMap attributes.

EZMap16::LoadImage

Restores the EZMap16 image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the ZMap is resized if needed. This function does not restore the ZMap attributes, only the image associated with the EZMap16 is updated.

EZMap16::LoadImageAndMetadata

Loads image format and Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImageAndMetadata(
 const std::string& path,
 const std::string& pathMetadata
 )
```

Parameters

path

-

pathMetadata

Full path to the file.

EZMap16::LoadMetadata

Loads Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EZMap16::GetMapToWorldMatrix

EZMap16::SetMapToWorldMatrix

E3DTransformMatrix that transforms positions from the EZMap16 space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetMapToWorldMatrix() const

void SetMapToWorldMatrix(const E3DTransformMatrix& matrix)
```

EZMap16::ModifyMetadata

Changes an existing metadata key and value. Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

-

value

-

EZMap16::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EZMap16& operator=(
 const EZMap16& other
 )
```

Parameters

other

The source EZMap16.

EZMap16::ResetWorldTransformation

Reset the world transformation, Map to World and World to Map matrices become identity matrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ResetWorldTransformation(
 )
```

EZMap16::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetRowPitch() const
```

EZMap16::Save

Saves the EZMap16 object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This format save the EZMap16 in a Open eVision file. This function stores all the ZMap attributes.

EZMap16::SaveImage

Saves the EZMap16 image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This format save the image associated to EZMap16 in a standard image file and thus does not store ZMap attributes.

EZMap16::SaveImageAndMetadata

Saves image format and Metadata JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& path,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

path

-

pathMetadata

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

EZMap16::SaveMetadata

Saves Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EZMap16::SetBufferPtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the data.

bitsPerRow

The total number of bits contained in a row, padding included.
Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EZMap16::SetBufferPtr.

EZMap16::SetPixel

Sets the value of a pixel .

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetPixel(
 EDepth16 value,
 int x,
 int y
 )
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EZMap16::SetResolution

Sets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetResolution(
 float rx,
 float ry,
 float rz
 )

void SetResolution(
 E3DPoint resolution
 )
```

Parameters

rx
Resolution along the X axis.

ry
Resolution along the Y axis.

rz
Resolution along the Z axis.

resolution
Resolution for X,Y and Z axis.

EZMap16::SetSize

Sets the width and height of the EZMap16.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetSize(
 int width,
 int height
 )

void SetSize(
 const EZMap& other
 )
```

Parameters

width

The new requested width.

height

The new requested height.

other

The other ZMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an ZMap* is specified as a number of columns (width) and rows (height).
The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries.

EZMap16::SetZValue

Sets Z value (in metric coordinate) at pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetZValue(
 const float value,
 const int x,
 const int y
 )
```

Parameters

value

Value of the pixel in metric space.

x

X Coordinate.

y

Y Coordinate.

EZMap16::GetType

Pixel accessor type.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::EImageType GetType() const
```

EZMap16::GetUndefinedValue

Returns the Undefined value. That value is used to set to mark pixels with no valid depth value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth16 GetUndefinedValue() const
```

EZMap16::GetWidth

EZMap16::SetWidth

Access ZMap Width.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetWidth() const
void SetWidth(int width)
```

EZMap16::GetWorldShape

Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap16 in real space coordinates (e.g mm).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EWorldShape& GetWorldShape() const
```

EZMap16::WorldToImage

Transforms a 3D world position to a floating point (sub)pixel image position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The Z value is given in grey scale value.
Returns true if the pixel position is inside the image limits and the Z value is positive.
The parameter pixelPt is filled even if the point is outside the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool WorldToImage(
 const E3DPoint& worldPt,
 E3DPoint& pixelPt
 )
```

Parameters

worldPt

Position in the 3D world space.

pixelPt

Position in the image space.

EZMap16::GetWorldToMapMatrix

EZMap16::SetWorldToMapMatrix

Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap16 space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetWorldToMapMatrix() const

void SetWorldToMapMatrix(const E3DTransformMatrix& matrix)
```

EZMap16::WorldToZMap

Transforms a 3D world position to a 3D EZMap16 position.
The ZMap space origin is at the lower left corner of the image.
The scales of the world space and ZMap space are the same.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void WorldToZMap(
 const E3DPoint& worldPt,
 E3DPoint& zmapPt
 )
```

Parameters

worldPt

Position in the 3D world space.

zmapPt

Position in the ZMap space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap16::GetXResolution

EZMap16::SetXResolution

Resolution of the EZMap16 along the X axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetXResolution() const

void SetXResolution(float resolution)
```

EZMap16::GetYResolution

EZMap16::SetYResolution

Resolution of the EZMap16 along the Y axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetYResolution() const

void SetYResolution(float resolution)
```

EZMap16::ZMapToImage

Converts a 2D coordinate in the EZMap16 space to image (sub)pixel space.
(x,y) is the ZMap position (which has the same scale as the world space).
(u,v) is the corresponding pixel position (with its origin in the upper left corner of the image).
All values are expressed in floating point numbers.
Returns true if the pixel position is inside the image limits.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ZMapToImage(
 float x,
 float y,
 float& u,
 float& v
 )
```

Parameters

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

u

Column of the pixel as a floating point value.

v

Row of the pixel as a floating point value.

EZMap16::ZMapToWorld

Transforms a 3D EZMap16 position to a 3D world space position.
The ZMap space origin is at the lower left corner of the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ZMapToWorld(
 const E3DPoint& zmapPt,
 E3DPoint& worldPt
 )
```

Parameters

zmapPt

Position in the ZMap space.

worldPt

Position in the 3D world space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap16::GetZResolution

EZMap16::SetZResolution

Resolution of the EZMap16 along the Z axis
The resolution is the number of metric units per grey value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.260. EZMap32f Class

A ZMap32f is a 32bits corrected 2.5D image.
ZMap pixel values (32 bits floating point numbers) represent distances from a 3D reference plane.
Distances are positive, during the ZMap generation all points below the reference plane are discarded.
The EZMap class also stores the transformation from the pixel coordinates to the real world coordinate system.
There could be undefined pixels in the ZMap.

Base Class:EZMap

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. If the metadata key already exists, its value will be overwritten.
AsEImage	Returns the EZMap32f as an EImageBW32 (32 bits gray scale) to use with existing eVision 2D tools.
Clear	Clears the ZMap: replaces all pixels with the undefined value.
ClearMetadata	Deletes all metadata.

ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Buffer coordinates
ConvertCoordinatesPixelToMap	Converts Buffer coordinates to 3D Map coordinates
CopyMetadataTo	Copies all metadata.
DeleteMetadata	Deletes value of this existing metadata key . Throws an exception if it does not exist.
Draw	Draws an EZMap32f in a device context.
DrawImage	Displays the internal image buffer
EZMap32f	Creates a 32 bits EZMap.
FillUndefinedPixels	Fills undefined pixels, used to fill the "holes" in the ZMap.
FillUndefinedPixelsWithMedian	Fills undefined pixels, used to fill the "holes" in the ZMap using a median rectangular kernel of odd size.
GetBufferPtr	Retrieves the pointer to the internal pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer to the pixel buffer.
GetHeight	Access ZMap Height.
GetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap32f space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
GetMetadata	Returns the string value of the given metadata. Throws an exception if it does not exist.
GetPixel	Gets the value of a pixel .
GetPixelPositionFromWorldPosition	Returns in the u, v and value parameters the EZMap32f values corresponding to a 3D world position. The world position is projected on the ZMap reference plane to get a position in the ZMap. If the projected position is outside the ZMap, the method returns false.
GetResolution	Gets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.
GetRowPitch	Returns the buffer row pitch.
GetSizeInWorld	Returns the dimensions of the EZMap32f in real world space (e.g metric unit).
GetType	Pixel accessor type.
GetUndefinedValue	Returns the Undefined value. That value is used to set to mark pixels with no valid depth value.
GetWidth	Access ZMap Width.

GetWorldPositionFromMapPosition	Returns the 3D world position corresponding to a EZMap32f 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap32f, otherwise an exception will be thrown.
GetWorldPositionFromPixelPosition	Returns the 3D world position corresponding to a EZMap32f pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.
GetWorldShape	Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap32f in real space coordinates (e.g mm).
GetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap32f space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
GetXResolution	Resolution of the EZMap32f along the X axis The resolution is the number of metric units per pixel.
GetYResolution	Resolution of the EZMap32f along the Y axis The resolution is the number of metric units per pixel.
GetZMapPositionFromPixelPosition	Returns the corresponding EZMap32f 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.
GetZRange	Compute the minimum and maximum pixel values, excluding the undefined pixels.
GetZResolution	Resolution of the EZMap32f along the Z axis The resolution is the number of metric units per grey value.
GetZValue	Gets Z value (in metric coordinate) at pixel coordinates.
ImageToWorld	Transforms a floating point (sub)pixel image position to a 3D world position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The pixel Z value is in grey scale values (its range depends on the ZMap type).
ImageToZMap	Converts a 2D image (sub)pixel coordinate to the EZMap32f space. (u,v) is the pixel position (with its origin in the upper left corner of the image). (x,y) is the corresponding ZMap position (which has the same scale as the world space). All values are expressed in floating point numbers.
IsVoid	Tests if the EZMap32f object size is zero.

Load	Restores the EZMap32f stored in the given Open eVision file.
LoadImage	Restores the EZMap32f image stored in the given image file.
LoadImageAndMetadata	Loads image format and Metadata in JSON format.
LoadMetadata	Loads Metadata in JSON format.
ModifyMetadata	Changes an existing metadata key and value. Throws an exception if it does not exist.
operator=	Assignment operator.
ResetWorldTransformation	Reset the world transformation, Map to World and World to Map matrices become identity matrix.
Save	Saves the EZMap32f object to the given Open eVision file.
SaveImage	Saves the EZMap32f image to the given image file.
SaveImageAndMetadata	Saves image format and Metadata JSON format.
SaveMetadata	Saves Metadata in JSON format.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.
SetHeight	Access ZMap Height.
SetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap32f space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
SetPixel	Sets the value of a pixel .
SetResolution	Sets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.
SetSize	Sets the width and height of the EZMap32f.
SetWidth	Access ZMap Width.
SetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap32f space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
SetXResolution	Resolution of the EZMap32f along the X axis The resolution is the number of metric units per pixel.
SetYResolution	Resolution of the EZMap32f along the Y axis The resolution is the number of metric units per pixel.
SetZResolution	Resolution of the EZMap32f along the Z axis The resolution is the number of metric units per grey value.
SetZValue	Sets Z value (in metric coordinate) at pixel coordinates.

WorldToImage	Transforms a 3D world position to a floating point (sub)pixel image position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The Z value is given in grey scale value. Returns true if the pixel position is inside the image limits and the Z value is positive. The parameter pixelPt is filled even if the point is outside the image.
WorldToZMap	Transforms a 3D world position to a 3D EZMap32f position. The ZMap space origin is at the lower left corner of the image. The scales of the world space and ZMap space are the same.
ZMapToImage	Converts a 2D coordinate in the EZMap32f space to image (sub)pixel space. (x,y) is the ZMap position (which has the same scale as the world space). (u,v) is the corresponding pixel position (with its origin in the upper left corner of the image). All values are expressed in floating point numbers. Returns true if the pixel position is inside the image limits.
ZMapToWorld	Transforms a 3D EZMap32f position to a 3D world space position. The ZMap space origin is at the lower left corner of the image.

EZMap32f::AddMetadata

Adds a metadata key (name) and value.
If the metadata key already exists, its value will be overwritten.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EZMap32f::AsEImage

Returns the EZMap32f as an EImageBW32 (32 bits gray scale) to use with existing eVision 2D tools.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EImageBW32f& AsEImage(
 )

const EImageBW32f& AsEImage(
 )
```

EZMap32f::Clear

Clears the ZMap: replaces all pixels with the undefined value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Clear(
 )
```

EZMap32f::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ClearMetadata(
 )
```

EZMap32f::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Buffer coordinates

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D
The Map X coordinate.

y3D
The Map Y coordinate.

xBuffer
The returned Pixel X coordinate.

yBuffer
The returned Pixel Y coordinate.

EZMap32f::ConvertCoordinatesPixelToMap

Converts Buffer coordinates to 3D Map coordinates

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EZMap32f::CopyMetadataTo

Copies all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void CopyMetadataTo(
 EZMap32f& other
 )
```

Parameters

other

An other EZMap32f.

EZMap32f::DeleteMetadata

Deletes value of this existing metadata key . Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key

-

EZMap32f::Draw

Draws an EZMap32f in a device context.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the ZMap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A ZMap can be drawn (its pixels rendered) using a device context. EZMap32f::Draw and EZMap32f::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap32f::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context. EZMap32f::Draw and EZMap32f::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap32f::EZMap32f

Creates a 32 bits EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EZMap32f(
 )

void EZMap32f(
 int width,
 int height
 )

void EZMap32f(
 const EZMap32f& other
 )
```

Parameters

width

The width of the new ZMap.

height

The height of the new ZMap.

other

Another ZMap.

EZMap32f::FillUndefinedPixels

Fills undefined pixels, used to fill the "holes" in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillUndefinedPixels(
 EZMap32f& outMap,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

outMap

The destination ZMap.

direction

Direction in which the undefined pixels are filled in a ZMap from EFillUndefinedPixelsDirection.

method

Which values used to fill the undefined pixels in a ZMap from EFillUndefinedPixelsMethod.

EZMap32f::FillUndefinedPixelsWithMedian

Fills undefined pixels, used to fill the "holes" in the ZMap using a median rectangular kernel of odd size.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void FillUndefinedPixelsWithMedian(
 EZMap32f& outMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

outMap

The destination ZMap.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 2).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth).

EZMap32f::GetBufferPtr

Retrieves the pointer to the internal pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )
void* GetBufferPtr(
 int x,
 int y
 )
const void* GetBufferPtr(
 )
const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel which we want the address.

y

Row of the pixel which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EZMap32f::GetCheckedBufferPtr

Retrieves the pointer to the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )
const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

EZMap32f::GetMetadata

Returns the string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key
The name of an existing metadata.

EZMap32f::GetPixel

Gets the value of a pixel .

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepth32f GetPixel(
 int x,
 int y
 )
```

Parameters

x
Column of the pixel.

y
Row of the pixel.

EZMap32f::GetPixelPositionFromWorldPosition

Returns in the u, v and value parameters the EZMap32f values corresponding to a 3D world position.
The world position is projected on the ZMap reference plane to get a position in the ZMap.
If the projected position is outside the ZMap, the method returns false.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool GetPixelPositionFromWorldPosition(
 const E3DPoint& world_position,
 int& u,
 int& v,
 EDepth32f& value
 )
```

Parameters

world_position
The 3D coordinates of a world position.

u
Column of the ZMap pixel in [0,width[.

v
Row of the ZMap pixel in [0,height[.

value
Value of the pixel.

EZMap32f::GetResolution

Gets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetResolution(
 float& sx,
 float& sy,
 float& sz
 )
E3DPoint GetResolution(
 )
```

Parameters

sx
Resolution along the X axis.

sy
Resolution along the Y axis.

sz
Resolution along the Z axis.

EZMap32f::GetSizeInWorld

Returns the dimensions of the EZMap32f in real world space (e.g metric unit).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void GetSizeInWorld(
 float& worldWidth,
 float& worldHeight
 )
```

Parameters

worldWidth

Contains the size of the ZMap along the X axis (column).

worldHeight

Contains the size of the ZMap along the Y axis (row).

EZMap32f::GetWorldPositionFromMapPosition

Returns the 3D world position corresponding to a EZMap32f 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap32f, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
E3DPoint GetWorldPositionFromMapPosition(
 float x,
 float y
 )
```

Parameters

x

The X coordinate.

y

The Y coordinate.

EZMap32f::GetWorldPositionFromPixelPosition

Returns the 3D world position corresponding to a EZMap32f pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap32f::GetZMapPositionFromPixelPosition

Returns the corresponding EZMap32f 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetZMapPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap32f::GetZRange

Compute the minimum and maximum pixel values, excluding the undefined pixels.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetZRange(
 EBW32f& min,
 EBW32f& max
 )
```

Parameters

min
The lowest pixel value.

max
The highest pixel value.

EZMap32f::GetZValue

Gets Z value (in metric coordinate) at pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x
X Coordinate.

y
Y Coordinate.

EZMap32f::GetHeight

EZMap32f::SetHeight

Access ZMap Height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetHeight() const

void SetHeight(int height)
```

EZMap32f::ImageToWorld

Transforms a floating point (sub)pixel image position to a 3D world position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The pixel Z value is in grey scale values (its range depends on the ZMap type).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToWorld(
 const E3DPoint& pixelPt,
 E3DPoint& worldPt
 )
```

Parameters

pixelPt

Position in the image space.

worldPt

Position in the 3D world space.

EZMap32f::ImageToZMap

Converts a 2D image (sub)pixel coordinate to the EZMap32f space.
(u,v) is the pixel position (with its origin in the upper left corner of the image).
(x,y) is the corresponding ZMap position (which has the same scale as the world space).
All values are expressed in floating point numbers.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToZMap(
 float u,
 float v,
 float& x,
 float& y
 )
```

Parameters

u

X Coordinate of the pixel as a floating point value.

v

Y Coordinate of the pixel as a floating point value.

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

EZMap32f::IsVoid

Tests if the EZMap32f object size is zero.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool IsVoid(
 )
```

Remarks

Returns true if the ZMap size is zero.

EZMap32f::Load

Restores the EZMap32f stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path to the file.

serializer

-

Remarks

When loading, the ZMap is resized if needed. This function restores all the ZMap attributes.

EZMap32f::LoadImage

Restores the EZMap32f image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the ZMap is resized if needed. This function does not restore the ZMap attributes, only the image associated with the EZMap32f is updated.

EZMap32f::LoadImageAndMetadata

Loads image format and Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImageAndMetadata(
 const std::string& path,
 const std::string& pathMetadata
 )
```

Parameters

path

-

pathMetadata

Full path to the file.

EZMap32f::LoadMetadata

Loads Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EZMap32f::GetMapToWorldMatrix

EZMap32f::SetMapToWorldMatrix

E3DTransformMatrix that transforms positions from the EZMap32f space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetMapToWorldMatrix() const

void SetMapToWorldMatrix(const E3DTransformMatrix& matrix)
```

EZMap32f::ModifyMetadata

Changes an existing metadata key and value. Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

-

value

-

EZMap32f::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EZMap32f& operator=(
 const EZMap32f& other
 )
```

Parameters

other

The source EZMap32f.

EZMap32f::ResetWorldTransformation

Reset the world transformation, Map to World and World to Map matrices become identity matrix.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ResetWorldTransformation(
 )
```

EZMap32f::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetRowPitch() const
```

EZMap32f::Save

Saves the EZMap32f object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This format save the EZMap32f in a Open eVision file. This function stores all the ZMap attributes.

EZMap32f::SaveImage

Saves the EZMap32f image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This format save the image associated to EZMap32f in a standard image file and thus does not store ZMap attributes.

EZMap32f::SaveImageAndMetadata

Saves image format and Metadata JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& path,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

path

-

pathMetadata

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

EZMap32f::SaveMetadata

Saves Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EZMap32f::SetBufferPtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the data.

bitsPerRow

The total number of bits contained in a row, padding included.
Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EZMap32f::SetBufferPtr.

EZMap32f::SetPixel

Sets the value of a pixel .

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPixel(
 EDepth32f value,
 int x,
 int y
 )
```

Parameters

value

Value of the pixel.

x

Column of the pixel.

y

Row of the pixel.

EZMap32f::SetResolution

Sets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetResolution(
 float rx,
 float ry,
 float rz
 )

void SetResolution(
 E3DPoint resolution
 )
```

Parameters

rx
Resolution along the X axis.

ry
Resolution along the Y axis.

rz
Resolution along the Z axis.

resolution
Resolution for X,Y and Z axis.

EZMap32f::SetSize

Sets the width and height of the EZMap32f.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetSize(
 int width,
 int height
 )

void SetSize(
 const EZMap& other
 )
```

Parameters

width

The new requested width.

height

The new requested height.

other

The other ZMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an ZMap* is specified as a number of columns (width) and rows (height).
The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries.

EZMap32f::SetZValue

Sets Z value (in metric coordinate) at pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetZValue(
 const float value,
 const int x,
 const int y
 )
```

Parameters

value

Value of the pixel in metric space.

x

X Coordinate.

y

Y Coordinate.

EZMap32f::GetType

Pixel accessor type.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::EImageType GetType() const
```

EZMap32f::GetUndefinedValue

Returns the Undefined value. That value is used to set to mark pixels with no valid depth value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth32f GetUndefinedValue() const
```

EZMap32f::GetWidth

EZMap32f::SetWidth

Access ZMap Width.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetWidth() const
void SetWidth(int width)
```

EZMap32f::GetWorldShape

Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap32f in real space coordinates (e.g mm).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const EWorldShape& GetWorldShape() const
```

EZMap32f::WorldToImage

Transforms a 3D world position to a floating point (sub)pixel image position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The Z value is given in grey scale value.
Returns true if the pixel position is inside the image limits and the Z value is positive.
The parameter pixelPt is filled even if the point is outside the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool WorldToImage(
 const E3DPoint& worldPt,
 E3DPoint& pixelPt
 )
```

Parameters

worldPt

Position in the 3D world space.

pixelPt

Position in the image space.

EZMap32f::GetWorldToMapMatrix

EZMap32f::SetWorldToMapMatrix

Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap32f space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetWorldToMapMatrix() const

void SetWorldToMapMatrix(const E3DTransformMatrix& matrix)
```

EZMap32f::WorldToZMap

Transforms a 3D world position to a 3D EZMap32f position.
The ZMap space origin is at the lower left corner of the image.
The scales of the world space and ZMap space are the same.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void WorldToZMap(
 const E3DPoint& worldPt,
 E3DPoint& zmapPt
 )
```

Parameters

worldPt

Position in the 3D world space.

zmapPt

Position in the ZMap space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap32f::GetXResolution

EZMap32f::SetXResolution

Resolution of the EZMap32f along the X axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetXResolution() const

void SetXResolution(float resolution)
```

EZMap32f::GetYResolution

EZMap32f::SetYResolution

Resolution of the EZMap32f along the Y axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetYResolution() const

void SetYResolution(float resolution)
```

EZMap32f::ZMapToImage

Converts a 2D coordinate in the EZMap32f space to image (sub)pixel space.
(x,y) is the ZMap position (which has the same scale as the world space).
(u,v) is the corresponding pixel position (with its origin in the upper left corner of the image).
All values are expressed in floating point numbers.
Returns true if the pixel position is inside the image limits.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ZMapToImage(
 float x,
 float y,
 float& u,
 float& v
 )
```

Parameters

x
Position along horizontal axis in the ZMap space.

y
Position along vertical axis in the ZMap space.

u
Column of the pixel as a floating point value.

v
Row of the pixel as a floating point value.

EZMap32f::ZMapToWorld

Transforms a 3D EZMap32f position to a 3D world space position.
The ZMap space origin is at the lower left corner of the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ZMapToWorld(
 const E3DPoint& zmapPt,
 E3DPoint& worldPt
 )
```

Parameters

zmapPt

Position in the ZMap space.

worldPt

Position in the 3D world space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap32f::GetZResolution

EZMap32f::SetZResolution

Resolution of the EZMap32f along the Z axis
The resolution is the number of metric units per grey value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.261. EZMap8 Class

A ZMap8 is a 8bits corrected 2.5D image.
ZMap Pixel values (8 bits integers) represent distances from a 3D reference plane.
Distances are positive, during the ZMap generation all points below the reference plane are discarded.
The EZMap class also stores the transformation from the pixel coordinates to the real world coordinate system.
There could be undefined pixels in the ZMap.

Base Class:EZMap

Namespace: Euresys::Open_eVision::Easy3D

Methods

AddMetadata	Adds a metadata key (name) and value. If the metadata key already exists, its value will be overwritten.
AsEImage	Returns the EZMap8 as an EImageBW8 (8 bits gray scale) to use with existing eVision 2D tools.
Clear	Clears the ZMap: replaces all pixels with the undefined value.
ClearMetadata	Deletes all metadata.

ConvertCoordinatesMapToPixel	Converts 3D Map coordinates to Buffer coordinates
ConvertCoordinatesPixelToMap	Converts Buffer coordinates to 3D Map coordinates
CopyMetadataTo	Copies all metadata.
DeleteMetadata	Deletes value of this existing metadata key . Throws an exception if it does not exist.
Draw	Draws an EZMap8 in a device context.
DrawImage	Displays the internal image buffer
EZMap8	Creates a 8 bits EZMap.
FillUndefinedPixels	Fills undefined pixels, used to fill the "holes" in the ZMap.
FillUndefinedPixelsWithMedian	Fills undefined pixels, used to fill the "holes" in the ZMap using a median rectangular kernel of odd size.
GetBufferPtr	Retrieves the pointer to the internal pixel buffer.
GetCheckedBufferPtr	Retrieves the pointer to the pixel buffer.
GetHeight	Access ZMap Height.
GetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap8 space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
GetMetadata	Returns the string value of the given metadata. Throws an exception if it does not exist.
GetPixel	Gets the value of a pixel .
GetPixelPositionFromWorldPosition	Returns in the u, v and value parameters the EZMap8 values corresponding to a 3D world position. The world position is projected on the ZMap reference plane to get a position in the ZMap. If the projected position is outside the ZMap, the method returns false.
GetResolution	Gets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.
GetRowPitch	Returns the buffer row pitch.
GetSizeInWorld	Returns the dimensions of the EZMap8 in real world space (e.g metric unit).
GetType	Pixel accessor type.
GetUndefinedValue	Returns the Undefined value. That value is used to set to mark pixels with no valid depth value.
GetWidth	Access ZMap Width.
GetWorldPositionFromMapPosition	Returns the 3D world position corresponding to a EZMap8 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap8, otherwise an exception will be thrown.

GetWorldPositionFromPixelPosition	Returns the 3D world position corresponding to a EZMap8 pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.
GetWorldShape	Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap8 in real space coordinates (e.g mm).
GetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap8 space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
GetXResolution	Resolution of the EZMap8 along the X axis The resolution is the number of metric units per pixel.
GetYResolution	Resolution of the EZMap8 along the Y axis The resolution is the number of metric units per pixel.
GetZMapPositionFromPixelPosition	Returns the corresponding EZMap8 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.
GetZRange	Compute the minimum and maximum pixel values, excluding the undefined pixels.
GetZResolution	Resolution of the EZMap8 along the Z axis The resolution is the number of metric units per grey value.
GetZValue	Gets Z value (in metric coordinate) at pixel coordinates.
ImageToWorld	Transforms a floating point (sub)pixel image position to a 3D world position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The pixel Z value is in grey scale values (its range depends on the ZMap type).
ImageToZMap	Converts a 2D image (sub)pixel coordinate to the EZMap8 space. (u,v) is the pixel position (with its origin in the upper left corner of the image). (x,y) is the corresponding ZMap position (which has the same scale as the world space). All values are expressed in floating point numbers.
IsVoid	Tests if the EZMap8 object size is zero.
Load	Restores the EZMap8 stored in the given Open eVision file.
LoadImage	Restores the EZMap8 image stored in the given image file.
LoadImageAndMetadata	Loads image format and Metadata in JSON format.
LoadMetadata	Loads Metadata in JSON format.

ModifyMetadata	Changes an existing metadata key and value. Throws an exception if it does not exist.
operator=	Assignment operator.
ResetWorldTransformation	Reset the world transformation, Map to World and World to Map matrices become identity matrix.
Save	Saves the EZMap8 object to the given Open eVision file.
SaveImage	Saves the EZMap8 image to the given image file.
SaveImageAndMetadata	Saves image format and Metadata JSON format.
SaveMetadata	Saves Metadata in JSON format.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.
SetHeight	Access ZMap Height.
SetMapToWorldMatrix	E3DTransformMatrix that transforms positions from the EZMap8 space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.
SetPixel	Sets the value of a pixel .
SetResolution	Sets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.
SetSize	Sets the width and height of the EZMap8.
SetWidth	Access ZMap Width.
SetWorldToMapMatrix	Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap8 space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.
SetXResolution	Resolution of the EZMap8 along the X axis The resolution is the number of metric units per pixel.
SetYResolution	Resolution of the EZMap8 along the Y axis The resolution is the number of metric units per pixel.
SetZResolution	Resolution of the EZMap8 along the Z axis The resolution is the number of metric units per grey value.
SetZValue	Sets Z value (in metric coordinate) at pixel coordinates.
WorldToImage	Transforms a 3D world position to a floating point (sub)pixel image position. The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel. The Z value is given in grey scale value. Returns true if the pixel position is inside the image limits and the Z value is positive. The parameter pixelPt is filled even if the point is outside the image.
WorldToZMap	Transforms a 3D world position to a 3D EZMap8 position. The ZMap space origin is at the lower left corner of the image. The scales of the world space and ZMap space are the same.

ZMapToImage	Converts a 2D coordinate in the EZMap8 space to image (sub)pixel space. (x,y) is the ZMap position (which has the same scale as the world space). (u,v) is the corresponding pixel position (with its origin in the upper left corner of the image). All values are expressed in floating point numbers. Returns true if the pixel position is inside the image limits.
ZMapToWorld	Transforms a 3D EZMap8 position to a 3D world space position. The ZMap space origin is at the lower left corner of the image.

EZMap8::AddMetadata

Adds a metadata key (name) and value.
If the metadata key already exists, its value will be overwritten.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void AddMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

The name of the metadata. Names are unique.

value

The value for the given metadata.

EZMap8::AsEImage

Returns the EZMap8 as an EImageBW8 (8 bits gray scale) to use with existing eVision 2D tools.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EImageBW8& AsEImage(
 )

const EImageBW8& AsEImage(
 )
```

EZMap8::Clear

Clears the ZMap: replaces all pixels with the undefined value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Clear(
 )
```

EZMap8::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void ClearMetadata(
 )
```

EZMap8::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Buffer coordinates

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool ConvertCoordinatesMapToPixel(
 float x3D,
 float y3D,
 int& xBuffer,
 int& yBuffer
 )
```

Parameters

x3D
The Map X coordinate.

y3D
The Map Y coordinate.

xBuffer
The returned Pixel X coordinate.

yBuffer
The returned Pixel Y coordinate.

EZMap8::ConvertCoordinatesPixelToMap

Converts Buffer coordinates to 3D Map coordinates

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ConvertCoordinatesPixelToMap(
 int xBuffer,
 int yBuffer,
 float& x3D,
 float& y3D
 )
```

Parameters

xBuffer
The pixel X coordinate.

yBuffer
The pixel Y coordinate.

x3D
The returned Map X coordinate.

y3D
The returned Map Y coordinate.

EZMap8::CopyMetadataTo

Copies all metadata.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void CopyMetadataTo(
 EZMap8& other
 )
```

Parameters

other
An other EZMap8.

EZMap8::DeleteMetadata

Deletes value of this existing metadata key . Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key

-

EZMap8::Draw

Draws an EZMap8 in a device context.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )

void Draw(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY
 )
void Draw(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the ZMap is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

A ZMap can be drawn (its pixels rendered) using a device context. EZMap8::Draw and EZMap8::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap8::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void DrawImage(
 EDrawAdapter* graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 EDrawAdapter* graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )

void DrawImage(
 HDC graphicContext,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

```
void DrawImage(
 HDC graphicContext,
 EC24Vector* c24Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
void DrawImage(
 HDC graphicContext,
 EBW8Vector* bw8Vector,
 float zoomX,
 float zoomY,
 float panX,
 float panY,
 EC24 colorUndefinedPixel
 )
```

Parameters

graphicContext

Handle to the device context of the destination window.

zoomX

Magnification factor for zooming in or out in the horizontal direction. By default, the image is displayed in 1:1 scale.

zoomY

Magnification factor for zooming in or out in the vertical direction. Setting a 0 value (which is the default) will result in isotropic scaling (i.e. equal horizontal and vertical factors).

panX

Horizontal panning value expressed in pixels. By default, no panning occurs.

panY

Vertical panning value expressed in pixels. By default, no panning occurs.

colorUndefinedPixel

An optional parameter to choose the drawing color of undefined pixels.

c24Vector

When supplied, this parameter allows using a LUT that maps from Depth to C24 when drawing (false colors).

bw8Vector

When supplied, this parameter allows using a LUT that maps from Depth to BW8 when drawing.

Remarks

An image can be drawn (its pixels rendered) using a device context. EZMap8::Draw and EZMap8::DrawImage produce the same output.
The horizontal and vertical zooming factors can be different but must be in the 1/16..16 range.
(MFC users can use the CDC::GetSafeHdc() method to obtain a suitable device context handle from a CDC instance.) Deprecation notice: All methods taking HDC as parameter are deprecated. It is recommended to use their alternative taking a EDrawAdapter by using a instance of EWindowsDrawAdapter.

EZMap8::EZMap8

Creates a 8 bits EZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EZMap8(
 )

void EZMap8(
 int width,
 int height
 )

void EZMap8(
 const EZMap8& other
 )
```

Parameters

width
The width of the new ZMap.

height
The height of the new ZMap.

other
Another ZMap.

EZMap8::FillUndefinedPixels

Fills undefined pixels, used to fill the "holes" in the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillUndefinedPixels(
 EZMap8& outMap,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision::Easy3D::EFillUndefinedPixelsMethod method
 )
```

Parameters

outMap

The destination ZMap.

direction

Direction in which the undefined pixels are filled in a ZMap from EFillUndefinedPixelsDirection.

method

Which values used to fill the undefined pixels in a ZMap from EFillUndefinedPixelsMethod.

EZMap8::FillUndefinedPixelsWithMedian

Fills undefined pixels, used to fill the "holes" in the ZMap using a median rectangular kernel of odd size.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void FillUndefinedPixelsWithMedian(
 EZMap8& outMap,
 OEV_UINT32 halfOfKernelWidth,
 OEV_UINT32 halfOfKernelHeight
 )
```

Parameters

outMap

The destination ZMap.

halfOfKernelWidth

Half of the box width minus one (by default, halfOfKernelWidth = 2).

halfOfKernelHeight

Half of the box height minus one (by default, same as halfOfKernelWidth).

EZMap8::GetBufferPtr

Retrieves the pointer to the internal pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetBufferPtr(
 )

void* GetBufferPtr(
 int x,
 int y
 )

const void* GetBufferPtr(
 )
```

```
const void* GetBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel which we want the address.

y

Row of the pixel which we want the address.

Remarks

This function does not check the value of the parameters.
Use carefully.

EZMap8::GetCheckedBufferPtr

Retrieves the pointer to the pixel buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void* GetCheckedBufferPtr(
 int x,
 int y
 )
const void* GetCheckedBufferPtr(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel of which we want the address.

y

Row of the pixel of which we want the address.

EZMap8::GetMetadata

Returns the string value of the given metadata.
Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an existing metadata.

EZMap8::GetPixel

Gets the value of a pixel .

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EDepth8 GetPixel(
 int x,
 int y
 )
```

Parameters

x

Column of the pixel.

y

Row of the pixel.

EZMap8::GetPixelPositionFromWorldPosition

Returns in the u, v and value parameters the EZMap8 values corresponding to a 3D world position.
The world position is projected on the ZMap reference plane to get a position in the ZMap.
If the projected position is outside the ZMap, the method returns false.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetPixelPositionFromWorldPosition(
 const E3DPoint& world_position,
 int& u,
 int& v,
 EDepth8& value
 )
```

Parameters

world_position
The 3D coordinates of a world position.

u
Column of the ZMap pixel in [0,width[.

v
Row of the ZMap pixel in [0,height[.

value
Value of the pixel.

EZMap8::GetResolution

Gets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetResolution(
 float& sx,
 float& sy,
 float& sz
 )

E3DPoint GetResolution(
 )
```

Parameters

sx
Resolution along the X axis.

sy
Resolution along the Y axis.

sz
Resolution along the Z axis.

EZMap8::GetSizeInWorld

Returns the dimensions of the EZMap8 in real world space (e.g metric unit).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetSizeInWorld(
 float& worldWidth,
 float& worldHeight
 )
```

Parameters

worldWidth

Contains the size of the ZMap along the X axis (column).

worldHeight

Contains the size of the ZMap along the Y axis (row).

EZMap8::GetWorldPositionFromMapPosition

Returns the 3D world position corresponding to a EZMap8 2D coordinate. The world position is in the original point cloud space. (x,y) is the ZMap position (which has the same scale as the world space). The value at position (x, y) must not be undefined in the EZMap8, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromMapPosition(
 float x,
 float y
 )
```

Parameters

x

The X coordinate.

y

The Y coordinate.

EZMap8::GetWorldPositionFromPixelPosition

Returns the 3D world position corresponding to a EZMap8 pixel position. The world position is in the original point cloud space. The pixel space origin is at the upper left corner of the image and ranges to (width-1, height-1). The transformation to the world position is calculated using the center of the pixel. The pixel value at position (u,v) must not be undefined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap8::GetZMapPositionFromPixelPosition

Returns the corresponding EZMap8 3D position of a ZMap pixel. The ZMap position is in the original point ZMap space. The pixel position origin is the upper left corner of the image and ranges to (width-1, height-1). The center of the pixel is used for the transformation to the ZMap position. Pixel at position (u,v) must be defined, otherwise an exception will be thrown.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetZMapPositionFromPixelPosition(
 int u,
 int v
 )
```

Parameters

u

Column of the pixel (bounds: [0,width[).

v

Row of the pixel (bounds: [0,height[).

EZMap8::GetZRange

Compute the minimum and maximum pixel values, excluding the undefined pixels.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void GetZRange(
 EBW8& min,
 EBW8& max
 )
```

Parameters

min

The lowest pixel value.

max

The highest pixel value.

EZMap8::GetZValue

Gets Z value (in metric coordinate) at pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZValue(
 const int x,
 const int y
 )
```

Parameters

x

X Coordinate.

y

Y Coordinate.

EZMap8::GetHeight

EZMap8::SetHeight

Access ZMap Height.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetHeight() const

void SetHeight(int height)
```

EZMap8::ImageToWorld

Transforms a floating point (sub)pixel image position to a 3D world position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The pixel Z value is in grey scale values (its range depends on the ZMap type).

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToWorld(
 const E3DPoint& pixelPt,
 E3DPoint& worldPt
 )
```

Parameters

pixelPt

Position in the image space.

worldPt

Position in the 3D world space.

EZMap8::ImageToZMap

Converts a 2D image (sub)pixel coordinate to the EZMap8 space.
(u,v) is the pixel position (with its origin in the upper left corner of the image).
(x,y) is the corresponding ZMap position (which has the same scale as the world space).
All values are expressed in floating point numbers.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ImageToZMap(
 float u,
 float v,
 float& x,
 float& y
 )
```

Parameters

u

X Coordinate of the pixel as a floating point value.

v

Y Coordinate of the pixel as a floating point value.

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

EZMap8::IsVoid

Tests if the EZMap8 object size is zero.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool IsVoid(
 )
```

Remarks

Returns true if the ZMap size is zero.

EZMap8::Load

Restores the EZMap8 stored in the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

Full path to the file.

serializer

-

Remarks

When loading, the ZMap is resized if needed. This function restores all the ZMap attributes.

EZMap8::LoadImage

Restores the EZMap8 image stored in the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void LoadImage(
 const std::string& path,
 bool withMetadata
 )
```

Parameters

path

Full path to the file.

withMetadata

Parameter to load or not the metadata that has the same filename. False by default.

Remarks

When loading, the ZMap is resized if needed. This function does not restore the ZMap attributes, only the image associated with the EZMap8 is updated.

EZMap8::LoadImageAndMetadata

Loads image format and Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata
 )
```

Parameters

pathImage

Full path to the file.

pathMetadata

Full path to the file.

EZMap8::LoadMetadata

Loads Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void LoadMetadata(
 const std::string& path
 )
```

Parameters

path

Full path to the file.

EZMap8::GetMapToWorldMatrix

EZMap8::SetMapToWorldMatrix

E3DTransformMatrix that transforms positions from the EZMap8 space to the world space. This transformation is composed of rotation and translation only, so it is a rigid transformation and preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const E3DTransformMatrix& GetMapToWorldMatrix() const
```

```
void SetMapToWorldMatrix(const E3DTransformMatrix& matrix)
```

EZMap8::ModifyMetadata

Changes an existing metadata key and value. Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

-

value

-

EZMap8::operator=

Assignment operator.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EZMap8& operator=(
 const EZMap8& other
 )
```

Parameters

other

The source EZMap8.

EZMap8::ResetWorldTransformation

Reset the world transformation, Map to World and World to Map matrices become identity matrix.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ResetWorldTransformation(
 )
```

EZMap8::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
int GetRowPitch() const
```

EZMap8::Save

Saves the EZMap8 object to the given Open eVision file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The full path to the destination file.

serializer

-

Remarks

This format save the EZMap8 in a Open eVision file. This function stores all the ZMap attributes.

EZMap8::SaveImage

Saves the EZMap8 image to the given image file.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision::EImageFileType type,
 bool withMetadata
 )
```

Parameters

path

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

withMetadata

Parameter to save or not the metadata that with the same filename next. False by default.

Remarks

This format save the image associated to EZMap8 in a standard image file and thus does not store ZMap attributes.

EZMap8::SaveImageAndMetadata

Saves image format and Metadata JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision::EImageFileType type
 )
```

Parameters

pathImage

The full path to the destination file.

pathMetadata

The full path to the destination file.

type

File format, as defined by EImageFileType. If not specified, the file format is determined from the file extension.

EZMap8::SaveMetadata

Saves Metadata in JSON format.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SaveMetadata(
 const std::string& path
 )
```

Parameters

path

The full path to the destination file.

EZMap8::SetBufferPtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetBufferPtr(
 int width,
 int height,
 void* imagePointer,
 int bitsPerRow
 )
```

Parameters

width

The width of the supplied buffer, in pixels.

height

The height of the supplied buffer, in pixels.

imagePointer

The pointer (aligned on 4 bytes) to the buffer, which must be large enough to hold the data.

bitsPerRow

The total number of bits contained in a row, padding included.
Using the value 0 (default) means that this size is computed from the buffer width and the pixel size plus a padding with the smallest possible value that leads to a multiple of 4 bytes (32 bits), which is the minimum padding accepted by EZMap8::SetBufferPtr.

EZMap8::SetPixel

Sets the value of a pixel .

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetPixel(
 EDepth8 value,
 int x,
 int y
 )
```

Parameters

value
Value of the pixel.

x
Column of the pixel.

y
Row of the pixel.

EZMap8::SetResolution

Sets the resolution. For the X and Y axes, the resolution is expressed in metric units per pixel. For the Z axis it is expressed in metric units per grey scale value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetResolution(
 float rx,
 float ry,
 float rz
 )

void SetResolution(
 E3DPoint resolution
 )
```

Parameters

rx
Resolution along the X axis.

ry
Resolution along the Y axis.

rz
Resolution along the Z axis.

resolution
Resolution for X,Y and Z axis.

EZMap8::SetSize

Sets the width and height of the EZMap8.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetSize(
 int width,
 int height
 )
```

```
void SetSize(
 const EZMap& other
 )
```

Parameters

width

The new requested width.

height

The new requested height.

other

The other ZMap whose dimensions have to be used for the current object.

Remarks

Open eVision will allocate a new image buffer (deallocate the old image buffer) if the supplied width and height are different from the existing ones.
If an external buffer has been specified by means of SetImagePtr, it will be kept only if the size does not change.
Creating a new Open eVision image buffer and setting its size creates a 4-byte aligned buffer, by default. The *size of an ZMap* is specified as a number of columns (width) and rows (height).
The maximum image dimensions are 32767 by 32767. Furthermore, it must fit into the available memory, that depends upon the physical memory, the operating system and the memory already allocated by the process in other modules or libraries.

EZMap8::SetZValue

Sets Z value (in metric coordinate) at pixel coordinates.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void SetZValue(
 const float value,
 const int x,
 const int y
 )
```

Parameters

value

Value of the pixel in metric space.

x

X Coordinate.

y

Y Coordinate.

EZMap8::GetType

Pixel accessor type.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
Euresys::Open_eVision::EImageType GetType() const
```

EZMap8::GetUndefinedValue

Returns the Undefined value. That value is used to set to mark pixels with no valid depth value.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EDepth8 GetUndefinedValue() const
```

EZMap8::GetWidth

EZMap8::SetWidth

Access ZMap Width.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
int GetWidth() const
void SetWidth(int width)
```

EZMap8::GetWorldShape

Returns the EWorldShape for conversion between 2D image and 2D world space coordinates. This EWorldShape can be used by EasyGauge to do measurements on a EZMap8 in real space coordinates (e.g mm).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
const EWorldShape& GetWorldShape() const
```

EZMap8::WorldToImage

Transforms a 3D world position to a floating point (sub)pixel image position.
The image space origin is at the upper left corner of the image and the X/Y unit size is one pixel.
The Z value is given in grey scale value.
Returns true if the pixel position is inside the image limits and the Z value is positive.
The parameter pixelPt is filled even if the point is outside the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool WorldToImage(
 const E3DPoint& worldPt,
 E3DPoint& pixelPt
 )
```

Parameters

worldPt

Position in the 3D world space.

pixelPt

Position in the image space.

EZMap8::GetWorldToMapMatrix

EZMap8::SetWorldToMapMatrix

Sets/Gets the E3DTransformMatrix that transforms positions from the world space to the EZMap8 space. This transformation is composed of rotation and translation only, so it is a rigid transformation and it preserves distances.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
const E3DTransformMatrix& GetWorldToMapMatrix() const

void SetWorldToMapMatrix(const E3DTransformMatrix& matrix)
```

EZMap8::WorldToZMap

Transforms a 3D world position to a 3D EZMap8 position.
The ZMap space origin is at the lower left corner of the image.
The scales of the world space and ZMap space are the same.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void WorldToZMap(
 const E3DPoint& worldPt,
 E3DPoint& zmapPt
 )
```

Parameters

worldPt

Position in the 3D world space.

zmapPt

Position in the ZMap space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap8::GetXResolution

EZMap8::SetXResolution

Resolution of the EZMap8 along the X axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetXResolution() const

void SetXResolution(float resolution)
```

EZMap8::GetYResolution

EZMap8::SetYResolution

Resolution of the EZMap8 along the Y axis
The resolution is the number of metric units per pixel.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetYResolution() const

void SetYResolution(float resolution)
```

EZMap8::ZMapToImage

Converts a 2D coordinate in the EZMap8 space to image (sub)pixel space.
(x,y) is the ZMap position (which has the same scale as the world space).
(u,v) is the corresponding pixel position (with its origin in the upper left corner of the image).
All values are expressed in floating point numbers.
Returns true if the pixel position is inside the image limits.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool ZMapToImage(
 float x,
 float y,
 float& u,
 float& v
 )
```

Parameters

x

Position along horizontal axis in the ZMap space.

y

Position along vertical axis in the ZMap space.

u

Column of the pixel as a floating point value.

v

Row of the pixel as a floating point value.

EZMap8::ZMapToWorld

Transforms a 3D EZMap8 position to a 3D world space position.
The ZMap space origin is at the lower left corner of the image.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ZMapToWorld(
 const E3DPoint& zmapPt,
 E3DPoint& worldPt
 )
```

Parameters

zmapPt

Position in the ZMap space.

worldPt

Position in the 3D world space.

Remarks

Do not use this method with the same variable as input and output. It might lead to incorrect results.

EZMap8::GetZResolution

EZMap8::SetZResolution

Resolution of the EZMap8 along the Z axis
The resolution is the number of metric units per grey value.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.262. EZMapToMeshConverter Class

Performs the conversion from an EZMap to an EMesh.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	The method 'Convert' performs the conversion from an EZMap to an EMesh.
EZMapToMeshConverter	Creates an EZMapToMeshConverter object.
GetMaxEdgeLength	Sets/Gets the maximal length of the longest edge of a triangle of the mesh. Larger triangles will be filtered out. A value of 0 does not perform any filtering. Set to 0 by default.
Load	Loads the converter configuration. The given ESerializer must have been created for reading.
operator=	Assignment operator
Save	Saves the converter configuration. The given ESerializer must have been created for writing.

SetMaxEdgeLength	Sets/Gets the maximal length of the longest edge of a triangle of the mesh. Larger triangles will be filtered out. A value of 0 does not perform any filtering. Set to 0 by default.

EZMapToMeshConverter::Convert

The method 'Convert' performs the conversion from an EZMap to an EMesh.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Convert(
 const EZMap8& srcZMap,
 EMesh& mesh
 )

void Convert(
 const EZMap16& srcZMap,
 EMesh& mesh
 )

void Convert(
 const EZMap32f& srcZMap,
 EMesh& mesh
 )
```

Parameters

srcZMap

The ZMap to convert.

mesh

The destination mesh.

EZMapToMeshConverter::EZMapToMeshConverter

Creates an EZMapToMeshConverter object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EZMapToMeshConverter(
 )

void EZMapToMeshConverter(
 const EZMapToMeshConverter& other
 )
```

Parameters

other

Reference to the EZMapToMeshConverter object used for the initialization.

EZMapToMeshConverter::Load

Loads the converter configuration. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The serializer.

EZMapToMeshConverter::GetMaxEdgeLength

EZMapToMeshConverter::SetMaxEdgeLength

Sets/Gets the maximal length of the longest edge of a triangle of the mesh. Larger triangles will be filtered out. A value of 0 does not perform any filtering. Set to 0 by default.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float GetMaxEdgeLength() const

void SetMaxEdgeLength(float maxEdgeLength)
```

EZMapToMeshConverter::operator=

Assignment operator

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EZMapToMeshConverter& operator=(
 const EZMapToMeshConverter& other
 )
```

Parameters

other

Reference to the EZMapToMeshConverter object used for the assignment.

EZMapToMeshConverter::Save

Saves the converter configuration. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

Pointer to the ESerializer created for writing.

4.263. EZMapToPointCloudConverter Class

Generates an EPointCloud from a ZMap.

Namespace: Euresys::Open_eVision::Easy3D

Methods

Convert	Generates an EPointCloud from a ZMap (EZMap8, EZMap16 or EZMap32f). ZMap pixel positions/values (U,V,W) are converted to 3D positions (X,Y,Z) and written to the given point cloud. By default, the target coordinate system is the original world space. Optional parameter inZMapSpace can switch to the ZMap space as the target coordinate system. Normals may be generated from the ZMap and set as an attribute of the EPointCloud.
EZMapToPointCloudConverter	Creates an EZMapToPointCloudConverter object.

EZMapToPointCloudConverter::Convert

Generates an EPointCloud from a ZMap (EZMap8, EZMap16 or EZMap32f). ZMap pixel positions/values (U,V,W) are converted to 3D positions (X,Y,Z) and written to the given point cloud. By default, the target coordinate system is the original world space. Optional parameter inZMapSpace can switch to the ZMap space as the target coordinate system. Normals may be generated from the ZMap and set as an attribute of the EPointCloud.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Convert(
 const EZMap8& srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )

void Convert(
 const EZMap8& srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )

void Convert(
 const EZMap16& srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )

void Convert(
 const EZMap16& srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )

void Convert(
 const EZMap32f& srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )
```

```
void Convert(
 const EZMap32f& srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )
void Convert(
 const EZMap* srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )
void Convert(
 const EZMap* srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace,
 bool computeNormals
 )
```

Parameters

srcZMap

The ZMap to convert.

pointCloud

The destination point cloud.

inZMapSpace

When true, converts to 3D ZMap space instead of world space (default is false).

computeNormals

When true, computes normals and put them in the EPointCloud (default is false).

region

The region of interest (default is no region).

Remarks

The destination point cloud will be cleared before being (re-)populated.

EZMapToPointCloudConverter::EZMapToPointCloudConverter

Creates an EZMapToPointCloudConverter object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void EZMapToPointCloudConverter(
 )
```

4.264. TextLabel Class

A class representing a text label that can be displayed using E3DViewer.

Namespace: Euresys::Open_eVision::Easy3D

Methods

GetAlignment	The alignment of the label.
GetAnchor	The E3DPoint linked to the text label.
GetBackgroundColor	The background of the label is set to this color.
GetColor	The color of the text label.
GetDisplayAnchor	If set to true, a line is drawn between the anchor and the label.
GetFixed	If set to true, the label stays fixed when the view changes.
GetPosX	The x coordinate of the text box. Value between -1 (left) and 1 (right).
GetPosY	The y coordinate of the text box. Value between -1 (bottom) and 1 (top).
GetSize	The size of the text font. Value between 0 et 1.
GetText	The text of the text label.
operator=	-
SetAlignment	The alignment of the label.
SetAnchor	The E3DPoint linked to the text label.
SetBackgroundColor	The background of the label is set to this color.
SetColor	The color of the text label.
SetDisplayAnchor	If set to true, a line is drawn between the anchor and the label.
SetFixed	If set to true, the label stays fixed when the view changes.
SetPoxX	-
SetPoxY	-
SetSize	The size of the text font. Value between 0 et 1.
SetText	The text of the text label.
TextLabel	Constructs a text label.

TextLabel::GetAlignment

TextLabel::SetAlignment

The alignment of the label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
Euresys::Open_eVision::Easy3D::ETextLabelAlignment GetAlignment() const
void SetAlignment(Euresys::Open_eVision::Easy3D::ETextLabelAlignment value)
```

TextLabel::GetAnchor

TextLabel::SetAnchor

The E3DPoint linked to the text label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
E3DPoint GetAnchor() const
void SetAnchor(E3DPoint value)
```

TextLabel::GetBackgroundColor

TextLabel::SetBackgroundColor

The background of the label is set to this color.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EC24A GetBackgroundColor() const
void SetBackgroundColor(EC24A value)
```

TextLabel::GetColor

TextLabel::SetColor

The color of the text label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
EC24 GetColor() const
void SetColor(EC24 value)
```

TextLabel::GetDisplayAnchor

TextLabel::SetDisplayAnchor

If set to true, a line is drawn between the anchor and the label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetDisplayAnchor() const
void SetDisplayAnchor(bool value)
```

TextLabel::GetFixed

TextLabel::SetFixed

If set to true, the label stays fixed when the view changes.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool GetFixed() const
void SetFixed(bool value)
```

TextLabel::operator=

-

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
TextLabel& operator=(
 const TextLabel& other
 )
```

Parameters

other

-

TextLabel::GetPosX

The x coordinate of the text box. Value between -1 (left) and 1 (right).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetPosX() const
```

TextLabel::GetPosY

The y coordinate of the text box. Value between -1 (bottom) and 1 (top).

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetPosY() const
```

TextLabel::SetPoxX

-

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetPoxX(float value)
```

TextLabel::SetPoxY

-

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
void SetPoxY(float value)
```

TextLabel::GetSize

TextLabel::SetSize

The size of the text font. Value between 0 et 1.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float GetSize() const
```

```
void SetSize(float value)
```

TextLabel::GetText

TextLabel::SetText

The text of the text label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
std::string GetText() const
void SetText(const std::string& value)
```

TextLabel::TextLabel

Constructs a text label.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void TextLabel(
 const E3DPoint& anchor,
 float posX,
 float posY,
 EC24 color,
 float size,
 const std::string& text,
 bool displayAnchor,
 EC24A backgroundColor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
void TextLabel(
 const E3DPoint& anchor,
 EC24 color,
 float size,
 const std::string& text,
 bool fixLabelPosition,
 bool displayAnchor,
 EC24A backgroundColor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
```

```
void TextLabel(
 float posX,
 float posY,
 EC24 color,
 float size,
 const std::string& text,
 EC24A backgroundColor,
 Euresys::Open_eVision::Easy3D::ETextLabelAlignment alignment
 )
void TextLabel(
 const TextLabel& other
 )
```

Parameters

anchor
The E3DPoint linked to the text label.

posX
The x coordinate of the text box. Value between -1 (left) and 1 (right).

posY
The y coordinate of the text box. Value between -1 (bottom) and 1 (top).

color
The color of the text label.

size
The size of the text font. Value between 0 et 1.

text
The text of the text label.

displayAnchor
If set to true, a line is drawn between the anchor and the label (default: true).

backgroundColor
If specified, the background of the label is set to this color (default: black).

alignment
-

fixLabelPosition
If set to true, the label stays fixed when the view changes (default: false).

other
-

5. Structures

5.1. E3DPoint Struct

Represents a 3D point with floating point coordinates.

Namespace: Euresys::Open_eVision::Easy3D

Properties

X	X coordinate of the point.
Y	Y coordinate of the point.
Z	Z coordinate of the point.

Methods

DistanceTo	Returns the euclidean distance to another E3DPoint.
DistanceToSegment	Returns the euclidean distance between the E3DPoint and a segment represented by 2 other E3DPoint.
E3DPoint	Constructs a default E3DPoint object.
Load	Loads the E3DPoint.
operator!=	Checks if two E3DPoint are strictly different.
operator==	Checks if two E3DPoint are strictly equal.
Save	Saves the E3DPoint.
SquareDistanceTo	Returns the euclidean squared distance to another E3DPoint.

E3DPoint::DistanceTo

Returns the euclidean distance to another E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float DistanceTo(
 const E3DPoint& point
 )
```

Parameters

point

The other point.

E3DPoint::DistanceToSegment

Returns the euclidean distance between the E3DPoint and a segment represented by 2 other E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float DistanceToSegment(
 const E3DPoint& from,
 const E3DPoint& to
 )
```

Parameters

from

One end point of the segment

to

The other end point of the segment

E3DPoint::E3DPoint

Constructs a default E3DPoint object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void E3DPoint(
 )

void E3DPoint(
 float x,
 float y,
 float z
 )
```

Parameters

x

X coordinate of the point.

y

Y coordinate of the point.

z

Z coordinate of the point.

Remarks

If the default constructor is used, the point is initialized to (0, 0, 0).

E3DPoint::Load

Loads the E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Load(
 const std::string& fileName
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

fileName

-

serializer

Serializer. Must be in read mode.

E3DPoint::operator!=

Checks if two E3DPoint are strictly different.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator!=(
 const E3DPoint& point
 )
```

Parameters

point

The other point.

E3DPoint::operator==

Checks if two E3DPoint are strictly equal.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
bool operator==(
 const E3DPoint& point
 )
```

Parameters

point

The other point.

E3DPoint::Save

Saves the E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void Save(
 const std::string& fileName
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

fileName

-

serializer

Serializer. Must be in write mode.

E3DPoint::SquareDistanceTo

Returns the euclidean squared distance to another E3DPoint.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float SquareDistanceTo(
 const E3DPoint& point
 )
```

Parameters

point

The other point.

E3DPoint::X

X coordinate of the point.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
float X
```

E3DPoint::Y

Y coordinate of the point.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float Y
```

E3DPoint::Z

Z coordinate of the point.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
float Z
```

5.2. EBarCodeGradingParameters Struct

Represents the grading Parameters associated with an EBarCode as defined by ISO15416.

Remarks

Each grade is expressed as an integer between 0 and 40 instead of a float from 0.0 to 4.0 by steps of 0.1 to avoid rounding errors. Each grade represents the average grade obtained on 10 horizontal lines in the barcode.

Namespace: Euresys::Open_eVision::EasyBarCode2

Properties

DecodeGrade	Indicates whether the barcode was decoded. Grade is 40 if barcode was decoded and 0 if it was not.
SymbolContrastGrade	Indicates the fraction of total image contrast used by the barcode. Grade is between 0 (<= 51 gray values) and 40 (>= 180 gray values).
MinimumReflectanceGrade	Indicates the ratio between the smallest and largest gray value in the barcode. Grade is 40 if the smallest gray value of the barcode is <= half of its highest value and 0 otherwise.
MinimumEdgeContrastGrade	Indicates the smallest contrast between two adjacent bar and space (including quiet zones). Grade is 40 if the minimum edge contrast is >= 15% and 0 otherwise.
ModulationGrade	Indicates the importance of the minimum edge contrast relative to the symbol contrast. Grade is between 40 (>= 70%) and 0 (< 40%).

DefectsGrade	Indicates the importance of irregularities found within elements and quiet zones. Grade is between 40 (small irregularities) and 0 (large irregularities).
DecodabilityGrade	Indicates the importance of differences between the distances measured and expected. Grade is between 40 (distances measured are close to those expected) and 0 (distances measured are far from those expected).
AdditionalRequirementsGrade	Indicates a symbology specific requirement. The name of that specific requirement for the current EBarCodeGradingParameters may be retrieved with EBarCodeGradingParameters::AdditionalRequirementName. EBarCodeSymbologies_Code128, EBarCodeSymbologies_Gs1_128 and EBarCodeSymbologies_Ean13 all grade the size of the quiet zone of the barcode. Grade is 40 (large enough quiet zone) and 0 (quiet zone too small).
GlobalGrade	The global grade is the smallest of all of the other grades. If two scan lines yield different decoded strings, the Global Grade is set to 0. Grade is between 40 (best) and 0 (worst).

Methods

ConvertToAlphabeticGrade	ISO15416 defines grades as either number of letters. We use numbers internally but this function can be used to convert number to letters.
GetAdditionalRequirementName	ISO15416 defines an additional requirement that is symbology specific. This function returns its name for the current EBarCodeGradingParameters. Currently, "Quiet Zone" is always returned.

EBarCodeGradingParameters::GetAdditionalRequirementName

ISO15416 defines an additional requirement that is symbology specific. This function returns its name for the current EBarCodeGradingParameters. Currently, "Quiet Zone" is always returned.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
std::string GetAdditionalRequirementName() const
```

EBarCodeGradingParameters::AdditionalRequirementsGrade

Indicates a symbology specific requirement. The name of that specific requirement for the current EBarCodeGradingParameters may be retrieved with EBarCodeGradingParameters::AdditionalRequirementName. Code128, Gs1_128 and Ean13 all grade the size of the quiet zone of the barcode. Grade is 40 (large enough quiet zone) and 0 (quiet zone too small).

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 AdditionalRequirementsGrade
```

EBarCodeGradingParameters::ConvertToAlphabeticGrade

ISO15416 defines grades as either number of letters. We use numbers internally but this function can be used to convert number to letters.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
char ConvertToAlphabeticGrade(
 OEV_UINT8 grade
 )
```

Parameters

grade

The grade as an integer between 0 and 40.

EBarCodeGradingParameters::DecodabilityGrade

Indicates the importance of differences between the distances measured and expected. Grade is between 40 (distances measured are close to those expected) and 0 (distances measured are far from those expected).

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 DecodabilityGrade
```

EBarCodeGradingParameters::DecodeGrade

Indicates whether the barcode was decoded. Grade is 40 if barcode was decoded and 0 if it was not.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 DecodeGrade
```

EBarCodeGradingParameters::DefectsGrade

Indicates the importance of irregularities found within elements and quiet zones. Grade is between 40 (small irregularities) and 0 (large irregularities).

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 DefectsGrade
```

EBarCodeGradingParameters::GlobalGrade

The global grade is the smallest of all of the other grades. If two scan lines yield different decoded strings, the Global Grade is set to 0. Grade is between 40 (best) and 0 (worst).

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 GlobalGrade
```

EBarCodeGradingParameters::MinimumEdgeContrastGrade

Indicates the smallest contrast between two adjacent bar and space (including quiet zones). Grade is 40 if the minimum edge contrast is >= 15% and 0 otherwise.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 MinimumEdgeContrastGrade
```

EBarCodeGradingParameters::MinimumReflectanceGrade

Indicates the ratio between the smallest and largest gray value in the barcode. Grade is 40 if the smallest gray value of the barcode is <= half of its highest value and 0 otherwise.

Namespace: Euresys::Open_eVision::EasyBarCode2

[C++]

```
OEV_UINT8 MinimumReflectanceGrade
```

EBarCodeGradingParameters::ModulationGrade

Indicates the importance of the minimum edge contrast relative to the symbol contrast. Grade is between 40 (>= 70%) and 0 (< 40%).

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
OEV_UINT8 ModulationGrade
```

EBarCodeGradingParameters::SymbolContrastGrade

Indicates the fraction of total image contrast used by the barcode. Grade is between 0 (<= 51 gray values) and 40 (>= 180 gray values).

Namespace: Euresys::Open_eVision::EasyBarCode2

```
[C++]
OEV_UINT8 SymbolContrastGrade
```

5.3. EBrush Struct

Brush.

Namespace: Euresys::Open_eVision

Properties

Color	Color of the brush.
Opacity	Opacity of the brush.

Methods

EBrush	Constructs an EBrush.
IsValid	Whether the brush is valid, i.e. when its color is defined.
Load	Loads the EBrush. The given ESerializer must have been created for reading.
operator!=	Inequality operator.
operator==	Equality operator.
Save	Saves the EBrush. The given ESerializer must have been created for writing.
Serialize	Serializes the EBrush.

EBrush::Color

Color of the brush.

Namespace: Euresys::Open_eVision

[C++]

```
ERGBColor Color
```

EBrush::EBrush

Constructs an EBrush.

Namespace: Euresys::Open_eVision

[C++]

```
void EBrush(
 )
void EBrush(
 const ERGBColor& color,
 float opacity
 )
```

Parameters

color
Brush color

opacity
Brush opacity

EBrush::IsValid

Whether the brush is valid, i.e. when its color is defined.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsValid(
 )
```

EBrush::Load

Loads the EBrush. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )
```

```
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EBrush::Opacity

Opacity of the brush.

Namespace: Euresys::Open_eVision

[C++]

```
float Opacity
```

EBrush::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EBrush& other
 )
```

Parameters

other
Other brush to compare with.

EBrush::operator==

Equality operator.

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const EBrush& other
 )
```

Parameters

other

Other brush to compare with.

EBrush::Save

Saves the EBrush. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

```
[C++]
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EBrush::Serialize

Serializes the EBrush.

Namespace: Euresys::Open_eVision

```
[C++]
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer

Serializer

5.4. EBW1 Struct

Black and white pixel value, coded as an unsigned 32-bit integer.

Remarks

Every pixel is coded on 1 bit, allowing to represent 2 different values. The value 0 stands for black (background), and the value 1 stands for white (foreground).

Namespace: Euresys::Open_eVision

Properties

Value	The value of the pixel.

Methods

EBW1	Constructs a EBW1 object.
GetSize	The number of bits in a pixel

EBW1::EBW1

Constructs a EBW1 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EBW1(
 )
void EBW1(
 OEV_UINT32 value
 )
```

Parameters

value

The value of the pixel.

EBW1::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

```
[C++]
int GetSize()
```

EBW1::Value

The value of the pixel.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 Value
```

5.5. EBW16 Struct

Gray-level pixel value, coded as an unsigned 16-bit integer.

Remarks

High-quality cameras or scanners are able to digitize on 10 or 12 bits. Sometimes too, to avoid numerical truncation errors, intermediate processing results require more than 8 bits of storage. In such situations, 8 bits gray-level images are no longer sufficient. 16 bits gray-level images are similar to 8 bits ones, but each pixel is, in this case, coded on 16 bits, which effect is to increase the levels of gray to 65,536. It is not possible to show the difference between a gray-level image quantized on 16 bits rather than 8. Under Windows, no display device is able to display 16-bit gray levels. Windows doesn't allow you to display more than 256 gray levels.

Namespace: Euresys::Open_eVision

Properties

Value	The value of the pixel.

Methods

EBW16	Constructs a EBW16 object.
GetSize	The number of bits in a pixel

EBW16::EBW16

Constructs a EBW16 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW16(
 )

void EBW16(
 OEV_UINT16 value
 )
```

Parameters

value

The value of the pixel.

EBW16::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

```
[C++]
int GetSize()
```

EBW16::Value

The value of the pixel.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 Value
```

5.6. EBW16Path Struct

Path from a EBW16 image: image pixel coordinates, and associated gray-level pixel value.

Namespace: Euresys::Open_eVision

Properties

X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.
Pixel	The value of the pixel at this coordinate.

EBW16Path::Pixel

The value of the pixel at this coordinate.

Namespace: Euresys::Open_eVision

```
[C++]
EBW16 Pixel
```

EBW16Path::X

Coordinate along the horizontal direction.

Namespace: Euresys::Open_eVision

```
[C++]
int X
```

EBW16Path::Y

Coordinate along the vertical direction.

Namespace: Euresys::Open_eVision

```
[C++]
int Y
```

5.7. EBW32 Struct

Gray-level pixel value, coded as an unsigned 32-bit integer.

Namespace: Euresys::Open_eVision

Properties

Value	The value of the pixel.

Methods

EBW32	Constructs a EBW32 object.
GetSize	The number of bits in a pixel

EBW32::EBW32

Constructs a EBW32 object.

Namespace: Euresys::Open_eVision

```
[C++]
void EBW32(
 )
void EBW32(
 OEV_UINT32 value
 )
```

Parameters

value

The value of the pixel.

EBW32::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

```
[C++]
int GetSize()
```

EBW32::Value

The value of the pixel.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT32 Value
```

5.8. EBW32f Struct

Gray-level pixel value, coded as a 32-bit floating-point number.

Namespace: Euresys::Open_eVision

Properties

Value	The depth value of the pixel.

Methods

EBW32f	Constructs a EBW32f object.
GetSize	The number of bits in a pixel

EBW32f::EBW32f

Constructs a EBW32f object.

Namespace: Euresys::Open_eVision

```
[C++]
void EBW32f(
 )
void EBW32f(
 float value
 )
```

Parameters

value

The depth value of the pixel.

EBW32f::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

```
[C++]
int GetSize()
```

EBW32f::Value

The depth value of the pixel.

Namespace: Euresys::Open_eVision

```
[C++]
float Value
```

5.9. EBW8 Struct

Gray-level pixel value, coded as an unsigned 8-bit integer.

Remarks

Every pixel is coded on 8 bits, allowing to represent 256 different values. The value 0 stands for black (background) and the value 255 stands for white (foreground). The 254 remaining values stand for shades of gray. This is sufficient for most applications. Most of the Open eVision gray-level operations apply to this pixel type.

Namespace: Euresys::Open_eVision

Properties

Value	The value of the pixel.

Methods

EBW8	Constructs a EBW8 object.
GetSize	The number of bits in a pixel

EBW8::EBW8

Constructs a EBW8 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EBW8(
 )
void EBW8(
 OEV_UINT8 value
 )
```

Parameters

value

The value of the pixel.

EBW8::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

EBW8::Value

The value of the pixel.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 Value
```

5.10. EBW8Path Struct

Path from a EBW8 image: image pixel coordinates, and associated gray-level pixel value.

Namespace: Euresys::Open_eVision

Properties

X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.
Pixel	The value of the pixel at this coordinate.

EBW8Path::Pixel

The value of the pixel at this coordinate.

Namespace: Euresys::Open_eVision

```
[C++]
EBW8 Pixel
```

EBW8Path::X

Coordinate along the horizontal direction.

Namespace: Euresys::Open_eVision

```
[C++]
int X
```

EBW8Path::Y

Coordinate along the vertical direction.

Namespace: Euresys::Open_eVision

```
[C++]
int Y
```

5.11. EC15 Struct

Color pixel value, coded as 3 fields of 5 bits each (red, green, blue components) and 1 field of 1 bit for padding.

Remarks

This class is suited to handle the Windows RGB15 color images. The pixel values are coded on 15 bits, leaving 32 possible levels per color component (red, green or blue).

Namespace: Euresys::Open_eVision

Properties

C2	The value of the third component of the pixel (blue channel).
C1	The value of the second component of the pixel (green channel).
C0	The value of the first component of the pixel (red channel).

Methods

EC15	Constructs a EC15 object.
GetSize	The number of bits in a pixel

EC15::C0

The value of the first component of the pixel (red channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 C0
```

EC15::C1

The value of the second component of the pixel (green channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 C1
```

EC15::C2

The value of the third component of the pixel (blue channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 C2
```

EC15::EC15

Constructs a EC15 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EC15(
 )

void EC15(
 OEV_UINT8 c0,
 OEV_UINT8 c1,
 OEV_UINT8 c2
 )
```

Parameters

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

EC15::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

5.12. EC16 Struct

Color pixel value, coded as 3 fields of 5 bits, 6 bits and 5 bits (red, green and blue components).

Remarks

This class is suited to handle the Windows RGB16 color images. The pixel values are coded on 16 bits (5-6-5), leaving 32 possible levels for R and B components, and 64 possible levels for G component.

Namespace: Euresys::Open_eVision

Properties

C2	The value of the third component of the pixel (blue channel).
C1	The value of the second component of the pixel (green channel).
C0	The value of the first component of the pixel (red channel).

Methods

EC16	Constructs a EC16 object.
GetSize	The number of bits in a pixel

EC16::C0

The value of the first component of the pixel (red channel).

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 C0
```

EC16::C1

The value of the second component of the pixel (green channel).

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 C1
```

EC16::C2

The value of the third component of the pixel (blue channel).

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 C2
```

EC16::EC16

Constructs a EC16 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EC16(
 )
void EC16(
 OEV_UINT8 c0,
 OEV_UINT8 c1,
 OEV_UINT8 c2
 )
```

Parameters

c0
The value of the first component of the pixel (red channel).

c1
The value of the second component of the pixel (green channel).

c2
The value of the third component of the pixel (blue channel).

EC16::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

5.13. EC24 Struct

Color pixel value coded as 3 unsigned 8-bit integers (red, green and blue components).

Remarks

(RGB triplet, windows 24 bpp bitmap format) The pixel values are coded on 24 bits, providing 256 possible levels per color component. This way, RGB images can represent 16,777,216 different colors. This is sufficient for most applications. Most of the Open eVision color operations apply to this pixel type.

Namespace: Euresys::Open_eVision

Properties

C2	The value of the third component of the pixel (blue channel).
C1	The value of the second component of the pixel (green channel).
C0	The value of the first component of the pixel (red channel).

Methods

EC24	Constructs a EC24 object.
GetSize	The number of bits in a pixel

EC24::C0

The value of the first component of the pixel (red channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 C0
```

EC24::C1

The value of the second component of the pixel (green channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 C1
```

EC24::C2

The value of the third component of the pixel (blue channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 C2
```

EC24::EC24

Constructs a EC24 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EC24(
 )
```

```
void EC24(
 const ERGBColor& rgbColor
 )
void EC24(
 OEV_UINT8 c0,
 OEV_UINT8 c1,
 OEV_UINT8 c2
 )
```

Parameters

rgbColor

-

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

EC24::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

5.14. EC24A Struct

Color pixel value coded as 4 unsigned 8-bit integers (red, green, blue and alpha components).

Remarks

This class is suited to handle the Windows RGB32 color format. The pixel values are coded on 32 bits, leaving 256 possible levels per color component (red, green or blue), and 8 more bits for an alpha channel. Currently, the alpha channel is not used for any purpose in the Open eVision processing functions. Users are free to use it to store an additional gray-level content.

Namespace: Euresys::Open_eVision

Properties

C2	The value of the third component of the pixel (blue channel).
C1	The value of the second component of the pixel (green channel).
C0	The value of the first component of the pixel (red channel).

A	The value of the alpha component of the pixel.

Methods

EC24A	Constructs a EC24A object.
GetSize	The number of bits in a pixel

EC24A::A

The value of the alpha component of the pixel.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 A
```

EC24A::C0

The value of the first component of the pixel (red channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 C0
```

EC24A::C1

The value of the second component of the pixel (green channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 C1
```

EC24A::C2

The value of the third component of the pixel (blue channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 C2
```

EC24A::EC24A

Constructs a EC24A object.

Namespace: Euresys::Open_eVision

```
[C++]
void EC24A(
 )
void EC24A(
 OEV_UINT8 c0,
 OEV_UINT8 c1,
 OEV_UINT8 c2,
 OEV_UINT8 a
 )
```

Parameters

c0

The value of the first component of the pixel (red channel).

c1

The value of the second component of the pixel (green channel).

c2

The value of the third component of the pixel (blue channel).

a

-

EC24A::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

```
[C++]
int GetSize()
```

5.15. EC24Path Struct

Path from a EC24 image: image pixel coordinates, and associated color pixel value.

Namespace: Euresys::Open_eVision

Properties

X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.

Pixel	The value of the pixel at this coordinate.

EC24Path::Pixel

The value of the pixel at this coordinate.

Namespace: Euresys::Open_eVision

```
[C++]
EC24 Pixel
```

EC24Path::X

Coordinate along the horizontal direction.

Namespace: Euresys::Open_eVision

```
[C++]
int X
```

EC24Path::Y

Coordinate along the vertical direction.

Namespace: Euresys::Open_eVision

```
[C++]
int Y
```

5.16. EC48 Struct

Color pixel value coded as 3 unsigned 16-bit integers (red, green, blue components).

Namespace: Euresys::Open_eVision

Properties

C2	The value of the third component of the pixel (blue channel).
C1	The value of the second component of the pixel (green channel).
C0	The value of the first component of the pixel (red channel).

Methods

EC48	Constructs a EC48 object.
GetSize	The size of a pixel.

EC48::C0

The value of the first component of the pixel (red channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 C0
```

EC48::C1

The value of the second component of the pixel (green channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 C1
```

EC48::C2

The value of the third component of the pixel (blue channel).

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT16 C2
```

EC48::EC48

Constructs a EC48 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EC48(
 )
```

```
void EC48(
 OEV_UINT16 c0,
 OEV_UINT16 c1,
 OEV_UINT16 c2
 )
```

Parameters

c0
The value of the first component of the pixel (red channel).

c1
The value of the second component of the pixel (green channel).

c2
The value of the third component of the pixel (blue channel).

EC48::GetSize

The size of a pixel.

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

5.17. EColor Struct

Triple of floating-point numbers that encode a color in a given color system.

Namespace: Euresys::Open_eVision

Properties

C0	First color component.
C1	Second color component.
C2	Third color component.

Methods

EColor	Constructor for EColor objects.

EColor::C0

First color component.

Namespace: Euresys::Open_eVision

[C++]

```
float C0
```

EColor::C1

Second color component.

Namespace: Euresys::Open_eVision

[C++]

```
float C1
```

EColor::C2

Third color component.

Namespace: Euresys::Open_eVision

[C++]

```
float C2
```

EColor::EColor

Constructor for EColor objects.

Namespace: Euresys::Open_eVision

[C++]

```
void EColor(
 )

void EColor(
 float c0,
 float c1,
 float c2
 )
```

Parameters

c0
value for the first color component

c1
value for the second color component

c2
value for the third color component

5.18. EDepth16 Struct

Depth value of the pixel, coded as an unsigned 16-bit integer.

Namespace: Euresys::Open_eVision

Properties

Value	The depth value of the pixel.

Methods

EDepth16	Constructs a EDepth16 object.
GetSize	The number of bits in a pixel

EDepth16::EDepth16

Constructs a EDepth16 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EDepth16(
 )

void EDepth16(
 OEV_UINT16 value
 )
```

Parameters

value

The depth value of the pixel.

EDepth16::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

EDepth16::Value

The depth value of the pixel.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 Value
```

5.19. EDepth32f Struct

Depth value of the pixel, coded as a 32-bits floating point value.

Namespace: Euresys::Open_eVision

Properties

Value	The depth value of the pixel.

Methods

EDepth32f	Constructs a EDepth32f object.
GetSize	The number of bits in a pixel

EDepth32f::EDepth32f

Constructs a EDepth32f object.

Namespace: Euresys::Open_eVision

```
[C++]
void EDepth32f(
 )
void EDepth32f(
 float value
 )
```

Parameters

value
The depth value of the pixel.

EDepth32f::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

[C++]

```
int GetSize()
```

EDepth32f::Value

The depth value of the pixel.

Namespace: Euresys::Open_eVision

[C++]

```
float Value
```

5.20. EDepth8 Struct

Depth value of the pixel, coded as an unsigned 8-bit integer.

Namespace: Euresys::Open_eVision

Properties

Value	The depth value of the pixel.

Methods

EDepth8	Constructs a EDepth8 object.
GetSize	The number of bits in a pixel

EDepth8::EDepth8

Constructs a EDepth8 object.

Namespace: Euresys::Open_eVision

[C++]

```
void EDepth8(
 )

void EDepth8(
 OEV_UINT8 value
 )
```

Parameters

value

The depth value of the pixel.

EDepth8::GetSize

The number of bits in a pixel

Namespace: Euresys::Open_eVision

```
[C++]
int GetSize()
```

EDepth8::Value

The depth value of the pixel.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT8 Value
```

5.21. EFeatureData Struct

This struct is deprecated.

Describes object features.

Remarks

A feature is associated to an array of values, each corresponding to an object of given identification number. A feature is also characterized by the size of the array, a feature number, data size/type information and pointers to both ends of the array. The features can be accessed by their number (see EFeature). To obtain the value of a given feature of a given object, just use the class member ECodedImage::GetObjectFeature. This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See ECodedImage2 for the new API.

Namespace: Euresys::Open_eVision

Properties

Size	Number of objects for which the feature is stored.
FeatNum	Code (see EFeature).
FeatDataType	Data type (see EDataType).
FeatDataSize	Data size (see EDataSize).

EFeatureData::FeatDataSize

This struct member is deprecated.

Data size (see EDataSize).

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EDataSize FeatDataSize
```

EFeatureData::FeatDataType

This struct member is deprecated.

Data type (see EDataType).

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::EDataType FeatDataType
```

EFeatureData::FeatNum

This struct member is deprecated.

Code (see EFeature).

Namespace: Euresys::Open_eVision

```
[C++]
Euresys::Open_eVision::ELegacyFeature FeatNum
```

EFeatureData::Size

This struct member is deprecated.

Number of objects for which the feature is stored.

Namespace: Euresys::Open_eVision

```
[C++]
int Size
```

5.22. EISH Struct

Intensity, Saturation, Hue color system.

Namespace: Euresys::Open_eVision

Properties

I	Intensity component.
S	Saturation component.
H	Hue component.

EISH::H

Hue component.

Namespace: Euresys::Open_eVision

```
[C++]
float H
```

EISH::I

Intensity component.

Namespace: Euresys::Open_eVision

```
[C++]
float I
```

EISH::S

Saturation component.

Namespace: Euresys::Open_eVision

```
[C++]
float S
```

5.23. ELAB Struct

CIE Lightness, a*, b* color system.

Namespace: Euresys::Open_eVision

Properties

L	Lightness component.
A	a* component.

B	b* component.

ELAB::A

a* component.

Namespace: Euresys::Open_eVision

```
[C++]
float A
```

ELAB::B

b* component.

Namespace: Euresys::Open_eVision

```
[C++]
float B
```

ELAB::L

Lightness component.

Namespace: Euresys::Open_eVision

```
[C++]
float L
```

5.24. ELCH Struct

Lightness, Chroma, Hue color system.

Namespace: Euresys::Open_eVision

Properties

L	Lightness component.
C	Chroma component.
H	Hue component.

ELCH::C

Chroma component.

Namespace: Euresys::Open_eVision

```
[C++]
float C
```

ELCH::H

Hue component.

Namespace: Euresys::Open_eVision

```
[C++]
float H
```

ELCH::L

Lightness component.

Namespace: Euresys::Open_eVision

```
[C++]
float L
```

5.25. ELSH Struct

Lightness, Saturation, Hue color system.

Namespace: Euresys::Open_eVision

Properties

L	Lightness component.
S	Saturation component.
H	Hue component.

ELSH::H

Hue component.

Namespace: Euresys::Open_eVision

```
[C++]
float H
```

ELSH::L

Lightness component.

Namespace: Euresys::Open_eVision

```
[C++]
float L
```

ELSH::S

Saturation component.

Namespace: Euresys::Open_eVision

```
[C++]
float S
```

5.26. ELUV Struct

CIE Lightness, u*, v* color system.

Namespace: Euresys::Open_eVision

Properties

L	Lightness component.
U	u* component.
V	v* component.

ELUV::L

Lightness component.

Namespace: Euresys::Open_eVision

[C++]

```
float L
```

ELUV::U

u* component.

Namespace: Euresys::Open_eVision

[C++]

```
float U
```

ELUV::V

v* component.

Namespace: Euresys::Open_eVision

[C++]

```
float V
```

5.27. EMatchPosition Struct

Represents a single instance of the pattern in the search field, as returned by the EasyMatch matching process.

Remarks

EMatcher::GetPosition returns instances of this class. A EMatchPosition object represents one matched instance, with all the needed information about it.

Namespace: Euresys::Open_eVision

Properties

CenterX	Abscissa of the center of the pattern found in the image.
CenterY	Ordinate of the center of the pattern found in the image.
Angle	Clockwise rotation angle, expressed using the current angle unit, of the pattern found in the image.
Scale	Scale factor of the pattern found in the image.
ScaleX	Measured horizontal scaling of the pattern found in the image, expressed as a dimensionless ratio.
ScaleY	Measured vertical scaling of the pattern found in the image, expressed as a dimensionless ratio.

Score	Indicates how good a matching was.
Interpolated	Indicates whether sub-pixel interpolation was actually performed on the position.
AreaRatio	Ratio between the found pattern area inside the search ROI and its complete area.

Methods

ToRegion	Creates an ERegion from the EMatchPosition. The ERegion represents all the pixels which are within the bounding box of the EMatchPosition.

EMatchPosition::Angle

Clockwise rotation angle, expressed using the current angle unit, of the pattern found in the image.

Namespace: Euresys::Open_eVision

[C++]

```
float Angle
```

Remarks

0 if no rotation is allowed.

EMatchPosition::AreaRatio

Ratio between the found pattern area inside the search ROI and its complete area.

Namespace: Euresys::Open_eVision

[C++]

```
float AreaRatio
```

EMatchPosition::CenterX

Abscissa of the center of the pattern found in the image.

Namespace: Euresys::Open_eVision

[C++]

```
float CenterX
```

EMatchPosition::CenterY

Ordinate of the center of the pattern found in the image.

Namespace: Euresys::Open_eVision

```
[C++]
float CenterY
```

EMatchPosition::Interpolated

Indicates whether sub-pixel interpolation was actually performed on the position.

Namespace: Euresys::Open_eVision

```
[C++]
bool Interpolated
```

Remarks

In some cases, when the pattern is found close to the ROI edge, sub-pixel interpolation cannot be used.

EMatchPosition::Scale

Scale factor of the pattern found in the image.

Namespace: Euresys::Open_eVision

```
[C++]
float Scale
```

Remarks

1 if no scaling is allowed.

EMatchPosition::ScaleX

Measured horizontal scaling of the pattern found in the image, expressed as a dimensionless ratio.

Namespace: Euresys::Open_eVision

```
[C++]
float ScaleX
```

EMatchPosition::ScaleY

Measured vertical scaling of the pattern found in the image, expressed as a dimensionless ratio.

Namespace: Euresys::Open_eVision

[C++]

```
float ScaleY
```

EMatchPosition::Score

Indicates how good a matching was.

Namespace: Euresys::Open_eVision

[C++]

```
float Score
```

Remarks

1 means that the matching was perfect. Lower values correspond to approximate matching; -1 corresponds to a perfect mismatch (pattern superimposed on its negative image).

EMatchPosition::ToRegion

Creates an ERegion from the EMatchPosition. The ERegion represents all the pixels which are within the bounding box of the EMatchPosition.

Namespace: Euresys::Open_eVision

[C++]

```
ERegion ToRegion(
 int modelWidth,
 int modelHeight
 )
```

Parameters

modelWidth

Width of the corresponding EasyMatch model.

modelHeight

Height of the corresponding EasyMatch model.

5.28. EMatrixCodeIso15415GradingParameters Struct

Holds all grading parameters pertaining to ISO/IEC 15415

Namespace: Euresys::Open_eVision

Properties

DecodingGrade	-
AxialNonUniformityGrade	Axial Non Uniformity Grade
GridNonUniformityGrade	Grid Non Uniformity Grade
UnusedErrorCorrectionGrade	Unused Error Correction Grade
SymbolContrastGrade	Symbol Contrast Grade
ModulationGrade	Modulation Grade
ReflectanceMarginGrade	Reflectance Margin Grade
FixedPatternDamageGrade	Fixed Pattern Damage Grade
OverallSymbolGrade	Overall Symbol Grade
ScanGrade	Scan Grade
HorizontalPrintGrowth	Horizontal Print Growth
VerticalPrintGrowth	Vertical Print Growth
GridNonUniformity	Grid Non Uniformity
AxialNonUniformity	Axial Non Uniformity
SymbolContrast	Symbol Contrast
UnusedErrorCorrection	Unused Error Correction

Methods

EMatrixCodeIso15415GradingParameters	-

EMatrixCodeIso15415GradingParameters::AxialNonUniformity

Axial Non Uniformity

Namespace: Euresys::Open_eVision

[C++]

```
float AxialNonUniformity
```

EMatrixCodeIso15415GradingParameters::AxialNonUniformityGrade

Axial Non Uniformity Grade

Namespace: Euresys::Open_eVision

[C++]

```
int AxialNonUniformityGrade
```

EMatrixCodeIso15415GradingParameters::DecodingGrade

-

Namespace: Euresys::Open_eVision

[C++]

```
int DecodingGrade
```

EMatrixCodeIso15415GradingParameters::EMatrixCodeIso15415GradingParameters

-

Namespace: Euresys::Open_eVision

[C++]

```
void EMatrixCodeIso15415GradingParameters(
 )
```

EMatrixCodeIso15415GradingParameters::FixedPatternDamageGrade

Fixed Pattern Damage Grade

Namespace: Euresys::Open_eVision

[C++]

```
int FixedPatternDamageGrade
```

EMatrixCodeIso15415GradingParameters::GridNonUniformity

Grid Non Uniformity

Namespace: Euresys::Open_eVision

```
[C++]
float GridNonUniformity
```

EMatrixCodeIso15415GradingParameters::GridNonUniformityGrade

Grid Non Uniformity Grade

Namespace: Euresys::Open_eVision

```
[C++]
int GridNonUniformityGrade
```

EMatrixCodeIso15415GradingParameters::HorizontalPrintGrowth

Horizontal Print Growth

Namespace: Euresys::Open_eVision

```
[C++]
float HorizontalPrintGrowth
```

EMatrixCodeIso15415GradingParameters::ModulationGrade

Modulation Grade

Namespace: Euresys::Open_eVision

```
[C++]
int ModulationGrade
```

EMatrixCodeIso15415GradingParameters::OverallSymbolGrade

This struct member is deprecated.

Overall Symbol Grade

Namespace: Euresys::Open_eVision

[C++]

```
int OverallSymbolGrade
```

EMatrixCodeIso15415GradingParameters::ReflectanceMarginGrade

Reflectance Margin Grade

Namespace: Euresys::Open_eVision

[C++]

```
int ReflectanceMarginGrade
```

EMatrixCodeIso15415GradingParameters::ScanGrade

Scan Grade

Namespace: Euresys::Open_eVision

[C++]

```
int ScanGrade
```

EMatrixCodeIso15415GradingParameters::SymbolContrast

Symbol Contrast

Namespace: Euresys::Open_eVision

[C++]

```
float SymbolContrast
```

EMatrixCodeIso15415GradingParameters::SymbolContrastGrade

Symbol Contrast Grade

Namespace: Euresys::Open_eVision

[C++]

```
int SymbolContrastGrade
```

EMatrixCodeIso15415GradingParameters::UnusedErrorCorrection

Unused Error Correction

Namespace: Euresys::Open_eVision

```
[C++]
float UnusedErrorCorrection
```

EMatrixCodeIso15415GradingParameters::UnusedErrorCorrectionGrade

Unused Error Correction Grade

Namespace: Euresys::Open_eVision

```
[C++]
int UnusedErrorCorrectionGrade
```

EMatrixCodeIso15415GradingParameters::VerticalPrintGrowth

Vertical Print Growth

Namespace: Euresys::Open_eVision

```
[C++]
float VerticalPrintGrowth
```

5.29. EMatrixCodeIso29158CalibrationParameters Struct

Holds all grading inputs pertaining to ISO/IEC 29158

Namespace: Euresys::Open_eVision

Properties

Rcal	Reported reflectance value, from a calibration standard.
MLcal	Mean of the light from a histogram of the calibrated standard.
SRcal	System response parameters(such as exposure and/or gain) used to create an image of the calibration standard.
SRtarget	System response parameters(such as exposure and/or gain) used to create an image of the symbole under test.

Methods

EMatrixCodeIso29158CalibrationParameters	-

EMatrixCodeIso29158CalibrationParameters::EMatrixCodeIso29158CalibrationParameters

-

Namespace: Euresys::Open_eVision

```
[C++]
void EMatrixCodeIso29158CalibrationParameters(
 )
```

EMatrixCodeIso29158CalibrationParameters::MLcal

Mean of the light from a histogram of the calibrated standard.

Namespace: Euresys::Open_eVision

```
[C++]
float MLcal
```

EMatrixCodeIso29158CalibrationParameters::Rcal

Reported reflectance value, from a calibration standard.

Namespace: Euresys::Open_eVision

```
[C++]
float Rcal
```

EMatrixCodeIso29158CalibrationParameters::SRcal

System response parameters(such as exposure and/or gain) used to create an image of the calibration standard.

Namespace: Euresys::Open_eVision

```
[C++]
float SRcal
```

EMatrixCodeIso29158CalibrationParameters::SRtarget

System response parameters(such as exposure and/or gain) used to create an image of the symbole under test.

Namespace: Euresys::Open_eVision

```
[C++]
float SRtarget
```

5.30. EMatrixCodeIso29158GradingParameters Struct

Holds all grading parameters pertaining to ISO/IEC 29158

Namespace: Euresys::Open_eVision

Properties

CellContrastGrade	-
CellModulationGrade	Cell Modulation Grade
MinimumReflectanceGrade	Minimum Reflectance Grade
FixedPatternDamageGrade	FixedPatternDamageGrade
OverallSymbolGrade	Overall Symbol Grade
ScanGrade	Scan Grade
MeanLight	Mean Light
IsMeanLightInRequiredBounds	Is Mean Light In Required Bounds

Methods

EMatrixCodeIso29158GradingParameters	-

EMatrixCodeIso29158GradingParameters::CellContrastGrade

-

Namespace: Euresys::Open_eVision

[C++]

```
int CellContrastGrade
```

EMatrixCodeIso29158GradingParameters::CellModulationGrade

Cell Modulation Grade

Namespace: Euresys::Open_eVision

[C++]

```
int CellModulationGrade
```

EMatrixCodeIso29158GradingParameters::EMatrixCodeIso29158GradingParameters

-

Namespace: Euresys::Open_eVision

[C++]

```
void EMatrixCodeIso29158GradingParameters(
 )
```

EMatrixCodeIso29158GradingParameters::FixedPatternDamageGrade

FixedPatternDamageGrade

Namespace: Euresys::Open_eVision

[C++]

```
int FixedPatternDamageGrade
```

EMatrixCodeIso29158GradingParameters::IsMeanLightInRequiredBounds

Is Mean Light In Required Bounds

Namespace: Euresys::Open_eVision

[C++]

```
bool IsMeanLightInRequiredBounds
```

EMatrixCodeIso29158GradingParameters::MeanLight

Mean Light

Namespace: Euresys::Open_eVision

```
[C++]
float MeanLight
```

EMatrixCodeIso29158GradingParameters::MinimumReflectanceGrade

Minimum Reflectance Grade

Namespace: Euresys::Open_eVision

```
[C++]
int MinimumReflectanceGrade
```

EMatrixCodeIso29158GradingParameters::OverallSymbolGrade

This struct member is deprecated.

Overall Symbol Grade

Namespace: Euresys::Open_eVision

```
[C++]
int OverallSymbolGrade
```

EMatrixCodeIso29158GradingParameters::ScanGrade

Scan Grade

Namespace: Euresys::Open_eVision

```
[C++]
int ScanGrade
```

5.31. EMatrixCodeSemiT10GradingParameters Struct

Holds all grading parameters pertaining to Semi T10-

Namespace: Euresys::Open_eVision

Properties

SymbolContrast	Symbol Contrast
SymbolContrastSNR	Symbol Contrast SNR
HorizontalMarkGrowth	Horizontal Mark Growth
VerticalMarkGrowth	Vertical Mark Growth
DataMatrixCellWidth	Data Matrix Cell Width
DataMatrixCellHeight	Data Matrix Cell Height
HorizontalMarkMisplacement	Horizontal Mark Misplacement
VerticalMarkMisplacement	Vertical Mark Misplacement
UnusedErrorCorrection	Unused Error Correction
CellDefects	Cell Defects
FinderPatternDefects	Finder Pattern Defects

Methods

EMatrixCodeSemiT10GradingParameters	-

EMatrixCodeSemiT10GradingParameters::CellDefects

Cell Defects

Namespace: Euresys::Open_eVision

[C++]

```
float CellDefects
```

EMatrixCodeSemiT10GradingParameters::DataMatrixCellHeight

Data Matrix Cell Height

Namespace: Euresys::Open_eVision

[C++]

```
float DataMatrixCellHeight
```

EMatrixCodeSemiT10GradingParameters::DataMatrixCellWidth

Data Matrix Cell Width

Namespace: Euresys::Open_eVision

```
[C++]
float DataMatrixCellWidth
```

EMatrixCodeSemiT10GradingParameters::EMatrixCodeSemiT10GradingParameters

-

Namespace: Euresys::Open_eVision

```
[C++]
void EMatrixCodeSemiT10GradingParameters(
 )
```

EMatrixCodeSemiT10GradingParameters::FinderPatternDefects

Finder Pattern Defects

Namespace: Euresys::Open_eVision

```
[C++]
float FinderPatternDefects
```

EMatrixCodeSemiT10GradingParameters::HorizontalMarkGrowth

Horizontal Mark Growth

Namespace: Euresys::Open_eVision

```
[C++]
float HorizontalMarkGrowth
```

EMatrixCodeSemiT10GradingParameters::HorizontalMarkMisplacement

Horizontal Mark Misplacement

Namespace: Euresys::Open_eVision

[C++]

```
float HorizontalMarkMisplacement
```

EMatrixCodeSemiT10GradingParameters::SymbolContrast

Symbol Contrast

Namespace: Euresys::Open_eVision

[C++]

```
float SymbolContrast
```

EMatrixCodeSemiT10GradingParameters::SymbolContrastSNR

Symbol Contrast SNR

Namespace: Euresys::Open_eVision

[C++]

```
float SymbolContrastSNR
```

EMatrixCodeSemiT10GradingParameters::UnusedErrorCorrection

Unused Error Correction

Namespace: Euresys::Open_eVision

[C++]

```
float UnusedErrorCorrection
```

EMatrixCodeSemiT10GradingParameters::VerticalMarkGrowth

Vertical Mark Growth

Namespace: Euresys::Open_eVision

[C++]

```
float VerticalMarkGrowth
```

EMatrixCodeSemiT10GradingParameters::VerticalMarkMisplacement

Vertical Mark Misplacement

Namespace: Euresys::Open_eVision

```
[C++]
float VerticalMarkMisplacement
```

5.32. EMatrixPosition Struct

Represents a position in a 2D Matrix.

Namespace: Euresys::Open_eVision

Properties

X	X position.
Y	Y position.

Methods

EMatrixPosition	Constructs a default EMatrixPosition object.
operator!=	Checks if two EMatrixPosition are stricly different
operator==	Checks if two EMatrixPosition are stricly equal

EMatrixPosition::EMatrixPosition

Constructs a default EMatrixPosition object.

Namespace: Euresys::Open_eVision

```
[C++]
void EMatrixPosition(
 )

void EMatrixPosition(
 int x,
 int y
 )
```

Parameters

x

X position.

y

Y position.

Remarks

If the default constructor is used, the position is initialized to (0, 0).

EMatrixPosition::operator!=

Checks if two EMatrixPosition are stricly different

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const EMatrixPosition& position
 )
```

Parameters

position

The other position.

EMatrixPosition::operator==

Checks if two EMatrixPosition are stricly equal

Namespace: Euresys::Open_eVision

[C++]

```
bool operator==(
 const EMatrixPosition& position
 )
```

Parameters

position

The other position.

EMatrixPosition::X

X position.

Namespace: Euresys::Open_eVision

[C++]

```
int X
```

EMatrixPosition::Y

Y position.

Namespace: Euresys::Open_eVision

[C++]

```
int Y
```

5.33. EObjectData Struct

This struct is deprecated.

Describes objects.

Remarks

An object is characterized by a class, a unique identification number, the number of its constituent runs, the number of its holes (if the object is a real object, not a hole), a selection flag, an identification flag (real object or hole) and the list of its constituent runs. After the object construction phase (real objects and eventually holes), all the objects are gathered in a single dynamic list. The objects can be accessed by their absolute position in the list as well as by their identification number. This structure pertains to the EasyObject legacy API and should not be used for new developments.

Note. After a sorting operation, the objects retain their identification number, not their absolute position in the list. If need be, the list of runs of an object can be traversed by means of the following functions: GetObjNbRun, ECodedImage::GetObjFirstRunPtr, ECodedImage::GetObjLastRunPtr.

Namespace: Euresys::Open_eVision

Properties

Class	Class code.
ObjNum	Identification number.
ObjNbRun	Number of runs.
ObjNbHole	Number of holes.
IsSelected	Selection flag.
IsHole	true if the object is a hole, false otherwise.

EObjectData::Class

This struct member is deprecated.

Class code.

Namespace: Euresys::Open_eVision

[C++]

```
int Class
```

EObjectData::IsHole

This struct member is deprecated.

true if the object is a hole, false otherwise.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsHole
```

EObjectData::IsSelected

This struct member is deprecated.

Selection flag.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT8 IsSelected
```

EObjectData::ObjNbHole

This struct member is deprecated.

Number of holes.

Namespace: Euresys::Open_eVision

[C++]

```
int ObjNbHole
```

EObjectData::ObjNbRun

This struct member is deprecated.

Number of runs.

Namespace: Euresys::Open_eVision

[C++]

```
int ObjNbRun
```

EObjectData::ObjNum

This struct member is deprecated.

Identification number.

Namespace: Euresys::Open_eVision

```
[C++]
int ObjNum
```

5.34. EOCR2CharacterCandidate Struct

Holds a single recognition score for a detected character from the image

Remarks

The variable "code" contains the ASCII-representation of the reference character from the database. The variable "score" contains the recognition score between the detected character and the reference character.

Namespace: Euresys::Open_eVision

Properties

Code	Contains the ASCII-representation of the reference character from the database.
Score	Contains the recognition score between the detected character and the reference character.

Methods

EOCR2CharacterCandidate	Constructs an EOCR2CharacterCandidate context.

EOCR2CharacterCandidate::Code

Contains the ASCII-representation of the reference character from the database.

Namespace: Euresys::Open_eVision

```
[C++]
OEV_UINT16 Code
```

EOCR2CharacterCandidate::EOCR2CharacterCandidate

Constructs an EOCR2CharacterCandidate context.

Namespace: Euresys::Open_eVision

[C++]

```
void EOCR2CharacterCandidate(
 )
void EOCR2CharacterCandidate(
 OEV_UINT16 code,
 float score
 )
```

Parameters

code

-

score

-

EOCR2CharacterCandidate::Score

Contains the recognition score between the detected character and the reference character.

Namespace: Euresys::Open_eVision

[C++]

```
float Score
```

5.35. EPath Struct

Path from an image: image pixel coordinates.

Namespace: Euresys::Open_eVision

Properties

X	Coordinate along the horizontal direction.
Y	Coordinate along the vertical direction.

EPath::X

Coordinate along the horizontal direction.

Namespace: Euresys::Open_eVision

```
[C++]
int X
```

EPath::Y

Coordinate along the vertical direction.

Namespace: Euresys::Open_eVision

```
[C++]
int Y
```

5.36. EPeak Struct

Represents a peak in a profile.

Namespace: Euresys::Open_eVision

Properties

Start	Start of the peak.
Length	Length of the peak.
Center	Center of the peak.
Amplitude	Amplitude of the peak.
Area	Area of the peak.

EPeak::Amplitude

Amplitude of the peak.

Namespace: Euresys::Open_eVision

```
[C++]
int Amplitude
```

EPeak::Area

Area of the peak.

Namespace: Euresys::Open_eVision

[C++]

```
int Area
```

EPeak::Center

Center of the peak.

Namespace: Euresys::Open_eVision

[C++]

```
float Center
```

EPeak::Length

Length of the peak.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 Length
```

EPeak::Start

Start of the peak.

Namespace: Euresys::Open_eVision

[C++]

```
OEV_UINT32 Start
```

5.37. EPen Struct

Pen.

Namespace: Euresys::Open_eVision

Properties

Brush	Brush used to draw the content of the pen.
Width	Width of the pen.
Style	Style of the pen.

Methods

EPen	Constructs an EPen.
IsValid	Whether the pen is valid, i.e. when its brush is valid and its width is bigger than 0.
Load	Loads the EPen. The given ESerializer must have been created for reading.
operator!=	Inequality operator.
operator==	Equality operator.
Save	Saves the EPen. The given ESerializer must have been created for writing.
Serialize	Serializes the EPen.

EPen::Brush

Brush used to draw the content of the pen.

Namespace: Euresys::Open_eVision

[C++]

```
EBrush Brush
```

EPen::EPen

Constructs an EPen.

Namespace: Euresys::Open_eVision

[C++]

```
void EPen(
 )

void EPen(
 const ERGBColor& color,
 int width,
 Euresys::Open_eVision::EPenStyle style
 )

void EPen(
 const ERGBColor& color,
 float opacity,
 int width,
 Euresys::Open_eVision::EPenStyle style
 )
```

```
void EPen(
 const EBrush& brush,
 int width,
 Euresys::Open_eVision::EPenStyle style
 )
```

Parameters

color
Color of the pen.
width
Width of the pen.
style
Style of the pen.
opacity
-
brush
Brush of the pen.

EPen::IsValid

Whether the pen is valid, i.e. when its brush is valid and its width is bigger than 0.

Namespace: Euresys::Open_eVision

[C++]

```
bool IsValid(
 )
```

EPen::Load

Loads the EPen. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision

[C++]

```
void Load(
 const std::string& path
 )
void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.

serializer
The serializer.

EPen::operator!=

Inequality operator.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const EPen& other
 )
```

Parameters

other
Other pen to compare with.

EPen::operator==

Equality operator.

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const EPen& other
 )
```

Parameters

other
Other pen to compare with.

EPen::Save

Saves the EPen. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision

[C++]

```
void Save(
 const std::string& path
 )
void Save(
 ESerializer* serializer
 )
```

Parameters

path

The file path.

serializer

The ESerializer object that is written to.

EPen::Serialize

Serializes the EPen.

Namespace: Euresys::Open_eVision

[C++]

```
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer

Serializer

EPen::Style

Style of the pen.

Namespace: Euresys::Open_eVision

[C++]

```
Euresys::Open_eVision::EPenStyle Style
```

EPen::Width

Width of the pen.

Namespace: Euresys::Open_eVision

[C++]

```
int Width
```

5.38. EQRCodeAdditionalParametersGrades Struct

Holds all grading inputs pertaining to ISO/IEC 29158

Namespace: Euresys::Open_eVision

Properties

VersionInformationGrade	Version Information Grade. Value is -1 if not available.
FormatInformationGrade	Format Information Grade.

Methods

EQRCodeAdditionalParametersGrades	-

EQRCodeAdditionalParametersGrades::EQRCodeAdditionalParametersGrades

-

Namespace: Euresys::Open_eVision

[C++]

```
void EQRCodeAdditionalParametersGrades(
 )
```

EQRCodeAdditionalParametersGrades::FormatInformationGrade

Format Information Grade.

Namespace: Euresys::Open_eVision

[C++]

```
int FormatInformationGrade
```

EQRCodeAdditionalParametersGrades::VersionInformationGrade

Version Information Grade. Value is -1 if not available.

Namespace: Euresys::Open_eVision

```
[C++]
int VersionInformationGrade
```

5.39. EQRCodeIso15415GradingParameters Struct

Holds all grading parameters pertaining to ISO/IEC 15415

Namespace: Euresys::Open_eVision

Properties

DecodingGrade	-
AxialNonUniformityGrade	Axial Non Uniformity Grade
GridNonUniformityGrade	Grid Non Uniformity Grade
UnusedErrorCorrectionGrade	Unused Error Correction Grade
SymbolContrastGrade	Symbol Contrast Grade
ModulationGrade	Modulation Grade
ReflectanceMarginGrade	Reflectance Margin Grade
FixedPatternDamageGrade	Fixed Pattern Damage Grade
OverallSymbolGrade	Overall Symbol Grade
ScanGrade	Scan Grade
HorizontalPrintGrowth	Horizontal Print Growth
VerticalPrintGrowth	Vertical Print Growth
GridNonUniformity	Grid Non Uniformity
AxialNonUniformity	Axial Non Uniformity
SymbolContrast	Symbol Contrast
UnusedErrorCorrection	Unused Error Correction
AdditionalParametersGrades	Additional Parameters Grades

Methods

EQRCodeIso15415Gradi - ngParameters

EQRCodeIso15415GradingParameters::AdditionalParametersGrades

Additional Parameters Grades

Namespace: Euresys::Open_eVision

[C++]

```
EQRCodeAdditionalParametersGrades AdditionalParametersGrades
```

EQRCodeIso15415GradingParameters::AxialNonUniformity

Axial Non Uniformity

Namespace: Euresys::Open_eVision

[C++]

```
float AxialNonUniformity
```

EQRCodeIso15415GradingParameters::AxialNonUniformityGrade

Axial Non Uniformity Grade

Namespace: Euresys::Open_eVision

[C++]

```
int AxialNonUniformityGrade
```

EQRCodeIso15415GradingParameters::DecodingGrade

-

Namespace: Euresys::Open_eVision

[C++]

```
int DecodingGrade
```

EQRCodeIso15415GradingParameters::EQRCodeIso15415GradingParameters

-

Namespace: Euresys::Open_eVision

```
[C++]
void EQRCodeIso15415GradingParameters(
 )
```

EQRCodeIso15415GradingParameters::FixedPatternDamageGrade

Fixed Pattern Damage Grade

Namespace: Euresys::Open_eVision

```
[C++]
int FixedPatternDamageGrade
```

EQRCodeIso15415GradingParameters::GridNonUniformity

Grid Non Uniformity

Namespace: Euresys::Open_eVision

```
[C++]
float GridNonUniformity
```

EQRCodeIso15415GradingParameters::GridNonUniformityGrade

Grid Non Uniformity Grade

Namespace: Euresys::Open_eVision

```
[C++]
int GridNonUniformityGrade
```

EQRCodeIso15415GradingParameters::HorizontalPrintGrowth

Horizontal Print Growth

Namespace: Euresys::Open_eVision

[C++]

```
float HorizontalPrintGrowth
```

EQRCodeIso15415GradingParameters::ModulationGrade

Modulation Grade

Namespace: Euresys::Open_eVision

[C++]

```
int ModulationGrade
```

EQRCodeIso15415GradingParameters::OverallSymbolGrade

This struct member is deprecated.

Overall Symbol Grade

Namespace: Euresys::Open_eVision

[C++]

```
int OverallSymbolGrade
```

EQRCodeIso15415GradingParameters::ReflectanceMarginGrade

Reflectance Margin Grade

Namespace: Euresys::Open_eVision

[C++]

```
int ReflectanceMarginGrade
```

EQRCodeIso15415GradingParameters::ScanGrade

Scan Grade

Namespace: Euresys::Open_eVision

[C++]

```
int ScanGrade
```

EQRCodeIso15415GradingParameters::SymbolContrast

Symbol Contrast

Namespace: Euresys::Open_eVision

```
[C++]
float SymbolContrast
```

EQRCodeIso15415GradingParameters::SymbolContrastGrade

Symbol Contrast Grade

Namespace: Euresys::Open_eVision

```
[C++]
int SymbolContrastGrade
```

EQRCodeIso15415GradingParameters::UnusedErrorCorrection

Unused Error Correction

Namespace: Euresys::Open_eVision

```
[C++]
float UnusedErrorCorrection
```

EQRCodeIso15415GradingParameters::UnusedErrorCorrectionGrade

Unused Error Correction Grade

Namespace: Euresys::Open_eVision

```
[C++]
int UnusedErrorCorrectionGrade
```

EQRCodeIso15415GradingParameters::VerticalPrintGrowth

Vertical Print Growth

Namespace: Euresys::Open_eVision

[C++]

```
float VerticalPrintGrowth
```

5.40. EQRCodeIso29158CalibrationParameters Struct

Holds all grading inputs pertaining to ISO/IEC 29158

Namespace: Euresys::Open_eVision

Properties

Rcal	Reported reflectance value, from a calibration standard.
MLcal	Mean of the light from a histogram of the calibrated standard.
SRcal	System response parameters(such as exposure and/or again) used to create an image of the calibration standard.
SRtarget	System response parameters(such as exposure and/or again) used to create an image of the symbole under test.

Methods

EQRCodeIso29158CalibrationParameters	-

EQRCodeIso29158CalibrationParameters::EQRCodeIso29158CalibrationParameters

-

Namespace: Euresys::Open_eVision

[C++]

```
void EQRCodeIso29158CalibrationParameters(
 )
```

EQRCodeIso29158CalibrationParameters::MLcal

Mean of the light from a histogram of the calibrated standard.

Namespace: Euresys::Open_eVision

[C++]

```
float MLcal
```

EQRCodeIso29158CalibrationParameters::Rcal

Reported reflectance value, from a calibration standard.

Namespace: Euresys::Open_eVision

[C++]

```
float Rcal
```

EQRCodeIso29158CalibrationParameters::SRcal

System response parameters(such as exposure and/or again) used to create an image of the calibration standard.

Namespace: Euresys::Open_eVision

[C++]

```
float SRcal
```

EQRCodeIso29158CalibrationParameters::SRtarget

System response parameters(such as exposure and/or again) used to create an image of the symbole under test.

Namespace: Euresys::Open_eVision

[C++]

```
float SRtarget
```

5.41. EQRCodeIso29158GradingParameters Struct

Holds all grading parameters pertaining to ISO/IEC 29158

Namespace: Euresys::Open_eVision

Properties

CellContrastGrade	-
CellModulationGrade	Cell Modulation Grade

MinimumReflectanceGrade	Minimum Reflectance Grade
FixedPatternDamageGrade	FixedPatternDamageGrade
OverallSymbolGrade	-
ScanGrade	Scan Grade
MeanLight	Mean Light
IsMeanLightInRequiredBounds	Is Mean Light In Required Bounds

Methods

EQRCodeIso29158GradingParameters	-

EQRCodeIso29158GradingParameters::CellContrastGrade

-

Namespace: Euresys::Open_eVision

```
[C++]
int CellContrastGrade
```

EQRCodeIso29158GradingParameters::CellModulationGrade

Cell Modulation Grade

Namespace: Euresys::Open_eVision

```
[C++]
int CellModulationGrade
```

EQRCodeIso29158GradingParameters::EQRCodeIso29158GradingParameters

-

Namespace: Euresys::Open_eVision

```
[C++]
void EQRCodeIso29158GradingParameters(
 )
```

EQRCodeIso29158GradingParameters::FixedPatternDamageGrade

FixedPatternDamageGrade

Namespace: Euresys::Open_eVision

```
[C++]
int FixedPatternDamageGrade
```

EQRCodeIso29158GradingParameters::IsMeanLightInRequiredBounds

Is Mean Light In Required Bounds

Namespace: Euresys::Open_eVision

```
[C++]
bool IsMeanLightInRequiredBounds
```

EQRCodeIso29158GradingParameters::MeanLight

Mean Light

Namespace: Euresys::Open_eVision

```
[C++]
float MeanLight
```

EQRCodeIso29158GradingParameters::MinimumReflectanceGrade

Minimum Reflectance Grade

Namespace: Euresys::Open_eVision

```
[C++]
int MinimumReflectanceGrade
```

EQRCodeIso29158GradingParameters::OverallSymbolGrade

-

Namespace: Euresys::Open_eVision

[C++]

```
int OverallSymbolGrade
```

EQRCodeIso29158GradingParameters::ScanGrade

This struct member is deprecated.

Scan Grade

Namespace: Euresys::Open_eVision

[C++]

```
int ScanGrade
```

5.42. ERenderStyle Struct

Represents how to visualize 3D Objects in a E3DViewer

Namespace: Euresys::Open_eVision::Easy3D

Properties

hasFill	Indicates that the inside of the object should be rendered.
fillRGB	Fill (inside) color of the object
hasLine	Indicates that the edges of the object should be rendered
lineRGB	Line (edge) color of the object.
pointRGB	Color of the object if it is a point.

Methods

ERenderStyle	Constructs a default ERenderStyle object.

ERenderStyle::ERenderStyle

Constructs a default ERenderStyle object.

Namespace: Euresys::Open_eVision::Easy3D

[C++]

```
void ERenderStyle(
 )
```

ERenderStyle::fillRGB

Fill (inside) color of the object

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EC24A fillRGB
```

Remarks

Only applied if the object is a polygon.

ERenderStyle::hasFill

Indicates that the inside of the object should be rendered.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool hasFill
```

Remarks

Only applied if the object is a polygon.

ERenderStyle::hasLine

Indicates that the edges of the object should be rendered

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
bool hasLine
```

Remarks

Only applied if the object is a polygon.

ERenderStyle::lineRGB

Line (edge) color of the object.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EC24A lineRGB
```

Remarks

Only applied if the object is a polygon.

ERenderStyle::pointRGB

Color of the object if it is a point.

Namespace: Euresys::Open_eVision::Easy3D

```
[C++]
EC24A pointRGB
```

5.43. ERGB Struct

NTSC/PAL/SMPTE Red, Green, Blue color system.

Namespace: Euresys::Open_eVision

Properties

R	Red component.
G	Green component.
B	Blue component.

ERGB::B

Blue component.

Namespace: Euresys::Open_eVision

```
[C++]
float B
```

ERGB::G

Green component.

Namespace: Euresys::Open_eVision

```
[C++]
float G
```

ERGB::R

Red component.

Namespace: Euresys::Open_eVision

```
[C++]
float R
```

5.44. ERGBColor Struct

NTSC/PAL/SMPTE Red, Green, Blue color system.

Namespace: Euresys::Open_eVision

Properties

Red	Red component.
Green	Green component.
Blue	Blue component.

Methods

ERGBColor	Constructs an ERGBColor object.

ERGBColor::Blue

Blue component.

Namespace: Euresys::Open_eVision

```
[C++]
int Blue
```

ERGBColor::ERGBColor

Constructs an ERGBColor object.

Namespace: Euresys::Open_eVision

[C++]

```
void ERGBColor(
 int red,
 int green,
 int blue
 )
void ERGBColor(
 int hexColor
 )
void ERGBColor(
 )
```

Parameters

red
Red component.

green
Green component.

blue
Blue component.

hexColor
Components defined as a hexadecimal color code (0xRRGGBB)

ERGBColor::Green

Green component.

Namespace: Euresys::Open_eVision

[C++]

```
int Green
```

ERGBColor::Red

Red component.

Namespace: Euresys::Open_eVision

[C++]

```
int Red
```

5.45. EROCPoint Struct

The structure representing a point on the ROC (Receiver Operating Characteristic) curve.

Namespace: Euresys::Open_eVision::EasyDeepLearning

Properties

TPR	The True Positive Rate: it is the number of defective images classified as defective, normalized by the number of defective images. The TPR can be NaN (Not A Number) if the metrics were computed using only good images.
TP	The True Positive count: it is the number of defective images classified as defective.
P	The Positive count: it is the number of defective images.
FPR	The False Positive Rate: it is the number of good image classified as defective, normalized by the number of good images.
FP	The False Positive count: it is the number of good image classified as defective.
N	The Negative count: it is the number of good images.
Threshold	The threshold associated with the true and false positive rates (see EUnsupervisedSegmenter::ClassificationThreshold).

Methods

EROCPoint	Constructs a EROCPoint.
Load	Loads the ROC point. The given ESerializer must have been created for reading.
Save	Saves the ROC point. The given ESerializer must have been created for writing.

EROCPoint::EROCPoint

Constructs a EROCPoint.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void EROCPoint(
 )

void EROCPoint(
 float threshold,
 int truePositiveCount,
 int positiveCount,
 int falsePositiveCount,
 int negativeCount
 )
```

Parameters

threshold
Threshold corresponding to the ROC point.

truePositiveCount
Number of defective images classified as defective.

positiveCount
Total number of defective images.

falsePositiveCount
Number of good images classified as defective.

negativeCount
Total number of good images.

EROCPoint::FP

The False Positive count: it is the number of good image classified as defective.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int FP
```

EROCPoint::FPR

The False Positive Rate: it is the number of good image classified as defective, normalized by the number of good images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
float FPR
```

EROCPoint::Load

Loads the ROC point. The given ESerializer must have been created for reading.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Load(
 const std::string& path
 )

void Load(
 ESerializer* serializer
 )
```

Parameters

path
The file path.
serializer
The serializer.

EROCPoint::N

The Negative count: it is the number of good images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int N
```

EROCPoint::P

The Positive count: it is the number of defective images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
int P
```

EROCPoint::Save

Saves the ROC point. The given ESerializer must have been created for writing.

Namespace: Euresys::Open_eVision::EasyDeepLearning

[C++]

```
void Save(
 const std::string& path
 )

void Save(
 ESerializer* serializer
 )
```

Parameters

path
The file path.
serializer
The serializer.

EROCPoint::Threshold

The threshold associated with the true and false positive rates (see EUnsupervisedSegmenter::ClassificationThreshold).

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float Threshold
```

EROCPoint::TP

The True Positive count: it is the number of defective images classified as defective.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
int TP
```

EROCPoint::TPR

The True Positive Rate: it is the number of defective images classified as defective, normalized by the number of defective images.
The TPR can be NaN (Not A Number) if the metrics were computed using only good images.

Namespace: Euresys::Open_eVision::EasyDeepLearning

```
[C++]
float TPR
```

5.46. ERun Struct

-

Namespace: Euresys::Open_eVision

Properties

OrgX	X origin of the Run
Length	Length of the Run
Y	Y position of the Run

Methods

ERun	Constructs an ERun context.
operator!=	Checks if this instance is not strictly equal to another
operator==	Checks if this instance is strictly equal to another

ERun::ERun

Constructs an ERun context.

Namespace: Euresys::Open_eVision

```
[C++]
void ERun(
 )
void ERun(
 int orgX,
 int length,
 int y
 )
```

Parameters

orgX
X origin of the Run.

length
Length of the Run.

y
Y position of the Run.

ERun::Length

Length of the Run

Namespace: Euresys::Open_eVision

```
[C++]
int Length
```

ERun::operator!=

Checks if this instance is not strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator!=(
 const ERun& other
 )
```

Parameters

other

Reference to the other ERun instance

ERun::operator==

Checks if this instance is strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ERun& other
 )
```

Parameters

other

Reference to the other ERun instance

ERun::OrgX

X origin of the Run

Namespace: Euresys::Open_eVision

```
[C++]
int OrgX
```

ERun::Y

Y position of the Run

Namespace: Euresys::Open_eVision

```
[C++]
int Y
```

5.47. ERunData Struct

This struct is deprecated.

Describes runs.

Remarks

A run is characterized by a starting point (OrgX, OrgY), by a length, a class, a unique identification number and the number of the object to which they belong. After the run construction phase, all the runs are gathered in a single dynamic list.

Namespace: Euresys::Open_eVision

Properties

OrgX	Start point abscissa.
OrgY	Start point ordinate.
Len	Length.
Class	Class.
ObjNum	Identification number of the object to which the run belongs.

ERunData::Class

This struct member is deprecated.

Class.

Namespace: Euresys::Open_eVision

```
[C++]
int Class
```

ERunData::Len

This struct member is deprecated.

Length.

Namespace: Euresys::Open_eVision

```
[C++]
int Len
```

ERunData::ObjNum

This struct member is deprecated.

Identification number of the object to which the run belongs.

Namespace: Euresys::Open_eVision

```
[C++]
int ObjNum
```

ERunData::OrgX

This struct member is deprecated.

Start point abscissa.

Namespace: Euresys::Open_eVision

```
[C++]
int OrgX
```

ERunData::OrgY

This struct member is deprecated.

Start point ordinate.

Namespace: Euresys::Open_eVision

```
[C++]
int OrgY
```

5.48. ESize Struct

-

Namespace: Euresys::Open_eVision

Properties

Width	Width.
Height	Height.

Methods

ESize	Constructs a ESize.
operator!=	Checks if this instance is not strictly equal to another
operator==	Checks if this instance is strictly equal to another

ESize::ESize

Constructs a ESize.

Namespace: Euresys::Open_eVision

[C++]

```
void ESize(
 )

void ESize(
 float width,
 float height
 )
```

Parameters

width
Width.

height
Height.

ESize::Height

Height.

Namespace: Euresys::Open_eVision

[C++]

```
float Height
```

ESize::operator!=

Checks if this instance is not strictly equal to another

Namespace: Euresys::Open_eVision

[C++]

```
bool operator!=(
 const ESize& other
 )
```

Parameters

other
Reference to the other ESize instance

ESize::operator==

Checks if this instance is strictly equal to another

Namespace: Euresys::Open_eVision

```
[C++]
bool operator==(
 const ESize& other
 )
```

Parameters

other

Reference to the other ESize instance

ESize::Width

Width.

Namespace: Euresys::Open_eVision

```
[C++]
float Width
```

5.49. EVSH Struct

Value, Saturation, Hue color system.

Namespace: Euresys::Open_eVision

Properties

V	Value component.
S	Saturation component.
H	Hue component.

EVSH::H

Hue component.

Namespace: Euresys::Open_eVision

[C++]

```
float H
```

EVSH::S

Saturation component.

Namespace: Euresys::Open_eVision

[C++]

```
float S
```

EVSH::V

Value component.

Namespace: Euresys::Open_eVision

[C++]

```
float V
```

5.50. EXYZ Struct

CIE XYZ color system.

Namespace: Euresys::Open_eVision

Properties

X	X component.
Y	Y component.
Z	Z component.

EXYZ::X

X component.

Namespace: Euresys::Open_eVision

[C++]

```
float X
```

EXYZ::Y

Y component.

Namespace: Euresys::Open_eVision

```
[C++]
float Y
```

EXYZ::Z

Z component.

Namespace: Euresys::Open_eVision

```
[C++]
float Z
```

5.51. EYIQ Struct

CCIR Luma, Inphase, Quadrature color system.

Namespace: Euresys::Open_eVision

Properties

Y	Luma component.
I	Inphase component.
Q	Quadrature component.

EYIQ::I

Inphase component.

Namespace: Euresys::Open_eVision

```
[C++]
float I
```

EYIQ::Q

Quadrature component.

Namespace: Euresys::Open_eVision

```
[C++]
float Q
```

EYIQ::Y

Luma component.

Namespace: Euresys::Open_eVision

```
[C++]
float Y
```

5.52. EYSH Struct

CCIR Luma, Saturation, Hue color system.

Namespace: Euresys::Open_eVision

Properties

Y	Luma component.
S	Saturation component.
H	Hue component.

EYSH::H

Hue component.

Namespace: Euresys::Open_eVision

```
[C++]
float H
```

EYSH::S

Saturation component.

Namespace: Euresys::Open_eVision

[C++]

```
float S
```

EYSH::Y

Luma component.

Namespace: Euresys::Open_eVision

[C++]

```
float Y
```

5.53. EYUV Struct

CCIR Luma, U Chroma, V Chroma color system.

Namespace: Euresys::Open_eVision

Properties

Y	Luma component.
U	U Chroma component.
V	V Chroma component.

EYUV::U

U Chroma component.

Namespace: Euresys::Open_eVision

[C++]

```
float U
```

EYUV::V

V Chroma component.

Namespace: Euresys::Open_eVision

[C++]

```
float V
```

EYUV::Y

Luma component.

Namespace: Euresys::Open_eVision

```
[C++]
float Y
```

6. Enumerations

6.1. E3DAttribute Enum

This enumeration contains the possible attributes of the EPointCloud in addition to the E3DPoint. The attributes have constraints about the type that is used to represent them. See also E3DAttribute.

Namespace: Euresys::Open_eVision::Easy3D

E3DAttribute_Color	Represents the color of an E3DPoint, it should use the type EC24A.
E3DAttribute_Normal	Represents the normal of an E3DPoint, it should use the type E3DPoint.
E3DAttribute_Intensity	Represents the intensity of an E3DPoint, it should use a numeric type.
E3DAttribute_Texture	Represents the texture of an E3DPoint, it can use any type.
E3DAttribute_Index	Represents an index related to an E3DPoint, it should use the type uint32_t or int32_t.
E3DAttribute_ Confidence	Represents the confidence of an E3DPoint, it should use the type float.
E3DAttribute_Distance	Represents the distance of an E3DPoint to another element, it should use the type float. This attribute can be set by E3DAligner, E3DMatcher, E3DComparer.
E3DAttribute_Curvature	Represents the local curvature of an E3DPoint, it should use the type float. This attribute can be set by EPointCloud::ComputeNormalsAndCurvatures.

Remarks

Numeric types are: uint8_t, uint16_t, uint32_t, int32_t, float, double.

6.2. E3DObjectFeature Enum

The list of possible extracted features for E3DObject

Namespace: Euresys::Open_eVision::Easy3D

E3DObjectFeature_ Length	Length of the object in metric units.
E3DObjectFeature_ Width	Width of the object in metric units.
E3DObjectFeature_ LocalHeight	Local height of the object in metric units.
E3DObjectFeature_ ReferenceHeight	Reference height of the object in metric units.
E3DObjectFeature_ Orientation	Orientation of the object.
E3DObjectFeature_ BoundingBox	3D oriented bounding box of the object.
E3DObjectFeature_ AveragePosition	3D average position of the object.
E3DObjectFeature_ LocalTopPosition	3D highest position of the object relatively to the object base plane.
E3DObjectFeature_ ReferenceTopPosition	3D top position relative to the ZMap origin (this is the position with the highest Z coordinate).

E3DObjectFeature_Plane	Plane fitted to the object 3D positions.
E3DObjectFeature_LocalTilt	Angle between the object plane and the base plane.
E3DObjectFeature_ReferenceTilt	Angle between the object plane and the vertical (Z) axis.
E3DObjectFeature_Area	Object area in metric units.
E3DObjectFeature_Volume	Object volume in metric units.
E3DObjectFeature_BasePlane	Base plane for the object.
E3DObjectFeature_BaseTilt	Angle between the base plane and the vertical (Z) axis.
E3DObjectFeature_ERectangleRegion	ERectangleRegion enclosing the object ZMap pixels.
E3DObjectFeature_ERegion	ERegion composed of the object ZMap pixels.
E3DObjectFeature_Sphere	Sphere fitted on the object

6.3. EAdaptiveThresholdMethod Enum

Adaptive thresholding modes.

Namespace: Euresys::Open_eVision

EAdaptiveThresholdMethod_Mean	Use the mean as threshold.
EAdaptiveThresholdMethod_Median	Use the median as threshold.
EAdaptiveThresholdMethod_Middle	Use the middle of the values interval as threshold.

6.4. EAlignmentPolarity Enum

Polarity of an alignment, used in EFeaturesAligner.

Namespace: Euresys::Open_eVision::Easy3D

EAlignmentPolarity_ModelToMeasured	The transformation from the model data to the measured data.

EAlignmentPolarity_MeasuredToModel	The transformation from the measured data to the model data.

6.5. EAngleUnit Enum

The angle units that are supported by Open eVision.

Namespace: Euresys::Open_eVision

EAngleUnit_Revolutions	Revolutions (0..1 corresponds to a full revolution).
EAngleUnit_Radians	Radians (0..2Pi corresponds to a full revolution).
EAngleUnit_Degrees	Degrees (0..360 corresponds to a full revolution).
EAngleUnit_Grades	Grades (0..400 corresponds to a full revolution).

6.6. EArithmeticLogicOperation Enum

Supported arithmetic or logic pixel-wise operators.

Namespace: Euresys::Open_eVision

EArithmeticLogicOperation_Copy	Sheer copy.
EArithmeticLogicOperation_Invert	Complement. (*)
EArithmeticLogicOperation_Add	Saturated addition. (*)
EArithmeticLogicOperation_Subtract	Saturated subtraction. (*)
EArithmeticLogicOperation_Multiply	Saturated multiplication. (*)
EArithmeticLogicOperation_Divide	Saturated division. (*)
EArithmeticLogicOperation_Modulo	Modulo. (*)
EArithmeticLogicOperation_ShiftLeft	Arithmetic left shift (multiplication by a power of 2). (*)
EArithmeticLogicOperation_ShiftRight	Arithmetic right shift (division by a power of 2). (*)
EArithmeticLogicOperation_ScaledAdd	Non saturating addition ((left + right) / 2). (*)

EArithmeticLogicOperation_ScaledSubtract	Non saturating subtraction ((left + complemented right) / 2). (*)
EArithmeticLogicOperation_ScaledMultiply	Non saturating multiplication (left * right / 256 in the BW8 case, and left * right / 65,536 in the BW16 one). (*)
EArithmeticLogicOperation_ScaledDivide	Non saturating division (256 * left / right in the BW8 case, and 65,536 * left / right in the BW16 one). (*)
EArithmeticLogicOperation_BitwiseAnd	Bitwise AND.
EArithmeticLogicOperation_BitwiseOr	Bitwise OR.
EArithmeticLogicOperation_BitwiseXor	Bitwise exclusive OR.
EArithmeticLogicOperation_LogicalAnd	Logical AND (non zero is true). (*)
EArithmeticLogicOperation_LogicalOr	Logical OR (non zero is true). (*)
EArithmeticLogicOperation_LogicalXor	Logical exclusive OR (non zero is true). (*)
EArithmeticLogicOperation_Min	Minimum. (*)
EArithmeticLogicOperation_Max	Maximum. (*)
EArithmeticLogicOperation_SetZero	Copy the right operand where the left operand is zero.
EArithmeticLogicOperation_SetNonZero	Copy the right operand where the left operand is non zero.
EArithmeticLogicOperation_Equal	Equality comparison. (*)
EArithmeticLogicOperation_NotEqual	Non equality comparison. (*)
EArithmeticLogicOperation_GreaterOrEqual	"Greater or equal" comparison. (*)
EArithmeticLogicOperation_LesserOrEqual	"Lesser or equal" comparison.
EArithmeticLogicOperation_Greater	"Greater" comparison. (*)
EArithmeticLogicOperation_Lesser	"Lesser" comparison. (*)
EArithmeticLogicOperation_Compare	Absolute value of the difference. (*)
EArithmeticLogicOperation_Overlay	Overlay of one image onto a source image giving a destination image. (See note at the end of the topic). (*) (**)

EArithmeticLogicOperation_BitwiseNot	Same as EArithmeticLogicOperation_Invert.
EArithmeticLogicOperation_Average	Same as EArithmeticLogicOperation_ScaledAdd. (*)

Remarks

(*) Not applicable for the BW1 images/ROIs. (**) In the overlay image, black pixels (0 valued) are considered as transparent. If a C24 image is used as overlay, all pixels (but the black ones) will be copied to the destination image. If a BW8 image is used as overlay, all non-black pixels will be converted to the color defined by the OverlayColor parameter before copy to the destination image. The destination image is always a C24 image. If no source image is given (only overlay and destination), the destination image is considered as the source image.

6.7. EasyOCR2CharacterFilter Enum

This enumeration contains the possible filters for loading fonts in EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2CharacterFilter_ASCII	All ASCII characters are loaded.
EasyOCR2CharacterFilter_Letters	Only (alphabetic) letters are loaded (both lowercases and uppercases).
EasyOCR2CharacterFilter_Digits	Only digits are loaded.
EasyOCR2CharacterFilter_LowerCaseLetters	Only (alphabetic) lowercase letters are loaded.
EasyOCR2CharacterFilter_UpperCaseLetters	Only (alphabetic) uppercase letters are loaded.

6.8. EasyOCR2CharSpacingBias Enum

This enumeration contains the possible biases for the optimised character spacing in the detection phase of EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2CharSpacingBias_Wide	The optimisation is biased toward wide character spacing.
EasyOCR2CharSpacingBias_Neutral	The optimisation is not biased.
EasyOCR2CharSpacingBias_Narrow	The optimisation is biased toward narrow character spacing.

6.9. EasyOCR2CharWidthBias Enum

This enumeration contains the possible biases for the optimised character width in the detection phase of EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2CharWidthBias_Widest	The optimisation is biased toward very wide boxes.
EasyOCR2CharWidthBias_Wide	The optimisation is biased toward wide boxes.
EasyOCR2CharWidthBias_Neutral	The optimisation is not biased.
EasyOCR2CharWidthBias_Narrow	The optimisation is biased toward narrow boxes.
EasyOCR2CharWidthBias_Narrowest	The optimisation is biased toward very narrow boxes.

6.10. EasyOCR2DrawDetectionStyle Enum

This enumeration contains the possible drawing styles for the detection results in EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2DrawDetectionStyle_DrawChars	A bounding box is drawn around each individual detected character
EasyOCR2DrawDetectionStyle_DrawWords	A bounding box is drawn around each detacted word, containing all characters in the word
EasyOCR2DrawDetectionStyle_DrawLines	A bounding box is drawn around each detacted line, containing all characters/words in the line
EasyOCR2DrawDetectionStyle_DrawText	A bounding box is drawn around the detected text, containing all characters/words/lines in the text

6.11. EasyOCR2DrawRecognitionStyle Enum

This enumeration contains the possible drawing styles for the recognition results in EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2DrawRecognitionStyle_LeftTop	The recognition result is drawn at the left top of the character

EasyOCR2DrawRecognitionStyle_LeftMiddle	The recognition result is drawn at the left middle of the character
EasyOCR2DrawRecognitionStyle_LeftBottom	The recognition result is drawn at the left bottom of the character
EasyOCR2DrawRecognitionStyle_BottomLeft	The recognition result is drawn at the bottom left of the character
EasyOCR2DrawRecognitionStyle_BottomMiddle	The recognition result is drawn at the bottom middle of the character
EasyOCR2DrawRecognitionStyle_BottomRight	The recognition result is drawn at the bottom right of the character
EasyOCR2DrawRecognitionStyle_RightBottom	The recognition result is drawn at the right bottom of the character
EasyOCR2DrawRecognitionStyle_RightMiddle	The recognition result is drawn at the right middle of the character
EasyOCR2DrawRecognitionStyle_RightTop	The recognition result is drawn at the right top of the character
EasyOCR2DrawRecognitionStyle_TopRight	The recognition result is drawn at the top right of the character
EasyOCR2DrawRecognitionStyle_TopMiddle	The recognition result is drawn at the top middle of the character
EasyOCR2DrawRecognitionStyle_TopLeft	The recognition result is drawn at the top left of the character

6.12. EasyOCR2DrawSegmentationStyle Enum

This enumeration contains the possible drawing styles for the segmentation results in EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2DrawSegmentationStyle_DrawBlobs	The segmented blobs are drawn directly.

6.13. EasyOCR2TextPolarity Enum

This enumeration contains the possible polarities of text searched during segmentation in EOCR2.

Namespace: Euresys::Open_eVision

EasyOCR2TextPolarity_WhiteOnBlack	The text is light on a dark background

EasyOCR2TextPolarity_BlackOnWhite	The text is dark on a light background

6.14. EAttributeType Enum

This enumeration contains the possible types for the E3DAttribute. See also EPointCloud::GetAttributeBufferType.

Namespace: Euresys::Open_eVision::Easy3D

EAttributeType_UINT8	Corresponds to an uint8_t.
EAttributeType_UINT16	Corresponds to an uint16_t.
EAttributeType_UINT32	Corresponds to an uint32_t.
EAttributeType_INT32	Corresponds to an int32_t.
EAttributeType_FLOAT	Corresponds to a float.
EAttributeType_DOUBLE	Corresponds to a double.
EAttributeType_EC24A	Corresponds to an EC24A.
EAttributeType_E3DPOINT	Corresponds to an E3DPoint.
EAttributeType_UNDEFINED	Corresponds to a non-defined type.

6.15. EAxisOriginMode Enum

This enumeration contains the possible values for the parameter of E3DViewer::AxisOrigin method.

Namespace: Euresys::Open_eVision::Easy3D

EAxisOriginMode_BoundingBoxOrigin	Use the bounding box lower point as axis origin
EAxisOriginMode_WorldOrigin	Use the world origin as axis origin
EAxisOriginMode_UserDefined	Use a user defined point as axis origin

6.16. EAxisSystemType Enum

-

Namespace: Euresys::Open_eVision::Easy3D

EAxisSystemType_CornerUpperLeft	-
EAxisSystemType_CornerLowerLeft	-
EAxisSystemType_Unknown	-

6.17. EBarCodeSymbologies Enum

-

Namespace: Euresys::Open_eVision::EasyBarCode2

EBarCodeSymbologies_AdsAnker	ADS Anker symbology.
EBarCodeSymbologies_Bc412	Code BC 412 symbology.
EBarCodeSymbologies_Codabar	Codabar symbology.
EBarCodeSymbologies_Code11	Code 11 symbology.
EBarCodeSymbologies_Code13	Code 13 symbology.
EBarCodeSymbologies_Code25Compressed	Code 25 Compressed symbology.
EBarCodeSymbologies_Code25Datalogic	Code 25 DataLogic symbology.
EBarCodeSymbologies_Code25Interleaved	Code 25 Interleaved symbology.
EBarCodeSymbologies_Code25Iata	Code 25 IATA symbology.
EBarCodeSymbologies_Code25Industry	Code 25 Industry symbology.
EBarCodeSymbologies_Code25Inverted	Code 25 Inverted symbology.
EBarCodeSymbologies_Code25Matrix	Code 25 Matrix symbology.

EBarCodeSymbologies_Code39	Code 39 symbology.
EBarCodeSymbologies_Code39Extended	Code 39 Extended symbology.
EBarCodeSymbologies_Code39Reduced	Code 39 Reduced symbology.
EBarCodeSymbologies_Code32	Code 32 symbology.
EBarCodeSymbologies_Code93	Code 93 symbology.
EBarCodeSymbologies_Code93Extended	Code 93 Extended symbology.
EBarCodeSymbologies_Code128	Code 128 symbology.
EBarCodeSymbologies_CodeBcdMatrix	Code BCD Matrix symbology.
EBarCodeSymbologies_CodeCip	Code CIP symbology.
EBarCodeSymbologies_Ean8	EAN 8 symbology.
EBarCodeSymbologies_Ean13	EAN 13 symbology.
EBarCodeSymbologies_Gs1_128	GS1 128 symbology.
EBarCodeSymbologies_IbmDeltaDistanceA	IBM Delta Distance A symbology.
EBarCodeSymbologies_Msi	MSI symbology.
EBarCodeSymbologies_Plessey	Plessey symbology.
EBarCodeSymbologies_Gs1DataBarOmnidirectional	GS1 DataBar Omnidirectional symbology. As we do not analyze the height of the detected barcodes at the moment, GS1 DataBar Truncated codes will be detected as omnidirectional ones.
EBarCodeSymbologies_Gs1DataBarLimited	GS1 DataBar Limited symbology.
EBarCodeSymbologies_Gs1DataBarExpanded	GS1 DataBar Expanded symbology.
EBarCodeSymbologies_Rss14	RSS-14 symbology. Present for backward-compatibility, replaced by EBarCodeSymbologies_Gs1DataBarOmnidirectional.
EBarCodeSymbologies_Rss14Limited	RSS-14 Limited symbology. Present for backward-compatibility, replaced by EBarCodeSymbologies_Gs1DataBarLimited.

EBarCodeSymbologies_Rss14Expanded	RSS-14 Expanded symbology. Present for backward-compatibility, replaced by EBarCodeSymbologies_Gs1DataBarExpanded.
EBarCodeSymbologies_Telepen	Telepen symbology.
EBarCodeSymbologies_UpcA	UPC-A symbology.
EBarCodeSymbologies_UpcE	UPC-E symbology.
EBarCodeSymbologies_CodeStk	Code STK symbology.
EBarCodeSymbologies_PharmacodeOneTrack	Pharmacode with one track symbology.
EBarCodeSymbologies_BinaryCode	Binary Code symbology.

6.18. EBayerConfiguration Enum

The Bayer image color configuration of the first two pixels.

Namespace: Euresys::Open_eVision

EBayerConfiguration_RG	The Bayer image starts with Red-Green pixels
EBayerConfiguration_GR	The Bayer image starts with Green-Red pixels
EBayerConfiguration_BG	The Bayer image starts with Blue-Green pixels
EBayerConfiguration_GB	The Bayer image starts with Green-Blue pixels

6.19. EByteInterpretationMode Enum

This enumeration contains the available modes to interpret bytes values of a decoded string.

Namespace: Euresys::Open_eVision

EByteInterpretationMode_Hexadecimal	Bytes will be converted to hexadecimal values surrounded by an escape character: "0xEFBFBD.
EByteInterpretationMode_UTF8	Bytes will be converted to UTF-8.

EByteInterpretationMode_Auto	Bytes are converted to UTF-8 or their supported ECI table if possible. They will be converted to hexadecimal with escape character otherwise.

Remarks

EByteInterpretationMode_Hexadecimal: each byte is encoded with its corresponding two characters hexadecimal value. This mode does not throw exceptions. This mode overrides the byte encoding dictated by the ECI supported tables.
If the conversion required by the selected mode is not feasible, an exception is thrown.
The EByteInterpretationMode_Auto will not generate exceptions if the ECI table is not supported or if the conversion is not feasible in UTF-8.

6.20. ECalibrationMode Enum

Allowed values for the calibration mode.

Namespace: Euresys::Open_eVision

ECalibrationMode_Raw	No calibration at all.
ECalibrationMode_Inverse	The ordinate axis points downwards instead of upwards.
ECalibrationMode_Scaled	The pixels are assigned a physical size.
ECalibrationMode_Anisotropic	The physical size of the pixels differ horizontally and vertically. (Beware there is a restriction pertaining to the allowed image anisotropy. The resulting pixels aspect ratio should be in the range [-4/3, -3/4] (or [3/4, 4/3]), otherwise the calibration process could fail).
ECalibrationMode_Skewed	The coordinate axis make an angle with the image edge.
ECalibrationMode_Tilted	The field-of-view plane is not perpendicular to the optical axis.
ECalibrationMode_Radial	The lens introduces some amount of radial distortion.
ECalibrationMode_Bilinear	This mode can not be combined with other calibration mode. The bilinear calibration is based on a first order polynomial approach.
ECalibrationMode_Quadratic	This mode can not be combined with other calibration mode. The quadratic calibration is based on a second order polynomial approach.

6.21. ECalibrationType Enum

The Easy3D calibration type.

Namespace: Euresys::Open_eVision::Easy3D

ECalibrationType_Scale	Calibration type is EScaleCalibrationModel.
ECalibrationType_ ExplicitGeometric	Calibration type is EExplicitGeometricCalibrationModel.
ECalibrationType_ ObjectBased	Calibration type is EObjectBasedCalibrationModel.
ECalibrationType_ Unknown	Calibration type is not defined.

6.22. ECannyThresholdingMode Enum

The thresholding modes for the Canny edge detector.

Namespace: Euresys::Open_eVision

ECannyThresholdingMode_Relative	Relative thresholding mode.
ECannyThresholdingMode_Absolute	Absolute thresholding mode.

6.23. ECC000Family Enum

-

Namespace: Euresys::Open_eVision::EasyMatrixCode2

ECC000Family_ECC000	-
ECC000Family_ECC050	-
ECC000Family_ECC080	-
ECC000Family_ECC100	-
ECC000Family_ECC140	-
ECC000Family_ Unknown	-

6.24. ECellColor Enum

Allowed values for a cell color (Black or White).

Namespace: Euresys::Open_eVision

ECellColor_Black	Black, the cell is dark with respect to the background.
ECellColor_White	White, the cell is light with respect to the background.

6.25. EClassifierCapacity Enum

The capacity of the classifier network.
A larger capacity means that the underlying neural network is capable of learning more information but it will be slower. Also a lower capacity allow for a smaller minimal height and width of input.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EClassifierCapacity_Small	Small capacity.
EClassifierCapacity_Normal	Normal capacity.
EClassifierCapacity_Large	Large capacity.
EClassifierCapacity_Onnx	Onnx classifier capacity.

6.26. EClippingMode Enum

Allows to choose how the fitted segment length and centre are computed

Namespace: Euresys::Open_eVision

EClippingMode_CenteredNominal	Regular mode: the fitted segment always has the nominal length of the line gauge.
EClippingMode_ClippedToValidSamples	The fitted segment does not extend beyond valid samples. It is clipped to the projection of the valid samples on the fitted line.
EClippingMode_ClippedInNominalShape	The segment is built by clipping the fitted line in the rectangular range where the ELineGauge looks for valid transition samples, i.e. the rectangle which is centered and aligned on the ELineGauge nominal line, and which height is two times the ELineGauge::Tolerance.

6.27. ECodeType Enum

-

Namespace: Euresys::Open_eVision

ECodeType_MatrixCode	-
ECodeType_QRCode	-
ECodeType_BarCode	-
ECodeType_Unknown	-

6.28. EColorQuantization Enum

Allowed values for the quantization mode in EasyColor.

Namespace: Euresys::Open_eVision

EColorQuantization_FullRange	Values are quantized in range 0..255.
EColorQuantization_Ccir601	Values are quantized in range 16..235 for the R, G, B or Y component, and in range 16..240 for the I, Q, U and y components.

Remarks

When quantizing the color values for the RGB or YIQ/YUV representation, one usually uses the full 0..255 range. Anyway, the CCIR has defined an alternate convention such that some values in this interval are reserved. Before performing a conversion, functions EasyColor::SrcQuantization and EasyColor::DstQuantization can be used to specify the rule used.

6.29. EColorRampMode Enum

This enumeration contains the possible values for the parameter of E3DViewer::GenerateColors method.

Namespace: Euresys::Open_eVision::Easy3D

EColorRampMode_HueFromZ	The Hue color component calculated from Z coordinate. The colors range from Blue (minimum Z) to Red (maximum Z).
EColorRampMode_HueFromY	The Hue color component calculated from Y coordinate. The colors range from Blue (minimum Y) to Red (maximum Y).
EColorRampMode_HueFromX	The Hue color component calculated from X coordinate. The colors range from Blue (minimum X) to Red (maximum X).
EColorRampMode_HueFastFromZ	The Hue color component with fast change calculated from Z coordinate. The colors range from Red (minimum Z) to Red (maximum Z) going through green and blue.
EColorRampMode_HueFastFromY	The Hue color component with fast change calculated from Y coordinate. The colors range from Red (minimum Y) to Red (maximum Y) going through green and blue.

EColorRampMode_ HueFastFromX	The Hue color component with fast change calculated from X coordinate. The colors range from Red (minimum X) to Red (maximum X) going through green and blue.
EColorRampMode_ RGBCube	RGB colors directly calculated from XYZ coordinates.
EColorRampMode_ HueFromIntensity	The colors used are calculated from the intensity attribute of EPointCloud.
EColorRampMode_ HueFromConfidence	The colors used are calculated from the confidence attribute of EPointCloud.
EColorRampMode_ HueFromDistance	The colors used are calculated from the distance attribute of EPointCloud.
EColorRampMode_ HueFromNormal	The colors used are calculated from the normal attribute of EPointCloud. A normal of (1, 0, 0) is converted to (255, 0, 0), one of (0, 1, 0) to (0, 255, 0) and one of (0, 0, 1) to (0, 0, 255).
EColorRampMode_ Undefined	The color ramp is not defined. This is the value used when there is no color and all E3DPoint will be displayed in white.

6.30. EColorSystem Enum

The color systems that are supported by Open eVision.

Namespace: Euresys::Open_eVision

EColorSystem_NoColor	Undefined
EColorSystem_Bilevel	Binary black & white.
EColorSystem_ GrayLevel	Continuous tone black & white.
EColorSystem_Xyz	CIE XYZ.
EColorSystem_Rgb	NTSC/PAL/SMPTE Red, Green, Blue.
EColorSystem_Lab	CIE Lightness, a*, b*.
EColorSystem_Luv	CIE Lightness, u*, v*.
EColorSystem_Yuv	CCIR Luma, U Chroma, V Chroma.
EColorSystem_Yiq	CCIR Luma, Inphase, Quadrature.
EColorSystem_Lch	Lightness, Chroma, Hue.
EColorSystem_Ish	Intensity, Saturation, Hue
EColorSystem_Lsh	Lightness, Saturation, Hue.
EColorSystem_Vsh	Value, Saturation, Hue.
EColorSystem_Ysh	CCIR Luma, Saturation, Hue.

Remarks

Open eVision supports several color systems. The achromatic ones are related to black and white and gray-level images (EImageBW1 and EImageBW8). The remaining ones apply to color images (EImageC24). Also see unquantized and quantized colors for the allowed ranges of values.

6.31. EComparisonDistanceMode Enum

This enumeration specifies the distance mode for the 3D comparison. See E3DMatcher::ComparisonDistanceMode and E3DComparer::ComparisonDistanceMode.

Namespace: Euresys::Open_eVision::Easy3D

EComparisonDistanceMode_Fast	Same as EComparisonDistanceMode_Euclidean_Fast, kept for backward compatibility for E3DMatcher.
EComparisonDistanceMode_Normal	Same as EComparisonDistanceMode_Euclidean, kept for backward compatibility for E3DMatcher.
EComparisonDistanceMode_Advanced	Same as EComparisonDistanceMode_Euclidean_Advanced, kept for backward compatibility for E3DMatcher.
EComparisonDistanceMode_Euclidean_Fast	An algorithm faster but less accurate than EComparisonDistanceMode_Euclidean (especially on edges). Not compatible with E3DComparer.
EComparisonDistanceMode_Euclidean	The default algorithm.
EComparisonDistanceMode_Euclidean_Advanced	A more advanced algorithm used to penalize bumps but with increased computation time. Not compatible with E3DComparer.
EComparisonDistanceMode_Normals	Works like EComparisonDistanceMode_Euclidean with the exception that the distances are not computed from points but from their normals. As a consequence, the distance threshold set by E3DMatcher::SetAnomalyThresholds or E3DComparer::SetAnomalyThresholds must be expressed in Easy::AngleUnit.
EComparisonDistanceMode_Normals_Advanced	A more advanced algorithm based on normals distances. This is more robust towards false positives near edges than EComparisonDistanceMode_Normals. This setting ignores the values set by E3DMatcher::AllComparisonNoExtraMaterial, E3DComparer::NoExtraMaterial and E3DMatcher::EnableMissingPointAsAnomaly (missing points are not detected). When using this option, the distance threshold set by E3DMatcher::SetAnomalyThresholds or E3DComparer::SetAnomalyThresholds must be expressed in Easy::AngleUnit.

6.32. EConfusionMatrixElement Enum

The various elements representing the cases of the 2x2 confusion matrix used in EConfusionMatrixElement.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EConfusionMatrixElement_CorrectlyClassifiedGoodSample	A CorrectlyClassifiedGoodSample is a good images classified as good. It is also called a true negative.
EConfusionMatrixElement_BadlyClassifiedGoodSample	A BadlyClassifiedGoodSample is a good images classified as defective. It is also called a false positive.
EConfusionMatrixElement_CorrectlyClassifiedDefectiveSample	A CorrectlyClassifiedDefectiveSample is a defective images classified as defective. It is also called a true positive.
EConfusionMatrixElement_BadlyClassifiedDefectiveSample	A BadlyClassifiedDefectiveSample is a good images classified as good. It is also called a false negative.

6.33. EConnexity Enum

Possible values for the connexity of a contour.

Namespace: Euresys::Open_eVision

EConnexity_Connexity4	Pixels touching by an edge are considered connected.
EConnexity_Connexity8	Pixels touching by an edge or a corner are considered connected.

6.34. EContourMode Enum

Possible modes for contour traversal.

Namespace: Euresys::Open_eVision

EContourMode_Clockwise	The contour is traversed clockwise.
EContourMode_ClockwiseAlwaysClosed	The contour is traversed clockwise and the image border may be followed if necessary to close the contour.

EContourMode_ ClockwiseContinueIfBo rder	Contour traversal is restarted counterclockwise when an image border is met.
EContourMode_ Anticlockwise	The contour is traversed counterclockwise.
EContourMode_ AnticlockwiseContinueI fBorder	Contour traversal is restarted clockwise when an image border is met.
EContourMode_ AnticlockwiseAlwaysCl osed	The contour is traversed anticlockwise and the image border may be followed if necessary to close the contour.

6.35. EContourThreshold Enum

Allowed thresholding modes for contour traversal.

Namespace: Euresys::Open_eVision

EContourThreshold_ Above	Traverse the pixels just above the threshold.
EContourThreshold_ Below	Traverse the pixels just below the threshold.

6.36. ECorrelationMode Enum

Allowed values for the EasyMatch correlation mode.

Namespace: Euresys::Open_eVision

ECorrelationMode_ Standard	Correlation sensitive to changes in intensity and/or contrast (computed from the raw image/pattern gray values).
ECorrelationMode_ OffsetNormalized	Correlation made unsensitive to changes in intensity (computed from the centered image/pattern gray values).
ECorrelationMode_ GainNormalized	Correlation made unsensitive to changes in contrast (computed from the reduced image/pattern gray values).
ECorrelationMode_ Normalized	Correlation made unsensitive to changes in both intensity and contrast (computed from the centered and reduced image/pattern gray values). Default mode.

6.37. EDatasetType Enum

Type of dataset split.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EDatasetType_Training	Training dataset (images used to train the model).
EDatasetType_Validation	Validation dataset (images used to select the model ith the best performance during training).
EDatasetType_Test	Test dataset (images not used during training).
EDatasetType_All	Training, validaton, or test dataset.

6.38. EDataSize Enum

Possible data sizes for an object feature.

Namespace: Euresys::Open_eVision

EDataSize_BitsPerPixel1	bit
EDataSize_BitsPerPixel8	byte
EDataSize_BitsPerPixel16	word
EDataSize_BitsPerPixel32	long word
EDataSize_BitsPerPixel64	quad word
EDataSize_BitsPerPixel24	3 bytes
EDataSize_BitsPerPixel96	12 bytes

6.39. EDataType Enum

Possible data types for an object feature.

Namespace: Euresys::Open_eVision

EDataType_UnsignedInt	Unsigned integer.

EDataType_SignedInt	Signed integer.
EDataType_Float	Floating-point.

6.40. EDeepLearningDeviceType Enum

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

EDeepLearningDeviceType_CPU	-
EDeepLearningDeviceType_GPU	-
EDeepLearningDeviceType_Other	-
EDeepLearningDeviceType_NotADevice	-

6.41. EDeepLearningInferencePrecision Enum

Precisions supported by a Deep Learning device.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EDeepLearningInferencePrecision_INT8	-
EDeepLearningInferencePrecision_FLOAT16	-
EDeepLearningInferencePrecision_FLOAT32	-

6.42. EDeepLearningToolType Enum

-

Namespace: Euresys::Open_eVision::EasyDeepLearning

EDeepLearningToolType_None	-

EDeepLearningToolType_EasyClassify	-
EDeepLearningToolType_EasySegmentUnsupervised	-
EDeepLearningToolType_EasySegmentSupervised	-
EDeepLearningToolType_EasyLocate	-
EDeepLearningToolType_EasyLocateInterestPoint	-

6.43. EDLDataAugmentationType Enum

EDLDataAugmentationType represents how the data augmentation transformation are generated.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EDLDataAugmentationType_Random	Random transformation between the lower and upper limits
EDLDataAugmentationType_LowerLimit	Transformation using the lower limits.
EDLDataAugmentationType_UpperLimit	Transformation using the upper limits.

6.44. EDongleType Enum

Dongle types.

Namespace: Euresys::Open_eVision

EDongleType_Legacy	Legacy Dongle
EDongleType_Neo	Neo Dongle
EDongleType_Unknown	All Dongles

6.45. EDoubleThresholdMode Enum

The double threshold mode for the selection of coded elements with respect to a given feature.

Namespace: Euresys::Open_eVision

EDoubleThresholdMode_Inside	The value of the feature must be greater or equal to the low threshold, and strictly less than the high threshold.
EDoubleThresholdMode_Outside	The value of the feature must be strictly less than the low threshold, or greater or equal to the high threshold.

6.46. EDraggingMode Enum

Defines how the shape could be dragged

Namespace: Euresys::Open_eVision

EDraggingMode_Standard	Allows positioning the shape edges symmetrically.
EDraggingMode_ToEdges	Allows positioning each shape edge individually.

6.47. EDragHandle Enum

Allowed values for a handle identifier in the context of handle dragging.

Namespace: Euresys::Open_eVision

EDragHandle_NoHandle	No handle.
EDragHandle_Inside	Inside handle.
EDragHandle_North	Northern handle.
EDragHandle_East	Eastern handle.
EDragHandle_South	Southern handle.
EDragHandle_West	Western handle.
EDragHandle_NorthWest	North-Western handle.
EDragHandle_SouthWest	South-Western handle.

EDragHandle_ NorthEast	North-Eastern handle.
EDragHandle_ SouthEast	South-Eastern handle.
EDragHandle_Center	Circle, rectangle or wedge center handle.
EDragHandle_Org	Line segment or circle arc origin handle.
EDragHandle_Mid	Line segment or circle arc middle handle.
EDragHandle_End	Line segment or circle arc end handle.
EDragHandle_Tol0	First tolerance handle.
EDragHandle_Tol1	Second tolerance handle.
EDragHandle_Tol_x0	Rectangle leftmost first tolerance handle.
EDragHandle_Tol_x1	Rectangle leftmost second tolerance handle.
EDragHandle_Tol_y0	Rectangle lower first tolerance handle.
EDragHandle_Tol_y1	Rectangle lower second tolerance handle.
EDragHandle_Tol_XX0	Rectangle rightmost first tolerance handle.
EDragHandle_Tol_XX1	Rectangle rightmost second tolerance handle.
EDragHandle_Tol_YY0	Rectangle upper first tolerance handle.
EDragHandle_Tol_YY1	Rectangle upper second tolerance handle.
EDragHandle_Vertex	Polygon vertex handle.
EDragHandle_Tol_a0	Wedge leftmost first tolerance handle.
EDragHandle_Tol_a1	Wedge leftmost second tolerance handle.
EDragHandle_Tol_AA0	Wedge rightmost first tolerance handle.
EDragHandle_Tol_AA1	Wedge rightmost second tolerance handle.
EDragHandle_Tol_r0	Wedge inner first tolerance handle.
EDragHandle_Tol_r1	Wedge inner second tolerance handle.
EDragHandle_Tol_RR0	Wedge outer first tolerance handle.
EDragHandle_Tol_RR1	Wedge outer second tolerance handle.
EDragHandle_Edge_x	Rectangle leftmost edge handle.
EDragHandle_Edge_XX	Rectangle rightmost edge handle.
EDragHandle_Edge_y	Rectangle lower edge handle.
EDragHandle_Edge_YY	Rectangle upper edge handle.
EDragHandle_Edge_a	Wedge leftmost edge handle.
EDragHandle_Edge_AA	Wedge rightmost edge handle.
EDragHandle_Edge_r	Wedge outer edge handle.
EDragHandle_Edge_RR	Wedge inner edge handle.

6.48. EDrawableFeature Enum

The various features that can be drawn for coded elements.

Namespace: Euresys::Open_eVision

EDrawableFeature_BoundingBox	The bounding box.
EDrawableFeature_ConvexHull	The convex hull.
EDrawableFeature_Ellipse	The ellipse of inertia.
EDrawableFeature_FeretBox22	The Feret box oriented at 22.5 degrees.
EDrawableFeature_FeretBox45	The Feret box oriented at 45 degrees.
EDrawableFeature_FeretBox68	The Feret box oriented at 67.5 degrees.
EDrawableFeature_GravityCenter	The gravity center.
EDrawableFeature_MinimumEnclosingRectangle	The minimum enclosing rectangle.
EDrawableFeature_FeretBox	The Feret box oriented at a fixed angle. Only available for selections of coded elements: The angle of interest is set through the EObjectSelection::FeretAngle property.
EDrawableFeature_WeightedGravityCenter	The gravity center of the pixels of the attached image over the coded element. Only available for selections of coded elements: The attached image is set through the EObjectSelection::AttachedImage property.
EDrawableFeature_Contour	The contour of the object.

6.49. EDrawingMode Enum

Allowed modes to draw the bounding box of a symbol.

Namespace: Euresys::Open_eVision

EDrawingMode_Nominal	Draws the nominal point location or model fitting gauge.
EDrawingMode_Actual	Draws the located point or the fitted model.
EDrawingMode_SampledPaths	Draws the sampled segments along the model.

EDrawingMode_SampledPath	Draws the sampled segment specified by ELineGauge::MeasureSample.
EDrawingMode_PointsInSkipRange	Draws the skipped sampled points in addition to the non-skipped points.
EDrawingMode_SampledPoints	Draws the sampled points along the model.
EDrawingMode_SampledPoint	Draws the sampled point specified by ELineGauge::MeasureSample.
EDrawingMode_InvalidSampledPoints	Draws the invalid sampled points along the model.
EDrawingMode_Learn	Draw the pattern learning ROI(s).
EDrawingMode_Match	Draw the pattern searching ROI(s).
EDrawingMode_Position	Draw the found pattern ROI(s).
EDrawingMode_Inspected	Draw the inspected ROI.
EDrawingMode_MaxInspected	Draw the largest possible inspected ROI.

6.50. EEditionMode Enum

This enumeration is used to select which graphical interactions are allowed.

Namespace: Euresys::Open_eVision

EEditionMode_None	The object cannot be edited.
EEditionMode_Move	The object can be moved.
EEditionMode_Rotate	The object can be rotated.
EEditionMode_Stretch	The object can be stretched.
EEditionMode_EdgeSplit	The object can have one of its edges split.
EEditionMode_All	All graphical interactions are allowed.

6.51. EEncodingConnexity Enum

The connexity mode for the encoding process.

Namespace: Euresys::Open_eVision

EEncodingConnexity_Four	Pixels touching by an edge are considered connected.
EEncodingConnexity_Eight	Pixels touching by an edge or a corner are considered connected.

6.52. EError Enum

Possible Open eVision error codes.

Namespace: Euresys::Open_eVision

EError_Ok	Success
EError_EndOfImageSequence	End of image sequence
EError_UserDialogFailed	User dialog failed
EError_ImageLimitsReached	Image limits reached
EError_InvalidAsciiPadding	Invalid ASCII padding
EError_InvalidOperation	Invalid operation - See Reference
EError_InvalidBitsPerPixel	Invalid depth (bits per pixel) - Check type compatibility
EError_InvalidDataType	Invalid data type - Check type compatibility
EError_InvalidDataSize	Invalid data size - Check type compatibility
EError_ParametersOutOfRange	Parameters out of range - See Reference
EError_InvalidMode	Invalid mode - See Reference
EError_EndSmallerThanStart	End smaller than start - Adjust indices
EError_Parameter1OutOfRange	Parameter 1 out of range - See Reference
EError_Parameter2OutOfRange	Parameter 2 out of range - See Reference
EError_Parameter3OutOfRange	Parameter 3 out of range - See Reference
EError_Parameter4OutOfRange	Parameter 4 out of range - See Reference
EError_Parameter5OutOfRange	Parameter 5 out of range - See Reference

EError_ Parameter6OutOfRange	Parameter 6 out of range - See Reference
EError_ Parameter7OutOfRange	Parameter 7 out of range - See Reference
EError_ Parameter8OutOfRange	Parameter 8 out of range - See Reference
EError_ Parameter9OutOfRange	Parameter 9 out of range - See Reference
EError_ Parameter10OutOfRange	Parameter 10 out of range - See Reference
EError_WindowsError	Out of GDI handles
EError_ InvalidPlanesPerPixel	Invalid planes per pixel - Check image type or file contents
EError_ BW1ImageExpected	1 bit black & white (BW1) image expected - Check image type or file contents
EError_ BW8ImageExpected	8 bits black & white (BW8) image expected - Check image type or file contents
EError_ BW16ImageExpected	16 bits black & white (BW16) image expected - Check image type or file contents
EError_ BW32ImageExpected	32 bits black & white (BW32) image expected - Check image type or file contents
EError_ TemplateCallNeedsSpecialization	Template call needs specialization
EError_ CannotCreateMutex	Cannot create Mutex
EError_ CannotLockMutex	Cannot lock Mutex
EError_ CannotUnlockMutex	Cannot unlock Mutex
EError_ CannotDeleteMutex	Cannot delete Mutex
EError_ TimeoutReached	Timeout reached
EError_ FunctionNotFound	Function not found
EError_ProcessStopped	Process stopped
EError_ CopyNotAllowed	Copy not allowed
EError_SingularMatrix	Internal error (code: 1050)
EError_DivisionByZero	Division by zero - Check parameters

EError_ ReadonlyProperty	This property is read-only and cannot be accessed
EError_ UndefinedProperty	This property is undefined and cannot be accessed
EError_ItemNotFound	The data structure does not contain the specified item
EError_ NextItemNotFound	The specified item has no next sibling
EError_ZeroDimension	Width and height must be different from zero.
EError_ StringConvertionFailed	String convertion failed.
EError_ FileAccessProblems	File access problems - Check file pathname and device state
EError_ FileCouldNotBeOpened	File could not be opened - Check file pathname, troubleshoot disk device
EError_FilwhileReading	File error while reading - Check file integrity, troubleshoot disk device
EError_FilwhileWriting	File error while writing - Check free disk space, troubleshoot disk device
EError_BadFileFormat	Bad file format - Check file source or contents
EError_ FileCouldNotBeClosed	File could not be closed - Check free disk space, troubleshoot disk device
EError_ UnsupportedFileFormatVersion	Unsupported file format version - Upgrade to newer release
EError_ MissingOrUnsupportedFileExtension	Missing or unsupported file extension - Check file name/format match
EError_FileIsReadOnly	File is read-only - Set to read-write or save under another name
EError_ UnsupportedObjectTypeInArchive	The archive does not contain the correct object type - Check your archiving routines
EError_ UnknownArchiveError	An unexpected error occurred during archive access
EError_ SerializerShouldBeInReadMode	Attempting to read with a serializer not open for read access
EError_ SerializerShouldBeInWriteMode	Attempting to write with a serializer not open for write access
EError_FileExists	Attempting to overwrite an existing file while not allowed to do so
EError_ SerializerNotOpen	Attempting to use a serializer that is not correctly opened

EError_ UnrecognizedFileFormat	Unrecognized File Format
EError_ WrongColorFormatFileFormatCombination	Wrong Color Format File Format Combination
EError_ FileDoesNotExist	Attempting to open a file which doesn't exist
EError_ ObjectTooLargeToBeSerialized	Object is too large to be serialized
EError_ UnsupportedFileFormat	Unsupported File Format
EError_ ResourcesDirectoryNotFound	Resources directory doesn't exist or cannot be opened.
EError_ ResourceFileNotFound	Resource doesn't exist or cannot be opened.
EError_ UnsupportedTiffFormat	Unsupported TIFF format - Convert TIFF file
EError_ UnsupportedBmpFormat	Unsupported BMP format - Convert BMP file
EError_ InvalidPngCompression	Invalid PNG compression level
EError_ UnsupportedJpegFormat	Unsupported JPEG format - Convert JPEG file
EError_ BilevelImageExpected	Bi-level image expected - Use BW1
EError_ GrayLevelImageExpected	Gray-level image expected - Use BW8
EError_ ColorImageExpected	Color image expected - Use C24
EError_ BilevelFormatExpected	Bi-level format expected - Convert file to black & white
EError_ GrayLevelFormatExpected	Gray-level format expected - Convert file to gray shades
EError_ ColorFormatExpected	Color format expected - Convert file to true color

EError_CannotReadJpegFile	Cannot read JPEG file - Troubleshoot disk device
EError_CannotWriteJpegFile	Cannot write JPEG file - Troubleshoot disk device
EError_WrongFileExtension	Wrong file extension
EError_UnableToAllocateTemporaryMemory	Unable to allocate temporary memory - Fix memory leaks or release memory
EError_BufferTooSmall	The supplied buffer is too small. Supply a bigger buffer.
EError_UnableToAllocateMemory	Not enough memory for this allocation.
EError_UnableToAccessImageMemory	Unable to access image memory - Load or size image
EError_RoiTooLarge	ROI too large - Fit ROI to image
EError_NotAValidImage	Not a valid image - Check image contents
EError_ImagesNotSameSize	Images not of the same size - Adjust image size(s)
EError_ImagesNotSameBitsPerPixel	Images not of the same depth (bits per pixel) - Choose compatible types
EError_SourceImageTooSmall	Origin image too small - Use a larger image
EError_PixelsMustHaveFiniteSize	Pixels must have finite size - Use non-zero parameters
EError_ConstantIsNull	Constant is NULL - Use non-zero value
EError_PixelNullEncountered	NULL pixel encountered - Avoid division by zero
EError_ImagesMayNotOverlap	Images cannot overlap - Use distinct images
EError_RoiOutOfImageLimits	ROI out of the top parent limits - Resize to fit in image
EError_RoiAlreadyHasAParent	ROI already has a parent - Detach ROI first
EError_RoiHasNoParentImage	There is no image ancestor for this ROI - Attach the ROI or one of its ancestor to an image
EError_CannotApplyToAnImage	Cannot apply to an image - Apply to a ROI instead

EError_UnsupportedImageType	Unsupported image type - Check type compatibility
EError_InvalidImageType	Invalid image type - Check type compatibility
EError_UnsupportedXserverDepth	Unsupported X server depth
EError_InconsistentRoiHierarchy	The hierarchy of ROI has been corrupted (inconsistent parent/daughters relationship)
EError_SourceImageTooBig	Original image too big - Use a smaller image
EError_BW1RoiNotAligned	First bit index of an aligned ROI must be 0 - Use an aligned ROI
EError_WrongRoiType	Wrong ROI or image type
EError_CyclingParenthoodNotAllowed	Cycling Parenthood not allowed
EError_WrongBitsPerRow	Bits per row must be a multiple of 32 (4 bytes) and must be enough to hold all the pixels of an image row.
EError_MisalignedImagePtr	The supplied image pointer must be aligned to 4 bytes.
EError_UnsupportedImageTypeConversion	Unsupported image type conversion
EError_ImageFromFileDoesNotFitIntoROI	The ROI is not the same size as the image file. When loading an image from a file into a ROI, the ROI must have the exact required size. On the other, when loading into an image object, it gets resized to the correct size.
EError_PixelCoordinatesOutOfROI	The specified coordinate is outside the ROI/Image
EError_ROIFromFileDoesNotFitIntoROI	The ROI is not the same size as the ROI previsously saved. ROIs directly linked to an pixel container must have the same size as the container and have a (0, 0) origin.
EError_ROIHasZeroArea	The ROI width and height must both be larger than 0 - resize the ROI.
EError_RegionTooSmall	The region is too small.
EError_ERegionHasNotBeenPrepared	The ERegion has not been prepared, please use function ERegion::Prepare().
EError_ERegionImpossibleCopy	Cannot copy non-prepared geometrical region into a base ERegion instance.

EError_ CanvasSizeNotSet	Canvas size not set.
EError_ RegionOutsideImageOrRoi	ERegion is (partially) outside of the corresponding image/ROI
EError_ PixelOutsidePerimeter	Pixel outside perimeter - Check pixel value
EError_ PixelInsidePerimeter	Pixel inside perimeter - Check pixel value
EError_IsolatedPixel	Isolated pixel - Check pixel value
EError_ MaxPixelInContourReached	Maximum pixels in contour reached
EError_ NotAValidContour	Not a valid contour - Initialize using a contouring function
EError_ UnableToAccessVectorMemory	Unable to access vector memory - Check proper vector initialization
EError_ NotAValidVectorDescriptor	Not a valid vector descriptor
EError_ VectorTypeIsNotHist	Vector type is not histogram
EError_ NotEnoughGroupsInVector	Not enough groups in vector
EError_ InvalidVectorDataSize	Invalid vector data size
EError_ InvalidVectorDataType	Invalid vector data type
EError_ InvalidVectorType	Invalid vector type
EError_ ResultTooBigToFitInVector	Result too big to fit in vector
EError_ GroupOutOfRange	Group out of range - Adjust group index
EError_ InvalidVectorGroupLength	Invalid vector group length
EError_ InvalidNumberOfVectorElements	Invalid number of vector elements - Check proper vector initialization

EError_VectorsNotSameSize	Vectors not of the same size - Adjust vector size(s)
EError_UnableToAccessKernelMemory	Unable to access kernel memory - Check proper kernel initialization
EError_NotAValidKernelDescriptor	Not a valid kernel descriptor
EError_InvalidKernel	Invalid kernel
EError_KernelInvalidSize	Invalid kernel size - Check proper kernel initialization
EError_KernelNotAllocated	Kernel not allocated - Check proper kernel initialization
EError_BadListPosition	Bad list position - Restart list traversal
EError_ListIsEmpty	List is empty
EError_TopOfList	Top of list - Do not traverse backwards
EError_BotOfList	Bottom of list - Do not traverse forwards
EError_ListError	List error
EError_LicenseMissing	The license for this library is not granted - Launch License Manager
EError_EasyImageLicenseMissing	The license for EasyImage is not granted - Launch License Manager
EError_EasyColorLicenseMissing	The license for EasyColor is not granted - Launch License Manager
EError_EasyObjectLicenseMissing	The license for EasyObject is not granted - Launch License Manager
EError_EasyMatchLicenseMissing	The license for EasyMatch is not granted - Launch License Manager
EError_EasyGaugeLicenseMissing	The license for EasyGauge is not granted - Launch License Manager
EError_EasyFindLicenseMissing	The license for EasyFind is not granted - Launch License Manager
EError_EasyOcrLicenseMissing	The license for EasyOCR is not granted - Launch License Manager
EError_EasyOcvLicenseMissing	The license for EasyOCV is not granted - Launch License Manager

EError_ EasyBarCodeLicenseMissing	The license for EasyBarCode is not granted - Launch License Manager
EError_ EasyMatrixCodeLicenseMissing	The license for EasyMatrixCode is not granted - Launch License Manager
EError_ EasyMatchAlignementModeLicenseMissing	The license for EasyMatch Alignment mode is not granted - Launch License Manager
EError_ EvisionStudioLicenseMissing	The license for eVison Studio is not granted - Launch License Manager
EError_ InvalidDongleIndex	The index do not match any available dongle
EError_ CannotWriteOEMKey	The OEM key cannot be set
EError_ OEMKeyIndexNotSupported	key Indexes are not supported by the selected DongleType
EError_ OEMKeyInvalidSize	The size of the OEM key is invalid, it should be between 8 and 64 chars
EError_ OEMKeyInvalidIndex	The index of the OEM key is invalid, it should be between 0 and 11
EError_ NoGradingComputed	This EBarCode was not graded
EError_ WarpImagesTooSmall	Warp images too small - Increase image size
EError_ UnsupportedImageSize	Unknown error code
EError_ InvalidThresholdValue	The threshold value is not supported
EError_ ImagesSizeIncompatible	The sizes of the images parameters is not compatible.
EError_ UnknownFeature	Unknown feature - Check parameters
EError_ InvalidSelectionArgument	Invalid selection argument - Check parameters
EError_ SortListTooLong	Sort list too long
EError_ NotAValidOperationCode	Not a valid operation code

EError_TooManyObjectsDetected	Too many objects detected - Increase MaxObjects
EError_InvalidFeature	Invalid feature - Check parameters
EError_FeatureNotCalculated	Feature not calculated - Call AnalyseObjects method
EError_BadObjectNumber	Bad object number - Check parameters
EError_NoObjectSelected	No object selected - Blob list is empty
EError_LowThresholdHigherThanHighThreshold	Low threshold higher than high threshold - Adjust thresholds
EError_InvalidThresholdMode	Invalid threshold mode - Use appropriate threshold setting method
EError_NoImageAttached	No image attached to the selection - Use Attach()
EError_OutOfContinuousMode	Invalid call out of continuous mode
EError_InvalidImageTypeForSegmenter	The current segmenter can not cope with this type of image
EError_LayersOverlapping	Two different layers are associated with the same layer index
EError_EndOfIterator	The iterator has reached the end of the enumeration
EError_NoThresholdComputedYet	The threshold valued has not been computed yet - First encode an image
EError_FeatureNotDrawable	This kind of feature cannot be drawn
EError_OnlyApplicableToObjectSelection	This kind of feature cannot be used out of EObjectSelection
EError_MoreThanOneLayerEncoded	Please specify the layer index (several layers are encoded)
EError_CodedElementNotSelected	The coded element is not present in the selection
EError_NoPatternLearnt	No pattern learnt - Load from file or train pattern
EError_PatternTooLarge	Pattern too large - Use a smaller one

EError_ PatternTooSmall	Pattern too small - Use a larger one
EError_ NotAnEasyMatchFile	Not an EasyMatch file - Check file source
EError_ UnsupportedEasyMatchFileVersion	Unsupported EasyMatch file version - Upgrade to a newer release
EError_NoImageLearnt	No image learnt - Call LearnImage() first
EError_ WrongNumberOfDegresOfFreedom	The number of degrees of freedom must be at least one, and no more than five - Use a value in this range
EError_ InsufficientDiscriminantFeaturesInPattern	There is not enough discriminant features in the selected region to learn a pattern
EError_Unknown_ Pattern_Style	The pattern style is not recognized
EError_ GrabCutNoImageSpecified	No image was specified to the GrabCut algorihtm. Set an image with SetImage().
EError_ GrabcutNotEnoughSamples	There is not enough background and/or foreground pixels to build a model and apply the GrabCut algorithm.
EError_ InsufficientContrast	Not enough feature points - Use a more contrasted pattern or reduce the Don't Care mask
EError_ PatternTooCloseToImageBorder	Pattern is too close to image border - Leave a margin around the pattern
EError_ IncompatibleModes	Incompatible modes (CoarseToFineAnalysisMode and PatternType)
EError_ AllowancesAndPatterntypeNotCompatible	Angle and Scale allowances can not be used with the current pattern type
EError_ ModelNotSuitedForContrastingregions	The model is unsuitable for ContrastingRegions pattern type - Try an other pattern type or increase surface of region(s)
EError_ ModelNotSuitedForConsistentedges	The model is unsuitable for ConsistentEdges pattern type - Try another pattern type or increase the model surface
EError_ OnlyConsistentedgesForVectorLearning	Learning using a vector model is only possible with the ConsistentEdges pattern type - Try ConsistentEdges pattern type
EError_ LoadedPatternCallsForContrastingRegions	The loaded pattern calls for Contrasting Regions pattern type, but Contrasting Regions pattern type is deprecated since Open eVision 23.08.

EError_ NoPatternsLoaded	No patterns loaded - Load font file or train
EError_ NoPatternsInTheseClasses	No patterns in these classes - Check pattern and text class assignments
EError_ CharacterTooSmall	Character too small - Enlarge to font size
EError_ CharacterCodeTooBig8	Character code too big to fit in a string, use ReadTextWide instead
EError_ CharacterCodeTooBig16	Character code too big to fit in a wide string, use GetFirstCharCode instead
EError_ InvalidTextStructure	Text parameter doesn't fit the text topology
EError_InvalidFontFile	The specified font file couldn't be loaded
EError_InvalidTopology	The specified topology is invalid
EError_ InvalidEOCR2File	The file-type and structure could not be verified
EError_ EOCR2InvalidCharWidth	Character widths must be larger than 0
EError_ EOCR2InvalidCharWidthTolerance	Character width tolerance must be between 0 and 1
EError_ EOCR2InvalidCharHeight	Character height must be larger than 0
EError_ EOCR2InvalidMaxVariation	The 'maximum variation' parameter must be between 0 and 1.
EError_ EOCR2InvalidDetectionDelta	The 'detection delta' parameter must be between 0 and 128.
EError_ EOCR2InvalidMaxFragmentation	The 'maximum fragmentation' parameter must be between 0 and 1.
EError_ EOCR2InvalidSpaceWidth	Space widths must be larger than or equal to 0.
EError_ EOCR2InvalidNumDetectionPasses	NumDetectionPasses must be either 1 or 2.
EError_ EOCR2CharCodeNotSet	Character code not set

EError_ EOCR2CharHeightNotSet	Character height not set
EError_ EOCR2CharWidthNotSet	Character width not set
EError_ EOCR2TopologyNotSet	Topology not set
EError_ EOCR2RangedTopologyNotSupported	Ranged topology is not supported with this method of detection.
EError_ EOCR2InvalidRelativeSpaceWidth	Relative space width must be larger than 0
EError_ EOCR2ClassiferNotFound	OCR2 Pre-trained classifier not found.
EError_ EOCR2ClassiferInvalid	OCR2 Pre-trained classifier is valid.
EError_ EOCR2TopologyNotSupported	OCR2 Topology not supported by the deep-learning classifier
EError_ EOCR2DetectionFailed	The given topology and parameters could not be fitted to the detected blobs.
EError_ MismatchingColorSystem	Mismatching color system - Check transform compatibility
EError_ ColorLookupMustBeInitialized	Color lookup must be initialized - Use initialization method
EError_ UnsupportedColorTransform	Unsupported color transform
EError_ UnknownSymbolSize	Unknown symbol size - Check size initialization
EError_ UnknownEccFamily	Unknown ECC family (ECC 000/050/080/100/140/200 only)
EError_ UncorrectableErrors	Too many errors, cannot correct contents - See Reference
EError_ CouldNotLocateSymbol	Could not locate the dot matrix symbol (no good candidate object) - See Reference
EError_ UnknownFormatId	Unknown Format ID in ECC 000-140 symbol (Base 11/27/41/37 and ASCII 7/8 only) - See Reference

EError_InvalidCrc	Invalid CRC after error correction in ECC 000-140 symbol - See Reference
EError_NoCodeFound	Could not find any codes in the image
EError_TimeoutReachedAndNoCodeFound	Could not find any codes in the image within the timeout
EError_CouldNotDecodeSymbol	Could not decode symbol - Try to improve image quality
EError_CouldNotGrade	Could not grade symbol - Quiet zone out of bounds
EError_CouldNotLocateBarcode	Could not locate bar code symbol - Improve contrast, avoid clutter
EError_UnrecognizedSignature	Unrecognized signature - Check enabled symbologies
EError_InvalidNumberOfBars	Invalid number of bars - Improve bar/space contrast
EError_ExtraEdgesFound	Extra edges found - Improve bar/space contrast or uniformity
EError_IncoherentBarSpaceThickness	Incoherent bar/space thickness sequence - Check enabled symbologies
EError_InvalidCheckCharacter	Invalid checksum character - Check enabled symbologies
EError_SymbologyNotEnabled	Symbology not enabled - Invoke method SetSymbologies()
EError_NoEdgesFound	No edges found - Adjust location or improve bar/space contrast
EError_InvalidEMailBarcodeReaderFile	The file-type and structure could not be verified
EError_InvalidGS1String	The given string could not be parsed as a GS1 machine readable string
EError_NotAnEasyOcvFile	Not an EasyOCV file - Check file source
EError_UnsupportedEasyOcvFileVersion	Unsupported EasyOCV file version - Upgrade Open eVision
EError_NotEnoughSampleImages	Not enough sample images - Use AddToStatistics
EError_NotAnEcheckerFile	Not an EChecker file - Check file source

EError_ UnsupportedEcheckerFileVersion	Unsupported EChecker file version - Upgrade Open eVision
EError_ NotEnoughSamplesLearnt	Not enough samples learnt - Use UpdateStatistics
EError_ InvalidNormalizationMode	Invalid normalization mode - Check SetNormalize call
EError_ ImageNotRegistered	Image not registered - Use method Register before Learn
EError_ InvalidLearningSequence	Invalid learning sequence - Use AVERAGE followed by ABS_DEVIATION, or RMS_DEVIATION, then READY
EError_E_ERROR_ CONTRAST_TOO_LOW	Image contrast is too low
EError_ MotherAlreadyHasThisDaughter	Mother already has this daughter - Detach daughter first
EError_ ShapeAlreadyHasDaughters	Shape already has daughters - Detach daughters first
EError_ NoValidPointFound	No valid point found in the transition computation.
EError_ NotInListAttachmentMode	Not in list attachment mode - Detach daughters first
EError_ NotInIndexedAttachmentMode	Not in indexed attachment mode - Detach daughters and call SetIndexed first
EError_ UnsupportedShapeVersion	Unsupported shape version - Upgrade Open eVision
EError_ RawCalibrationMode	Raw calibration mode - Cannot be used for this operation
EError_ BadLandmarkLayout	The layout of supplied landmarks makes the calibration impossible - Check landmarks positions
EError_ IncompatibleCalibrationModes	Incompatible calibration modes - Check calibration mode categories
EError_ NotEnoughLandmarks	Not enough landmarks to calibrate - Add landmarks or check calibration mode categories

EError_UnexpectedShapeTypeInFile	Unexpected shape type in file - Check target shape against file model root
EError_UnsupportedModelFileVersion	Unsupported model file version - Upgrade Open eVision
EError_CannotAttachDetachWorldShapes	Cannot Attach or Detach World shape - World shapes never have a mother
EError_UnexpectedWorldShapeInFile	Unexpected World Shape in file - Check target shape against file model root
EError_UnexpectedFrameShapeInFile	Unexpected Frame Shape in file - Check target shape against file model root
EError_UnexpectedPointShapeInFile	Unexpected Point Shape in file - Check target shape against file model root
EError_UnexpectedLineShapeInFile	Unexpected Line Shape in file - Check target shape against file model root
EError_UnexpectedCircleShapeInFile	Unexpected Circle Shape in file - Check target shape against file model root
EError_UnexpectedRectangleShapeInFile	Unexpected Rectangle Shape in file - Check target shape against file model root
EError_UnexpectedWedgeShapeInFile	Unexpected Wedge Shape in file - Check target shape against file model root
EError_UnexpectedPointGaugeInFile	Unexpected Point Gauge in file - Check target shape against file model root
EError_UnexpectedLineGaugeInFile	Unexpected Line Gauge in file - Check target shape against file model root
EError_UnexpectedCircleGaugeInFile	Unexpected Circle Gauge in file - Check target shape against file model root
EError_UnexpectedRectangleGaugeInFile	Unexpected Rectangle Gauge in file - Check target shape against file model root
EError_UnexpectedWedgeGaugeInFile	Unexpected Wedge Gauge in file - Check target shape against file model root

EError_UnexpectedPolygonGaugeInFile	Unexpected Polygon Gauge in file - Check target shape against file model root
EError_UnexpectedBarCodeInFile	Unexpected Bar Code model in file - Check target shape against file model root
EError_AnActiveCurvedEdgeIsRequired	At least one curved edge must be active - Activate r and/or R edges
EError_BrokenWedgeShapeConstraints	Constraints between the geometric and the tolerance of the wedge are broken
EError_NotEnoughVertices	The polygon is missing one or several vertices to be valid.
EError_InvalidGrid	The detected grid is invalid
EError_InvalidSymbolSize	The detected symbol size is invalid
EError_InvalidFixedPattern	The fixed pattern of the detected code is invalid
EError_LearnInvalidImageSize	The dimensions of the images should be the same after a learn.
EError_InvalidDimensionRangeLearned	No valid datamatrices dimensions have been learned.
EError_QRECIByteInterpretationTableNotSupported	The byte interpretation is dictated by the ECI coding mode.
EError_QRByteEncodingUnknownInterpretationMode	The byte interpretation is dictated by the ECI coding mode.
EError_QRByteInterpretationModeParameterNotCompatibleWithContent	The byte interpretation is dictated by the ECI coding mode.
EError_CalibrationModelNotDefined	A 3D calibration model is required to perform the conversion
EError_InvalidE3DModelFile	The file-type and structure could not be verified
EError_EmptyPointCloud	The point cloud should not be empty
EError_WrongOrientationVector	The supplied orientation vector is not correct

EError_InvalidE3DCalibrationGeneratorFile	The file-type and structure could not be verified
EError_CalibrationModelNotInitialized	A 3D calibration model is not Initialized
EError_InvalidE3DConverterFile	The file-type and structure of the E3D converter file could not be verified
EError_InvalidE3DCalibrationFile	The file-type and structure of the E3D calibration file could not be verified
EError_UnknownCalibrationObjectType	The calibration object type is not set
EError_ResultOutOfTolerances	Result out of tolerances
EError_MalformedTriangleIndexes	The triangle indexes are not correct in E3DObject
EError_FitFailed	The 3D fit operation failed
EError_ParamNotSet	Trying to get a parameter value that has not been set
EError_UndefinedPixelValue	The pixel has undefined value
EError_FindFailed	The 3D find operation failed
EError_AlignFailed	The 3D align operation failed
EError_WrongCalibrationParameters	Wrong calibration parameters
EError_WrongNormalVector	Wrong normal vector
EError_WrongNormalTolerance	Wrong normal angle tolerance
EError_CalibrationModelNotFound	A 3D calibration model is not found
EError_AxesNotNormal	The axis system is not normed
EError_AxesNotRightHanded	The axis system is not righthanded
EError_AxesNotOrthogonal	The axis system is not orthogonal
EError_MatrixNotRigid	A matrix is not rigid

EError_ AxisSystemNotRigid	An axis system is not rigid
EError_ CoordinatesOutOfMap	The coordinates are out of the map
EError_ InvalidAxisSystem	Axis system is invalid
EError_ InvalidPointCloud	The point cloud is not valid
EError_ PointCloudOutOfRange	The point cloud is out of range
EError_ DepthMapNotCompatibleWithCalibration	The depth map point is not compatible with the calibration model (wrong association)
EError_ InvalidE3DObjectFile	The file-type and structure of the E3D object file could not be verified
EError_ Map3DConversionModeMustBeInitialized	Conversion mode must be initialized
EError_ InvalidE3DBoxFile	The file-type and structure of the E3DBox file could not be verified
EError_ NoE3DPointFound	No 3D point found.
EError_ OutOffSpacePartition	The point or the index is not in the range of space partition.
EError_ NoIntersectionFound	No intersection found.
EError_ AttributeBufferNotInitialized	The attribute buffer is not initialized.
EError_ IncompatibleAttributeTypeConversion	The attribute type can not be converted to the destination type.
EError_ PhotometricStereoImagerNotInitialized	The PhotometricStereoImager is not initialized.
EError_ SphereDetectionFailed	The sphere detection failed
EError_ InvalidLightDirections	The light directions given or deduced from calibration are coplanar. Photometric stereo cannot be performed with such lights.
EError_ PhotometricStereoImagerNoComputationDone	The PhotometricStereoImager has not done any computation on new images since last calibration.

EError_PhotometricStereoImagerLightingCorrectionNotConfigured	The non uniform lightning correction must be configured before being enabled.
EError_PhotometricStereoImagerLightingCorrectionBadImageSize	The flat images size must be the same as the object image size.
EError_SphereOutsideOfROI	The sphere is outside of the ROI
EError_E3DViewerNotInitialized	The 3D viewer is not initialized
EError_E3DMatchNoReferenceModel	E3DMatch class was not provided a reference model
EError_E3DMatchEmptyModel	E3DMatch class was provided an empty model
EError_E3DMatchMatchFailed	E3DMatch class could not find a match
EError_EEmptyZMap	EZMap provided does not contain any defined pixel
EError_E3DMatchNoReferencePose	E3DMatch class was not provided a reference pose
EError_InvalidEColorRampMode	The EColorRampMode is not valid with an attribute buffer id.
EError_E3DMatchNoReferencePlane	E3DMatch class was not provided a reference plane
EError_IncompatibleAttribute	The attribute type can not be used for the operation.
EError_E3DMatchNoDistanceComputed	There is no distance computed.
EError_RenderSourceNameUnknown	The render source name is unknown
EError_RenderSourceAlreadyExists	The render source name already exists
EError_EPointCloudMergerCalibrationFailed	EPointCloudMerger's calibration failed

EError_EPointCloudMergerCalibrationModelNotFound	EPointCloudMerger calibration model's file not found
EError_E3DViewerPickingCallBackException	An exception occurred while calling the callback function after having picked a point.
EError_ColorRampFixBoundSDisabled	The fix bounds cannot be retrieved from the color ramp because it is disabled.
EError_E3DPointCloudInvalidSegment	The given line segment is invalid, for example, because the start and end points are the same.
EError_ESphereFitterNotEnoughPointsSupplied	ESphereFitter does not have enough points to generate a sphere
EError_ESphereFitterAllPointsCoplanar	ESphereFitter does not have enough points to generate a sphere
EError_FeatureNotComputed	The requested feature has not been computed.
EError_RadiusShouldBePositive	The radius of a E3DSphere cannot be negative.
EError_ESphereFitterCouldNotFit	The radius of the fitted sphere is negative.
EError_NoColorRampForShapes	Color ramp cannot be used as source color mode for primitives such as boxes, planes and spheres.
EError_OpenGLError	OpenGL Error.
EError_E3DMatchModelTooCoarse	The model given to the E3DMatch class does not contain enough points for it to work properly, try with a denser model.
EError_OpenGLInitError	Failed to initialize OpenGL. Your platform is not able to use the class E3DViewer.
EError_OpenGLESNotSupported	E3DViewer doesn't support OpenGL ES.
EError_DataAugmentationFailed	Data augmentation failed for an unknown reason.
EError_InvalidInputSpecification	Invalid input specification.

EError_ LabelDoesNotExist	The label does not exist in the dataset.
EError_ ImageIsNotAPath	The image was not given as a path.
EError_ ImageDoesNotConformToInputFormat	The images do not conform to the input format of the deep learning tool.
EError_ CannotDisableAutoreshape	The automatic procedure to make every image conform to the input specification cannot be disabled because there already are images in the dataset that do not conform to the input specification.
EError_ NotEnoughImagesToSplitDataset	There are not enough images in the dataset to perform a split such that each part has at least one image from each label of the original dataset.
EError_ NotAvailableIn32Bits	This method is not available for the 32 bits binaries of Open eVision.
EError_ DatasetIncompatibleWithDeepLearningTool	The dataset is incompatible with this deep learning tool. A trained tool can add constraints on some of the properties of the tool.
EError_ DeepLearningToolCurrentlyTraining	This operation is impossible because this deep learning tool is currently training.
EError_ DeepLearningToolNotTraining	Cannot wait because this deep learning tool is not training.
EError_ CannotChangeInputSpecification	A pre-trained classifier comes with some input specifications that can't be changed.
EError_ TrainingAndValidationDatasetIncompatible	The training and validation dataset have incompatible image format.
EError_ TrainingAndValidationLabelsIncompatible	The training and validation dataset have incompatible labels.
EError_ ClassifierTrainedWithIncompatibleLabels	The classifier was previously trained with incompatible labels.
EError_ CannotChangeClassifierType	The type of classifier can't be changed once the classifier is trained.
EError_ UnknownClassifierType	Unknown classifier type.
EError_ DeepLearningToolNotTrained	This deep learning tool is not trained.

EError_NoGPUFound	No GPU was found.
EError_InvalidMetrics	The metrics are not valid.
EError_MetricsIncompatibleWithResult	The metrics are incompatible with the given result.
EError_InvalidResult	The classification result is invalid.
EError_HeatmapGenerationFailure	Can not generate Heatmap.
EError_NotEnoughMemoryForTraining	There is not enough free memory on the CPU or GPU to perform the training.
EError_NotEnoughMemoryForPrediction	There is not enough free memory on the CPU or GPU to perform a prediction.
EError_NotEnoughMemoryForBatchPrediction	There is not enough free memory on the CPU or GPU to perform a batch prediction.
EError_LayerOutputDisabled	The layer has its output disabled.
EError_NotEnoughMemoryForCache	There is not enough free memory on the CPU for storing the dataset images in the cache.
EError_DeepLearningToolTrained	This operation is impossible because this deep learning tool is already trained.
EError_DeepLearningToolCannotStartTraining	There was an error while starting the training thread.
EError_LabelAlreadyExists	The label already exists in the dataset.
EError_LabelCannotBeChangedOrRemoved	The label can't be changed or removed from the dataset.
EError_IncompatibleLabels	The labels are not compatible with each other.
EError_InvalidLabelWeight	The label weight is invalid. It must be bigger than 0.
EError_ImageHasNoSegmentation	The image has no segmentation.
EError_ImageHasNoLabel	The image has no classification label.

EError_ ResultHasNoGroundtruth	The result doesn't have any groundtruth associated with it.
EError_ DeepLearningModelError	There was an error in the deep learning model.
EError_ ImageNotLabelledForObjectDetection	The image is not labelled for object detection (see ELocator).
EError_ InvalidLocatorObject	The locator object is invalid (no label or empty region).
EError_ RectangleRegionNotAxisAligned	The rectangle region is not axis aligned.
EError_ CapacityNotAvailable	The rectangle region is not axis aligned.
EError_ NoInferenceModel	No inference model is present in the tool.
EError_ NoTrainingModel	No training model is present in the tool.
EError_ PretrainedModelError	Pretrained model could not be loaded.
EError_ HeatmapNotAvailable	The heatmap is not available.
EError_ UndefinedDeepLearningProject	The deep learning project is undefined. Cannot save or load.
EError_ DeepLearningProjectDirectoryError	Directory doesn't exist or cannot be created.
EError_ IncompatibleDeepLearningToolType	The tool type is not compatible with the operation.
EError_ DatasetTypeLocked	The dataset type for this image is locked.
EError_ CannotChangeLayerParameter	This parameter of the layer cannot be changed once the layer is initialized.
EError_ LocatorObjectHasNoSize	The locator object has no size.
EError_ CannotSetAnchorsForEasyLocateInterestPoint	TCannot manually set anchors the EasyLocate Interest Point.

EError_LocatorObjectHasNoPosition	The locator object has no position.
EError_CannotParseOnnxFile	This onnx file cannot be parsed.
EError_OnnxModelDoesNotHaveGraph	This onnx model does not have a graph.
EError_OnnxModelIsNotAClassifier	This onnx model is not a classifier.
EError_OnnxModelDoesNotTakeOneInput	This onnx model does not take one input.
EError_OnnxModelAttributeNotDefined	A mandatory attribute is not defined.
EError_OnnxOperatorNotSupported	This onnx model is using an unsupported operator.
EError_OnnxLastLayerIsNotSoftmax	The last layer of the model is not a softmax.
EError_OnnxNumlabelsUndefined	The number of labels is undefined.
EError_OnnxUnknownArgument	Unknown argument
EError_ModelNotFound	The specified neural network model was not found.
EError_CudaError	An error occurred in a NVIDIA CUDA library.
EError_DeepLearningEngineError	An error occurred when executing the model with a Deep Learning engine.
EError_InferenceError	There was an unknown error during inference (e.g. not enough memory).
EError_AlignmentFailed	Alignment failed, please check AlignmentArea and AlignmentTolerance.
EError_InvalidESpotDetectorFile	The file-type and structure of the spot detector file could not be verified.
EError_InvalidESpotFile	The file-type and structure of the spot file could not be verified.
EError_DLClassifierNotSet	The Deep Learning classifier is not set.

EError_PixelHandling	Internal error during image processing
EError_EmptyMorphologicalKernel	Use of a morphological kernel without any element set
EError_MatrixOperation	Internal error during matrix processing
EError_NonSquareMatrix	The operation is only valid for square matrices
EError_IncompatibleMatrixSizes	The sizes of the matrix are incompatible for the operation
EError_UnderdeterminedMatrix	Unsupported operation: The matrix has less rows than columns
EError_OverdeterminedMatrix	Unsupported operation: The matrix has more rows than columns
EError_PointAtInfinity	Unable to apply this operation to points at infinity
EError_NotEnoughCalibrationPoints	Not enough points for the calibration process to succeed
EError_LineAtInfinity	Unable to apply this operation to lines at infinity
EError_UndeterminedGeometricEntity	Undetermined geometric entity in projective geometry
EError_NotANumber	Not a number
EError_MetadataAlreadyExists	Metadata already exists in the metadata store
EError_MetadataDoesNotExist	Metadata does not exist in the metadata store
EError_NotUTF8Compatible	The source is not UTF-8 compatible.
EError_InvalidTrainingMode	The chosen training mode is invalid.
EError_InvalidState	"The object is not in the correct state."
EError_FiducialNotFound	"One of the fiducials has not been found."
EError_InvalidModelFile	"Invalid Model File."
EError_InternalError_000	Internal error 0
EError_InternalError_001	Internal error 1
EError_InternalError_002	Internal error 2

EError_InternalError_003	Internal error 3
EError_InternalError_004	Internal error 4
EError_InternalError_005	Internal error 5
EError_InternalError_006	Internal error 6
EError_InternalError_007	Internal error 7
EError_InternalError_008	Internal error 8
EError_InternalError_009	Internal error 9
EError_InternalError_010	Internal error 10
EError_InternalError_011	Internal error 11
EError_InternalError_012	Internal error 12
EError_InternalError_013	Internal error 13
EError_InternalError_014	Internal error 14
EError_InternalError_015	Internal error 15
EError_InternalError_016	Internal error 16
EError_InternalError_017	Internal error 17
EError_InternalError_018	Internal error 18
EError_InternalError_019	Internal error 19
EError_InternalError_020	Internal error 20
EError_InternalError_021	Internal error 21
EError_InternalError_022	Internal error 22
EError_InternalError_023	Internal error 23

EError_InternalError_024	Internal error 24
EError_InternalError_025	Internal error 25
EError_InternalError_026	Internal error 26
EError_InternalError_027	Internal error 27
EError_InternalError_028	Internal error 28
EError_InternalError_029	Internal error 29
EError_InternalError_030	Internal error 30
EError_InternalError_031	Internal error 31
EError_InternalError_032	Internal error 32
EError_InternalError_033	Internal error 33
EError_InternalError_034	Internal error 34
EError_InternalError_035	Internal error 35
EError_InternalError_036	Internal error 36
EError_InternalError_037	Internal error 37
EError_InternalError_038	Internal error 38
EError_InternalError_039	Internal error 39
EError_InternalError_040	Internal error 40
EError_InternalError_041	Internal error 41
EError_InternalError_042	Internal error 42
EError_InternalError_043	Internal error 43
EError_InternalError_044	Internal error 44

EError_InternalError_045	Internal error 45
EError_InternalError_046	Internal error 46
EError_InternalError_047	Internal error 47
EError_InternalError_048	Internal error 48
EError_InternalError_049	Internal error 49
EError_InternalError_050	Internal error 50
EError_InternalError_051	Internal error 51
EError_InternalError_052	Internal error 52
EError_InternalError_053	Internal error 53
EError_InternalError_054	Internal error 54
EError_InternalError_055	Internal error 55
EError_InternalError_056	Internal error 56
EError_InternalError_057	Internal error 57
EError_InternalError_058	Internal error 58
EError_InternalError_059	Internal error 59
EError_InternalError_060	Internal error 60
EError_InternalError_061	Internal error 61
EError_InternalError_062	Internal error 62
EError_InternalError_063	Internal error 63
EError_InternalError_064	Internal error 64
EError_InternalError_065	Internal error 65

EError_InternalError_066	Internal error 66
EError_InternalError_067	Internal error 67
EError_InternalError_068	Internal error 68
EError_InternalError_069	Internal error 69
EError_InternalError_070	Internal error 70
EError_InternalError_071	Internal error 71
EError_InternalError_072	Internal error 72
EError_InternalError_073	Internal error 73
EError_InternalError_074	Internal error 74
EError_InternalError_075	Internal error 75
EError_InternalError_076	Internal error 76
EError_InternalError_077	Internal error 77
EError_InternalError_078	Internal error 78
EError_InternalError_079	Internal error 79
EError_InternalError_080	Internal error 80
EError_InternalError_081	Internal error 81
EError_InternalError_082	Internal error 82
EError_InternalError_083	Internal error 83
EError_InternalError_084	Internal error 84
EError_InternalError_085	Internal error 85
EError_InternalError_086	Internal error 86

EError_InternalError_087	Internal error 87
EError_InternalError_088	Internal error 88
EError_InternalError_089	Internal error 89
EError_InternalError_090	Internal error 90
EError_InternalError_091	Internal error 91
EError_InternalError_092	Internal error 92
EError_InternalError_093	Internal error 93
EError_InternalError_094	Internal error 94
EError_InternalError_095	Internal error 95
EError_InternalError_096	Internal error 96
EError_InternalError_097	Internal error 97
EError_InternalError_098	Internal error 98
EError_InternalError_099	Internal error 99
EError_InternalError_100	Internal error 100
EError_CannotTraceErrors	Cannot trace errors because of a system failure
EError_NotImplemented	Feature not implemented
EError_NullPointer	The supplied pointer is NULL
EError_InvalidTimeout	The current timeout value is 0
EError_InvalidTimeoutReentrancy	Cannot Stop a timeout that has not been started. Cannot Pop a timeout that has not been pushed
EError_InvalidTimeoutState	Cannot Start a timeout that has been reached Cannot Pop a timeout that is Active
EError_SharedLibraryLoadingError	Error during load of Open eVision shared library

EError_SharedLibraryFunctionLoadingError	Error during load of a function in the Open eVision shared library
EError_RegistryLookup	Error during load of Open eVision shared library
EError_LibraryHashFailed	Error indicating incompatibility between the current wrapper and the targeted library
EError_Unknown	Unknown error

6.53. EFamily Enum

This enum is deprecated.

Allowed values for the ECC symbol family in EasyMatrixCode.

Namespace: Euresys::Open_eVision

EFamily_ECC000	ECC 000, no error recovery capability by convolutional coding.
EFamily_ECC050	ECC 050, 2.8 % error recovery capability by convolutional coding.
EFamily_ECC080	ECC 080, 5.5 % error recovery capability by convolutional coding.
EFamily_ECC100	ECC 100, 12.6 % error recovery capability by convolutional coding.
EFamily_ECC140	ECC 140, 25 % error recovery capability by convolutional coding.
EFamily_ECC200	ECC 200, 20 % error recovery capability.
EFamily_Unknown	-

Remarks

This enumeration is in the Euresys::Open eVision namespace.

6.54. EFeature Enum

The various features that can be measured on the coded elements of a selection.

Namespace: Euresys::Open_eVision

EFeature_ElementIndex	Index of the coded element (cf. ECodedElement::ElementIndex).
EFeature_LayerIndex	Index of the layer of the coded element (cf. ECodedElement::LayerIndex).
EFeature_RunCount	Number of runs (cf. ECodedElement::RunCount).
EFeature_Area	Number of pixels (cf. ECodedElement::Area).
EFeature_LargestRun	Length of the largest run (cf. ECodedElement::LargestRun).

EFeature_ContourX	Starting point abscissa of the contour of the coded element (cf. ECodedElement::ContourX).
EFeature_ContourY	Starting point ordinate of the countour of the coded element (cf. ECodedElement::ContourY).
EFeature_LeftLimit	Abscissa of the leftmost pixel (cf. ECodedElement::LeftLimit).
EFeature_RightLimit	Abscissa of the rightmost pixel (cf. ECodedElement::RightLimit).
EFeature_TopLimit	Abscissa of the topmost pixel (cf. ECodedElement::TopLimit).
EFeature_BottomLimit	Ordinate of the bottommost pixel (cf. ECodedElement::BottomLimit).
EFeature_ GravityCenterX	Abscissa of the gravity center (cf. ECodedElement::GravityCenterX).
EFeature_ GravityCenterY	Ordinate of the gravity center (cf. ECodedElement::GravityCenterY).
EFeature_ BoundingBoxCenterX	Abscissa of the center of the bounding box (cf. ECodedElement::BoundingBoxCenterX).
EFeature_ BoundingBoxCenterY	Ordinate of the center of the bounding box (cf. ECodedElement::BoundingBoxCenterY).
EFeature_ BoundingBoxWidth	Width of the bounding box (Feret diameter 0 degrees - cf. ECodedElement::BoundingBoxWidth).
EFeature_ BoundingBoxHeight	Height of the bounding box (Feret diameter 90 degrees - cf. ECodedElement::BoundingBoxHeight).
EFeature_ FeretBox22CenterX	Abscissa of the center of the Feret box oriented at 22.5 degrees (cf. ECodedElement::FeretBox22CenterX).
EFeature_ FeretBox22CenterY	Ordinate of the center of the Feret box oriented at 22.5 degrees (cf. ECodedElement::FeretBox22CenterY).
EFeature_ FeretBox22Width	Width of the Feret box oriented at 22.5 degrees (Feret diameter at 22.5 degrees - cf. ECodedElement::FeretBox22Width).
EFeature_ FeretBox22Height	Height of the Feret box oriented at 22.5 degrees (Feret diameter at 112.5 degrees - cf. ECodedElement::FeretBox22Height).
EFeature_ FeretBox45CenterX	Abscissa of the center of the Feret box oriented at 45 degrees (cf. ECodedElement::FeretBox45CenterX).
EFeature_ FeretBox45CenterY	Ordinate of the center of the Feret box oriented at 45 degrees (cf. ECodedElement::FeretBox45CenterY).
EFeature_ FeretBox45Width	Width of the Feret box oriented at 45 degrees bounding box (Feret diameter at 45 degrees - cf. ECodedElement::FeretBox45Width).
EFeature_ FeretBox45Height	Height of the Feret box oriented at 45 degrees (Feret diameter at 135 degrees - cf. ECodedElement::FeretBox45Height).
EFeature_ FeretBox68CenterX	Abscissa of the center of the Feret box oriented at 67.5 degrees (cf. ECodedElement::FeretBox68CenterX).
EFeature_ FeretBox68CenterY	Ordinate of the center of the Feret box oriented at 67.5 degrees (cf. ECodedElement::FeretBox68CenterY).

EFeature_FeretBox68Width	Width of the Feret box oriented at 67.5 degrees (Feret diameter at 67.5 degrees - cf. ECodedElement::FeretBox68Width).
EFeature_FeretBox68Height	Height of the Feret box oriented at 67.5 degrees (Feret diameter at 157.5 degrees - cf. ECodedElement::FeretBox68Height).
EFeature_MinimumEnclosingRectangleCenterX	Abscissa of the Minimum Enclosing Rectangle center (cf. ECodedElement::MinimumEnclosingRectangleCenterX).
EFeature_MinimumEnclosingRectangleCenterY	Ordinate of the Minimum Enclosing Rectangle center (cf. ECodedElement::MinimumEnclosingRectangleCenterY).
EFeature_MinimumEnclosingRectangleWidth	Width of the Minimum Enclosing Rectangle (cf. ECodedElement::MinimumEnclosingRectangleWidth).
EFeature_MinimumEnclosingRectangleHeight	Height of the Minimum Enclosing Rectangle (cf. ECodedElement::MinimumEnclosingRectangleHeight).
EFeature_MinimumEnclosingRectangleAngle	Direction of the Minimum Enclosing Rectangle (cf. ECodedElement::MinimumEnclosingRectangleAngle).
EFeature_SigmaX	Centered moment of inertia around X (average squared X-deviation - cf. ECodedElement::SigmaX).
EFeature_SigmaY	Centered moment of inertia around Y (average squared Y-deviation - cf. ECodedElement::SigmaY).
EFeature_SigmaXX	Reduced, centered moment of inertia (around the principal inertia axis - cf. ECodedElement::SigmaXX).
EFeature_SigmaXY	Centered cross moment of inertia (average X-deviation * Y-deviation - cf. ECodedElement::SigmaXY).
EFeature_SigmaYY	Reduced, centered moment of inertia (around the secondary inertia axis - cf. ECodedElement::SigmaYY).
EFeature_EllipseWidth	Long axis of the ellipse of inertia (cf. ECodedElement::EllipseWidth).
EFeature_EllipseHeight	Short axis of the ellipse of inertia (cf. ECodedElement::EllipseHeight).
EFeature_EllipseAngle	Angle of the principal axis of the ellipse of inertia (cf. ECodedElement::EllipseAngle).
EFeature_Eccentricity	Eccentricity of the ellipse of inertia (cf. ECodedElement::Eccentricity).
EFeature_FeretBoxCenterX	Abscissa of the center of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle.
EFeature_FeretBoxCenterY	Ordinate of the center of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle.
EFeature_FeretBoxWidth	Width of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle.

EFeature_FeretBoxHeight	Height of the Feret box oriented at a fixed angle (cf. ECodedElement::ComputeFeretBox). The angle of interest is set through EObjectSelection::FeretAngle.
EFeature_PixelMin	Minimum gray level of the pixels of the attached image over the coded element (cf. ECodedElement::ComputePixelMin). The attached image is set through EObjectSelection::AttachedImage.
EFeature_PixelMax	Maximum gray level of the pixels of the attached image over the coded element (cf. ECodedElement::ComputePixelMax). The attached image is set through EObjectSelection::AttachedImage.
EFeature_WeightedGravityCenterX	Abscissa of the gravity center of the pixels of the attached image over the coded element (cf. ECodedElement::ComputeWeightedGravityCenter). The attached image is set through EObjectSelection::AttachedImage.
EFeature_WeightedGravityCenterY	Ordinate of the gravity center of the pixels of the attached image over the coded element (cf. ECodedElement::ComputeWeightedGravityCenter). The attached image is set through EObjectSelection::AttachedImage.
EFeature_PixelGrayAverage	Average gray-level value of the attached image over the coded element (cf. ECodedElement::ComputePixelGrayAverage). The attached image is set through EObjectSelection::AttachedImage.
EFeature_PixelGrayVariance	Variance of the gray-level value of the attached image over the coded element (cf. ECodedElement::ComputePixelGrayVariance). The attached image is set through EObjectSelection::AttachedImage.
EFeature_PixelGrayDeviation	Standard deviation of the gray-level value of the attached image over the coded element (cf. ECodedElement::ComputePixelGrayDeviation). The attached image is set through EObjectSelection::AttachedImage.

6.55. EFiducialMatchingMode Enum

Allowed values for the fiducial finder mode in EChecker.

Namespace: Euresys::Open_eVision

EFiducialMatchingMode_Geometric	Geometric finder. Use this mode if the fiducials are well defined and/or may be subject to occlusion.
EFiducialMatchingMode_Area	Area finder. Use this mode if the fiducials are not well defined (have poor edges) or are of inconsistent size.

6.56. EFillUndefinedPixelsDirection Enum

Direction in which the undefined pixels are filled in a depthmap.

Namespace: Euresys::Open_eVision::Easy3D

EFillUndefinedPixelsDirection_Vertical	Undefined pixels are filled using a vertical scan.
EFillUndefinedPixelsDirection_Horizontal	Undefined pixels are filled using an horizontal scan.
EFillUndefinedPixelsDirection_Combined	Undefined pixels are filled using both a vertical and an horizontal scan.
EFillUndefinedPixelsDirection_Local	Specialized method for filling undefined pixels. Undefined pixels are filled using their 4 neighboring pixels: If at least 2 out of 4 neighbors have a 'defined' value, the pixel will be filled with their average value. Else, the pixel will remain 'undefined'.

6.57. EFillUndefinedPixelsMethod Enum

Method to fill the undefined pixels in a depthmap.

Namespace: Euresys::Open_eVision::Easy3D

EFillUndefinedPixelsMethod_KeepMinimum	Undefined pixels are filled using the minimum of their neighbors.
EFillUndefinedPixelsMethod_KeepMaximum	Undefined pixels are filled using the maximum of their neighbors.
EFillUndefinedPixelsMethod_Average	Undefined pixels are filled using the average of their neighbors.
EFillUndefinedPixelsMethod_Ramp	Undefined pixels are filled using a ramp between their neighbors.

6.58. EFilteringMode Enum

Allowed values for the filtering mode of EasyMatch.

Namespace: Euresys::Open_eVision

EFilteringMode_Uniform	Filtering with a uniform 2x2 kernel. This is the preferred mode for natural images. Default mode.
EFilteringMode_LowPass	Filtering with a low-pass 3x3 kernel. This is the preferred mode for images featuring sharp gray-level transitions.

6.59. EFindContrastMode Enum

Allowed values for the contrast mode of EasyFind.

Namespace: Euresys::Open_eVision

EFindContrastMode_Normal	Accepts instances with normal contrast (default mode).
EFindContrastMode_Inverse	Accepts instances with reversed contrast.
EFindContrastMode_Any	Accepts instances with normal and/or reversed contrast.
EFindContrastMode_PointByPointNormal	Accepts instances with normal contrast. Computes pattern score based on a point by point score.
EFindContrastMode_PointByPointInverse	Accepts instances with reversed contrast. Computes pattern score based on a point by point score.
EFindContrastMode_PointByPointAny	Accepts instances with normal and/or reversed contrast. Computes pattern score based on a point by point score.
EFindContrastMode_Unknown	-

6.60. EFlipAxis Enum

Axis for flipping

Namespace: Euresys::Open_eVision

EFlip_Horinzontal_Axis	-
EFlip_Vertical_Axis	-
EFlip_Both_Axis	-

6.61. EFlipping Enum

This enum is deprecated.

Allowed values for the symbol flipping type in EasyMatrixCode.

Namespace: Euresys::Open_eVision

EFlipping_Yes	Image is flipped.
EFlipping_No	Image is not flipped.
EFlipping_Unknown	To be determined at Read or Learn time.

6.62. EFontStyle Enum

Font style.

Namespace: Euresys::Open_eVision

EFontStyle_Regular	-
EFontStyle_Bold	-
EFontStyle_Italic	-
EFontStyle_BoldItalic	-
EFontStyle_Underline	-
EFontStyle_Strikeout	-

Remarks

The values EFontStyle_Underline and EFontStyle_Strikeout are deprecated because they are not usable with the EGenericDrawAdapter.

6.63. EFramePosition Enum

This enumeration contains the possible values for the placement of the overlay frame edges that are drawn to highlight the position of an ROI.

Namespace: Euresys::Open_eVision

EFramePosition_On	The frame is centered on the ROI edges.
EFramePosition_Inside	The outer edges of the frame remain totally inside the ROI.
EFramePosition_ Outside	The inner edges of the frame remain totally outside the ROI.

6.64. EFrequentialDomainFormat Enum

This enumeration represents the supported data formats for frequential domain images. See EFourierTransformer.

Namespace: Euresys::Open_eVision

EFrequentialDomainFormat_Packed	Packed format, also called Complex Conjugate Symmetrical. This format takes benefit of the Symmetrical nature of the frequential domain to reduce output size to be the same as input size.
EFrequentialDomainFormat_ ComplexExtended	Extended Complex Format. The output image width is twice the input image width (height is the same). Each Odd Column represents the imaginary part of a complex number.

6.65. EGrayscaleSingleThreshold Enum

The modes that are available to segment a grayscale image using a single threshold.

Namespace: Euresys::Open_eVision

EGrayscaleSingleThreshold_Absolute	Thresholds the image against a fixed, absolute gray level. The threshold value is fixed through EGrayscaleSingleThresholdSegmenter::AbsoluteThreshold.
EGrayscaleSingleThreshold_Relative	Thresholds the image against a relative gray level: The actual threshold is selected so that a given fraction of the pixels of the image lie below it. The fraction is fixed through EGrayscaleSingleThresholdSegmenter::RelativeThreshold.
EGrayscaleSingleThreshold_MinResidue	Thresholds the image using an automatically-computed value such that the quadratic difference between the source and thresholded image is minimized.
EGrayscaleSingleThreshold_MaxEntropy	Thresholds the image using an automatically-computed value such that the entropy (i.e. the amount of information) of the resulting thresholded image is maximized.
EGrayscaleSingleThreshold_IsoData	Thresholds the image using an automatically-computed value halfway between the average dark gray value (i.e. gray levels below the threshold) and average light gray values (i.e. gray levels above the threshold).

6.66. EHarrisThresholdingMode Enum

The thresholding modes for the Harris corner detector.

Namespace: Euresys::Open_eVision

EHarrisThresholdingMode_Relative	Relative thresholding mode.
EHarrisThresholdingMode_Absolute	Absolute thresholding mode.

6.67. EHeatmapColormap Enum

Color maps for visualizing the heatmap.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EHeatmapColormap_YellowToRed	Colormap showing a gradient from yellow to red.

EHeatmapColormap_Jet	Colormap showing a gradient going from blue to gree to yellow to dark red.

6.68. EHistogramFeature Enum

The various parameters that can be extracted from a histogram.

Namespace: Euresys::Open_eVision

EHistogramFeature_MostFrequentPixelValue	Value of the most frequent pixel.
EHistogramFeature_MostFrequentPixelFrequency	Frequency of the most frequent pixel.
EHistogramFeature_LeastFrequentPixelValue	Value of the least frequent pixel.
EHistogramFeature_LeastFrequentPixelFrequency	Frequency of the least frequent pixel.
EHistogramFeature_SmallestPixelValue	Smallest pixel value.
EHistogramFeature_GreatestPixelValue	Largest pixel value.
EHistogramFeature_PixelCount	Number of pixels.
EHistogramFeature_AveragePixelValue	Mean of the pixel values.
EHistogramFeature_PixelValueStdDev	Standard deviation of the pixel values.

6.69. EHitAndMissValue Enum

The allowed values for the elements of a hit-and-miss kernel.

Namespace: Euresys::Open_eVision

EHitAndMissValue_Background	The element belongs to the background.
EHitAndMissValue_DontCare	The element does not belongs to the background, neither to the foreground. It is ignored by the kernel.

EHitAndMissValue_Foreground	The element belongs to the foreground.

6.70. EImageAnnotationFormat Enum

Supported formats of annotation files for image file.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EImageAnnotationFormat_None	No annotation.
EImageAnnotationFormat_YOLOTXT	The YOLO annotation format for EasyLocate bounding box. Annotation file: imagefilename.txt. Description of the format: each row of the file defines an object: class_id center_x center_y width height
EImageAnnotationFormat_PASCALVOCXML	Pascal VOC XML format for EasyLocate bounding box. Annotation file: imagefilename.xml. Description of the format: XML file. See the official Pascal VOC website for a description of the format (http://host.robots.ox.ac.uk/pascal/VOC/).

6.71. EImageFileType Enum

-

Namespace: Euresys::Open_eVision

EImageFileType_Bmp	-
EImageFileType_Jpeg2000	-
EImageFileType_Jpeg	-
EImageFileType_Png	-
EImageFileType_Tiff	-
EImageFileType_Auto	-
EImageFileType_Euresys	-
EImageFileType_Unknown	-

6.72. EImageType Enum

Image type.

Namespace: Euresys::Open_eVision

EImageType_BW1	Bi-level image.
EImageType_BW8	8 bits per pixel gray-level image.
EImageType_BW16	16 bits per pixel gray-level image
EImageType_BW32	32 bits per pixel gray-level image.
EImageType_C15	15 bits per pixel color image (R:5, G:5, B:5).
EImageType_C16	16 bits per pixel color image (R:5, G:6, B:5).
EImageType_C24	24 bits per pixel color image (RGB).
EImageType_C24A	32 bits per pixel color image (RGB + unused alpha channel).
EImageType_C48	48 bits per pixel color image (RGB).
EImageType_Depth8	8 bits per pixel gray-level image.
EImageType_Depth16	16 bits per pixel gray-level image.
EImageType_Depth32f	32 bits per pixel floating-point gray-level image.
EImageType_BW32f	32 bits per pixel floating-point gray-level image.

Remarks

For example, an EImageC24 has type value EImageType_C24 and its pixels are typed as EC24.

6.73. EKernelRectifier Enum

Possible values for the rectification mode of a kernel. This property allows specifying how negative convolution result values are handled.

Namespace: Euresys::Open_eVision

EKernelRectifier_DoNotRectify	The offset of the kernel is added to the values resulting from the convolution. Negative values are then set to zero, and the values that exceed the maximum value for the image type are set to this maximum value.
EKernelRectifier_KeepNegative	The positive values are discarded (set to zero) and the magnitude (absolute value) of the negative values is used.
EKernelRectifier_KeepPositive	Positive value is used. The negative values are discarded (set to zero).
EKernelRectifier_Absolute	The absolute value is used.

6.74. EKernelRotation Enum

Possible values for rotating a convolution kernel.

Namespace: Euresys::Open_eVision

EKernelRotation_NoRotation	No rotation of the structuring element.
EKernelRotation_Clockwise	Clockwise rotation (one full turn per pass).
EKernelRotation_Anticlockwise	Counterclockwise rotation (one full turn per pass).

6.75. EKernelType Enum

The types of convolution kernels that are supported by Open eVision.

Namespace: Euresys::Open_eVision

EKernelType_WhiteSkelet	White skeleton morphological probe.
EKernelType_BlackSkelet	Black skeleton morphological probe.
EKernelType_Edge	Edge detection morphological probe.
EKernelType_SobelX	X-axis Sobel derivative.
EKernelType_SobelY	Y-axis Sobel derivative.
EKernelType_PrewittX	X-axis Prewitt derivative.
EKernelType_PrewittY	Y-axis Prewitt derivative.
EKernelType_Laplacian4	4-connected Laplacian.
EKernelType_Laplacian8	8-connected Laplacian.
EKernelType_LowPass1	Low pass filter.
EKernelType_LowPass2	Low pass filter (average of neighbors).
EKernelType_LowPass3	Low pass filter (average).
EKernelType_HighPass1	High pass filter (value plus 4-connected Laplacian).
EKernelType_HighPass2	High pass filter (value plus 8-connected Laplacian).
EKernelType_Sobel	-
EKernelType_Prewitt	-

EKernelType_Roberts	-
EKernelType_ Uniform3x3	-
EKernelType_ Gaussian3x3	-
EKernelType_ Uniform5x5	-
EKernelType_ Gaussian5x5	-
EKernelType_ Gaussian7x7	-
EKernelType_ Uniform7x7	-
EKernelType_ LaplacianX	-
EKernelType_ LaplacianY	-
EKernelType_Gradient	-
EKernelType_GradientX	-
EKernelType_GradientY	-
EKernelType_Uniform	-
EKernelType_Gaussian	-

6.76. ELearnParam Enum

This enum is deprecated.

Allowed values for the kind of parameters that can be learnt by EasyMatrixCode.

Namespace: Euresys::Open_eVision

ELearnParam_ LogicalSize	The data matrix code symbol logical sizes the candidate is matched against at read time.
ELearnParam_ ContrastType	The data matrix code contrast types the candidate is matched against at read time.
ELearnParam_Flipping	The data matrix code flipping values the candidate is matched against at read time.
ELearnParam_Family	The data matrix code families the candidate is matched against at read time.
ELearnParam_ NumItems	-

6.77. ELegacyFeature Enum

The various parameters that can be extracted from a histogram. This enumeration pertains to the EasyObject legacy API. Please use ECodedImage2 instead.

Namespace: Euresys::Open_eVision

ELegacyFeature_NoFeature	-
ELegacyFeature_Class	Class number.
ELegacyFeature_RunsNumber	Number of runs.
ELegacyFeature_Area	Number of pixels. *(Signed Integer)*.
ELegacyFeature_LargestRun	Size of the longest run. *(Signed Integer)*.
ELegacyFeature_GravityCenterX	Abscissa of the gravity center. *(Float)*. (*)
ELegacyFeature_GravityCenterY	Ordinate of the gravity center. *(Float)*. (*)
ELegacyFeature_LimitCenterX	Abscissa of the center of the bounding box. *(Float)*. (*)
ELegacyFeature_LimitCenterY	Ordinate of the center of the bounding box. *(Float)*. (*)
ELegacyFeature_LimitWidth	Width of the bounding box (Feret's diameter 0 degrees). *(Float)*. (*)
ELegacyFeature_LimitHeight	Height of the bounding box (Feret's diameter 90 degrees). *(Float)*. (*)
ELegacyFeature_Limit45CenterX	Abscissa of the center of the 45 degrees bounding box. *(Float)*. (*)
ELegacyFeature_Limit45CenterY	Ordinate of the center of the 45 degrees bounding box. *(Float)*. (*)
ELegacyFeature_Limit45Width	Width of the 45 degrees bounding box (Feret's diameter 45 degrees). *(Float)*. (*)
ELegacyFeature_Limit45Height	Height of the 45 degrees bounding box (Feret's diameter 135 degrees). *(Float)*. (*)
ELegacyFeature_ContourX	Starting point abscissa of the object contour. *(Signed Integer)*.
ELegacyFeature_ContourY	Starting point ordinate of the object contour. *(Signed Integer)*.
ELegacyFeature_PixelMin	Minimum gray level of all pixels. *(Signed Integer)*.

ELegacyFeature_PixelMax	Maximum gray level of all pixels. *(Signed Integer)*.
ELegacyFeature_SigmaX	Centered moment of inertia around X (average squared X-deviation). *(Float)*.
ELegacyFeature_SigmaY	Centered moment of inertia around Y (average squared Y-deviation). *(Float)*.
ELegacyFeature_SigmaXY	Centered cross moment of inertia (average X-deviation * Y-deviation). *(Float)*.
ELegacyFeature_SigmaXX	Reduced, centered moment of inertia (around the principal inertia axis). *(Float)*.
ELegacyFeature_SigmaYY	Reduced, centered moment of inertia (around the secondary inertia axis). *(Float)*.
ELegacyFeature_EllipseWidth	Long axis of the ellipse of inertia. *(Float)*. (*)
ELegacyFeature_EllipseHeight	Short axis of the ellipse of inertia. *(Float)*. (*)
ELegacyFeature_EllipseAngle	Direction of the principal axis of inertia. *(Float)*. (*)
ELegacyFeature_CentroidX	Abscissa of the weighted gravity center. *(Float)*. (*)
ELegacyFeature_CentroidY	Ordinate of the weighted gravity center. *(Float)*. (*)
ELegacyFeature_PixelGrayAverage	Average gray-level value of the object pixels. *(Float)*.
ELegacyFeature_PixelGrayVariance	Variance of the gray-level value of the object pixels. *(Float)*.
ELegacyFeature_Limit22CenterX	Abscissa of the center of the 22.5 degrees bounding box. *(Float)*. (*)
ELegacyFeature_Limit22CenterY	Ordinate of the center of the 22.5 degrees bounding box. *(Float)*. (*)
ELegacyFeature_Limit22Width	Width of the 22.5 degrees bounding box (Feret's diameter 22.5 degrees). *(Float)*. (*)
ELegacyFeature_Limit22Height	Height the 22.5 degrees bounding box (Feret's diameter 112.5 degrees). *(Float)*. (*)
ELegacyFeature_Limit68CenterX	Abscissa of the center of the 67.5 degrees bounding box. *(Float)*. (*)
ELegacyFeature_Limit68CenterY	Ordinate of the center of the 67.5 degrees bounding box. *(Float)*. (*)
ELegacyFeature_Limit68Width	Width of the 67.5 degrees bounding box (Feret's diameter 67.5 degrees). *(Float)*. (*)
ELegacyFeature_Limit68Height	Height of the 67.5 degrees bounding box (Feret's diameter 157.5 degrees). *(Float)*. (*)

ELegacyFeature_ LimitAngledCenterX	Abscissa of the center of the bounding box having a skew angle defined by the LimitAngle property. *(Float)*.
ELegacyFeature_ LimitAngledCenterY	Ordinate of the center of the bounding box having a skew angle defined by the LimitAngle property. *(Float)*.
ELegacyFeature_ LimitAngledWidth	Width of the bounding box having a skew angle defined by the LimitAngle property (Feret's diameter [LimitAngle]). *(Float)*.
ELegacyFeature_ LimitAngledHeight	Height of the bounding box having a skew angle defined by the LimitAngle property (Feret's diameter [LimitAngle + 90 degrees]). *(Float)*.
ELegacyFeature_ FeretCenterX	Abscissa of the Feret's bounding box center. *(Float)*. (*)
ELegacyFeature_ FeretCenterY	Ordinate of the Feret's bounding box center. *(Float)*. (*)
ELegacyFeature_ FeretWidth	Width of the Feret's bounding box. *(Float)*. (*)
ELegacyFeature_ FeretHeight	Height of the Feret's bounding box. *(Float)*. (*)
ELegacyFeature_ FeretAngle	Direction of the Feret's bounding box. *(Float)*. (*)
ELegacyFeature_ ObjectNumber	Identification number.
ELegacyFeature_ GravityCenter	Abscissa of the gravity center. *(Float)*. (*)
ELegacyFeature_Limit	Abscissa of the center of the bounding box. *(Float)*. (*)
ELegacyFeature_ Limit22	Abscissa of the center of the 22.5 degrees bounding box. *(Float)*. (*)
ELegacyFeature_ Limit45	Abscissa of the center of the 45 degrees bounding box. *(Float)*. (*)
ELegacyFeature_ Limit68	Abscissa of the center of the 67.5 degrees bounding box. *(Float)*. (*)
ELegacyFeature_ LimitAngled	Abscissa of the center of the bounding box having a skew angle defined by the LimitAngle property. *(Float)*.
ELegacyFeature_Ellipse	Long axis of the ellipse of inertia. *(Float)*. (*)
ELegacyFeature_ Centroid	-
ELegacyFeature_Feret	-

6.78. ELineSpacingMode Enum

Allowed values for the line spacing mode of EasyOCR.

Namespace: Euresys::Open_eVision

ELineSpacingMode_ Overlap	Two characters will belong to two different lines when the difference between the bottom of their boxes is larger than 30% of EOCR::MaxCharHeight.
ELineSpacingMode_ Normal	Two characters will belong to two different lines when the bottom of one character box is higher than the top of the other.

6.79. ELocalSearchMode Enum

Allowed values for the local search mode of EasyFind.

Namespace: Euresys::Open_eVision

ELocalSearchMode_ Basic	Default local search neighborhood. Sets EPatternFinder::AngleSearchExtent, EPatternFinder::ScaleSearchExtent, EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 3.
ELocalSearchMode_ ExtendedTranslation	Local search neighborhood extended on the translation degrees of freedom. Sets EPatternFinder::AngleSearchExtent and EPatternFinder::ScaleSearchExtent to 3. Sets EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 5.
ELocalSearchMode_ ExtendedAll	Local search neighborhood extended on all degrees of freedom. Sets EPatternFinder::AngleSearchExtent, EPatternFinder::ScaleSearchExtent, EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 5.
ELocalSearchMode_ ExtendedMore	Local search neighborhood even more extended on all degrees of freedom. Sets EPatternFinder::AngleSearchExtent and EPatternFinder::ScaleSearchExtent to 7. Sets EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to 9.
ELocalSearchMode_ Reserved	Reserved for internal use.
ELocalSearchMode_ Custom	Custom local search neighborhood. Set EPatternFinder::AngleSearchExtent, EPatternFinder::ScaleSearchExtent, EPatternFinder::XSearchExtent and EPatternFinder::YSearchExtent to custom values.

6.80. ELocatorCapacity Enum

The capacity of the locator deep learning network.
A larger capacity means that the underlying neural network is capable of learning more information but it will be slower.

Namespace: Euresys::Open_eVision::EasyDeepLearning

ELocatorCapacity_Small	Small capacity.
ELocatorCapacity_Normal	Normal capacity.
ELocatorCapacity_Large	Large capacity.

6.81. ELocatorFeature Enum

Features supported by the ELocator or ELocatorObject.

Namespace: Euresys::Open_eVision::EasyDeepLearning

ELocatorFeature_None	No features.
ELocatorFeature_Position	The locator predicts the position of the object.
ELocatorFeature_Size	The locator predicts the size of the object (width and height).

6.82. ELogicalSize Enum

This enum is deprecated.

Allowed values for the logical size of Data Matrix codes in EasyMatrixCode.

Namespace: Euresys::Open_eVision

ELogicalSize__9x9	ECC 000-140 squares.
ELogicalSize__11x11	ECC 000-140 squares.
ELogicalSize__13x13	ECC 000-140 squares.
ELogicalSize__15x15	ECC 000-140 squares.
ELogicalSize__17x17	ECC 000-140 squares.
ELogicalSize__19x19	ECC 000-140 squares.
ELogicalSize__21x21	ECC 000-140 squares.
ELogicalSize__23x23	ECC 000-140 squares.
ELogicalSize__25x25	ECC 000-140 squares.
ELogicalSize__27x27	ECC 000-140 squares.
ELogicalSize__29x29	ECC 000-140 squares.
ELogicalSize__31x31	ECC 000-140 squares.
ELogicalSize__33x33	ECC 000-140 squares.

ELogicalSize__35x35	ECC 000-140 squares.
ELogicalSize__37x37	ECC 000-140 squares.
ELogicalSize__39x39	ECC 000-140 squares.
ELogicalSize__41x41	ECC 000-140 squares.
ELogicalSize__43x43	ECC 000-140 squares.
ELogicalSize__45x45	ECC 000-140 squares.
ELogicalSize__47x47	ECC 000-140 squares.
ELogicalSize__49x49	ECC 000-140 squares.
ELogicalSize__10x10	ECC 200 squares.
ELogicalSize__12x12	ECC 200 squares.
ELogicalSize__14x14	ECC 200 squares.
ELogicalSize__16x16	ECC 200 squares.
ELogicalSize__18x18	ECC 200 squares.
ELogicalSize__20x20	ECC 200 squares.
ELogicalSize__22x22	ECC 200 squares.
ELogicalSize__24x24	ECC 200 squares.
ELogicalSize__26x26	ECC 200 squares.
ELogicalSize__32x32	ECC 200 squares.
ELogicalSize__36x36	ECC 200 squares.
ELogicalSize__40x40	ECC 200 squares.
ELogicalSize__44x44	ECC 200 squares.
ELogicalSize__48x48	ECC 200 squares.
ELogicalSize__52x52	ECC 200 squares.
ELogicalSize__64x64	ECC 200 squares.
ELogicalSize__72x72	ECC 200 squares.
ELogicalSize__80x80	ECC 200 squares.
ELogicalSize__88x88	ECC 200 squares.
ELogicalSize__96x96	ECC 200 squares.
ELogicalSize__104x104	ECC 200 squares.
ELogicalSize__120x120	ECC 200 squares.
ELogicalSize__132x132	ECC 200 squares.
ELogicalSize__144x144	ECC 200 squares.
ELogicalSize__8x18	ECC 200 rectangles
ELogicalSize__8x32	ECC 200 rectangles
ELogicalSize__12x26	ECC 200 rectangles

ELogicalSize__12x36	ECC 200 rectangles
ELogicalSize__16x36	ECC 200 rectangles
ELogicalSize__16x48	ECC 200 rectangles
ELogicalSize_Unknown	To be determined at Read or Learn time.

Remarks

This enumeration is in the Euresys::Open eVision namespace.

6.83. EMailBarcodeOrientation Enum

The orientations supported by EMailBarcode

Namespace: Euresys::Open_eVision

EMailBarcodeOrientation_Unknown	Unknown orientation.
EMailBarcodeOrientation_NoRotation	Non rotated barcode. Horizontal and read from left to right.
EMailBarcodeOrientation_Rotated180	Upside-down barcode. Horizontal and read from right to left.
EMailBarcodeOrientation_Rotated90Right	Barcode rotated 90 degrees to the right. Vertical and read from top to bottom.
EMailBarcodeOrientation_Rotated90Left	Barcode rotated 90 degrees to the left. Vertical and read from bottom to top.
EMailBarcodeOrientation_Horizontal	Barcode is horizontal.
EMailBarcodeOrientation_Vertical	Barcode is vertical.
EMailBarcodeOrientation_Any	Barcode has any one of the supported orientations.

6.84. EMailBarcodeSymbologies Enum

The symbologies supported by EMailBarcode

Namespace: Euresys::Open_eVision

EMailBarcodeSymbologies_Unknown	Unknown symbology.
EMailBarcodeSymbologies_JapanPost	Japan Post symbology.

EMailBarcodeSymbologies_Postnet	US POSTNET symbology.
EMailBarcodeSymbologies_Planet	US PLANET symbology.
EMailBarcodeSymbologies_IntelligentMail	US Intelligent Mail symbology.
EMailBarcodeSymbologies_USMail	US symbologies.

6.85. EMapConversionMode Enum

Conversion modes to use in EConverter.

Namespace: Euresys::Open_eVision::Easy3D

EMapConversionMode_MaxDynamic	Use the mean as threshold.
EMapConversionMode_Shift	Use the mean as threshold.

6.86. EMapConversionMode Enum

Conversion modes to use in EConverter.

Namespace: Euresys::Open_eVision::Easy3D

EMapConversionMode_MaxDynamic	Use the mean as threshold.
EMapConversionMode_Shift	Use the mean as threshold.

6.87. EMatchContrastMode Enum

Allowed values for the contrast mode of EasyMatch.

Namespace: Euresys::Open_eVision

EMatchContrastMode_Normal	Normal contrast. Pattern occurrences will be found with the same contrast as at learn time. Default mode.
EMatchContrastMode_Inverse	Inverse contrast. Pattern occurrences will be found with reversed contrast.

EMatchContrastMode_Any	Normal or inverse contrast. Pattern occurrences can be found both with normal and inverse contrast.

6.88. EMatchingMode Enum

Allowed values for the matching mode of EasyOCR.

Namespace: Euresys::Open_eVision

EMatchingMode_Rms	Root-mean-square error method is used.
EMatchingMode_Standard	Gray-level correlation method is used.
EMatchingMode_Normalized	Normalized gray-level correlation method is used.

6.89. EMatrixCodeContrastMode Enum

This enum is deprecated.

-

Namespace: Euresys::Open_eVision

EMatrixCodeContrastMode_BlackOnWhite	Dark cells on a light background.
EMatrixCodeContrastMode_WhiteOnBlack	Light cells on a dark background.

6.90. EMaximumAnalysisMode Enum

This enumeration contains the possible values for the analysis mode of the ELaserLineExtractor object.

Namespace: Euresys::Open_eVision::Easy3D

EMaximumAnalysisMode_Peaks	Peak analysis mode.
EMaximumAnalysisMode_Max	Maximum analysis mode.
EMaximumAnalysisMode_COG	Center of gravity analysis mode.

6.91. ENoiseRemovalMethod Enum

Type of filter used in method EFilters::RemoveNoise.

Namespace: Euresys::Open_eVision::Easy3D

ENoiseRemovalMethod_AbsoluteDifferenceFromMean	Removes points for which the deviation from the average height in the filter window is larger than the specified threshold. The threshold is the absolute value of the maximum difference.
ENoiseRemovalMethod_RelativeDifferenceFromMean	Removes points for which the deviation from the average height in the filter window is larger than the specified threshold multiplied by the standard deviation (in the same filter window). The threshold is a factor.
ENoiseRemovalMethod_HighStandardDeviation	Removes points for which the standard deviation calculated in the filter window is larger than a specified threshold. The threshold is the maximum standard deviation.

6.92. ENormalizationMode Enum

Allowed values for the gray-level normalization mode in EChecker2 and EasyOCV.

Namespace: Euresys::Open_eVision

ENormalizationMode_NoNormalization	No gray-level normalization (contrast changes will be detected).
ENormalizationMode_Moments	Contrast normalization based on linear change.
ENormalizationMode_Threshold	Contrast normalization based on non-linear change.

6.93. EObjectBasedCalibrationPrecisionVsSpeedTradeOff Enum

The precision vs speed trade-off.

Namespace: Euresys::Open_eVision::Easy3D

EObjectBasedCalibrationPrecisionVsSpeedTradeOff_Fast	Fast mode.

EObjectBasedCalibrationPrecisionVsSpeedTradeOff_Balanced	Balanced mode.
EObjectBasedCalibrationPrecisionVsSpeedTradeOff_Precise	Precise mode.

6.94. EObjectBasedCalibrationType Enum

The type of the calibration object.

Namespace: Euresys::Open_eVision::Easy3D

EObjectBasedCalibrationType_DoublePyramid	Double Pyramid calibration model.
EObjectBasedCalibrationType_TruncatedDoublePyramid	Double Truncated Pyramid calibration model.
EObjectBasedCalibrationType_NotDefined	Not Defined.

6.95. EOCR2Classifier Enum

This enumeration contains the different types of classifier for character recognition in EOCR2.

Namespace: Euresys::Open_eVision

EOCR2Classifier_DatabaseClassifier	This classifier uses a EOCR2CharacterDatabase to recognize characters.
EOCR2Classifier_Industrial_A_Z_0_9_P	This classifier is a pre-trained classifier using Deep-Learning to recognize characters of various fonts commonly used in an industrial context (chips, processors, ...). Current allowed characters are numbers, upper-case letters and few punctuation symbols (. , : / + -). This classifier is not suited for characters using specific fonts like OCR-A or SEMI-FONT.
EOCR2Classifier_OCRA_A_Z_0_9_P	This classifier is a pre-trained classifier using Deep-Learning to recognize characters with OCR-A font.
EOCR2Classifier_SEMI_A_Z_0_9_P	This classifier is a pre-trained classifier using Deep-Learning to recognize characters with SEMI-OCR font.

6.96. EOCR2DetectionMethod Enum

This enumeration contains the possible methods for text detection in EOCR2.

Namespace: Euresys::Open_eVision

EOCR2DetectionMethod_FixedWidth	This method is suited for detecting texts with fixed-with fonts and dotted characters.
EOCR2DetectionMethod_Proportional	This method is suited for detecting texts with proportional fonts.

6.97. EOCR2SegmentationMethod Enum

This enumeration contains the possible methods for image segmentation in EOCR2.

Namespace: Euresys::Open_eVision

EOCR2SegmentationMethod_Local	This method is more complex and suited for segmenting images with text on a nonuniform background, it uses a local threshold value that can be different for each character of the text.
EOCR2SegmentationMethod_Global	This method is fast and suited for segmenting images with clear text on a uniform background, it uses a global threshold value.

6.98. EOCRClass Enum

Allowed values for the class of pattern in EasyOCR. These classes have no pre-defined meaning, the user is free to give them any meaning they like. For instance, class 0 could contains only digits, class 1 only the forward slash character, class 3 uppercase letters, etc. Any choice is allowed, as long as the correct classes are specified during the learning process. During recognition/reading, the user can specify the expected class for each character. This means that if a forward slash is expected at that position, they can say it should be class 1 (following the example from above). This will improve the recognition rate and speed because the algorithm only has to choose between characters within the specified class.

Namespace: Euresys::Open_eVision

EOCRClass__0	Character belongs to class 0.
EOCRClass__1	Character belongs to class 1.
EOCRClass__2	Character belongs to class 2.
EOCRClass__3	Character belongs to class 3.
EOCRClass__4	Character belongs to class 4.
EOCRClass__5	Character belongs to class 5.

EOCRClass__6	Character belongs to class 6.
EOCRClass__7	Character belongs to class 7.
EOCRClass__8	Character belongs to class 8.
EOCRClass__9	Character belongs to class 9.
EOCRClass__10	Character belongs to class 10.
EOCRClass__11	Character belongs to class 11.
EOCRClass__12	Character belongs to class 12.
EOCRClass__13	Character belongs to class 13.
EOCRClass__14	Character belongs to class 14.
EOCRClass__15	Character belongs to class 15.
EOCRClass__16	Character belongs to class 16.
EOCRClass__17	Character belongs to class 17.
EOCRClass__18	Character belongs to class 18.
EOCRClass__19	Character belongs to class 19.
EOCRClass__20	Character belongs to class 20.
EOCRClass__21	Character belongs to class 21.
EOCRClass__22	Character belongs to class 22.
EOCRClass__23	Character belongs to class 23.
EOCRClass__24	Character belongs to class 24.
EOCRClass__25	Character belongs to class 25.
EOCRClass__26	Character belongs to class 26.
EOCRClass__27	Character belongs to class 27.
EOCRClass__28	Character belongs to class 28.
EOCRClass__29	Character belongs to class 29.
EOCRClass__30	Character belongs to class 30.
EOCRClass_Digit	Character belongs to class 0. Equivalent to EOCRClass__0.
EOCRClass_UpperCase	Character belongs to class 1. Equivalent to EOCRClass__1.
EOCRClass_LowerCase	Character belongs to class 2. Equivalent to EOCRClass__2.
EOCRClass_Special	Character belongs to class 3. Equivalent to EOCRClass__3.
EOCRClass_Extended	Character belongs to class 4. Equivalent to EOCRClass__4.
EOCRClass_AllClasses	Character belongs to all classes, from 0 to 31 included.

6.99. EOCRColor Enum

Allowed values for the text color in EasyOCR.

Namespace: Euresys::Open_eVision

EOCRColor_BlackOnWhite	The characters appear darker than the background.
EOCRColor_WhiteOnBlack	The characters appear lighter than the background.
EOCRColor_DarkOnLight	The characters appear darker than the background. No thresholding takes place when the characters are learnt and/or recognized.
EOCRColor_LightOnDark	The characters appear lighter than the background. No thresholding takes place when the characters are learnt and/or recognized.

6.100. EPathVectorDrawOption Enum

-

Namespace: Euresys::Open_eVision

EPathVectorDrawOption_TopLeft	Link the pixel of the path by the top left corner of the pixels.
EPathVectorDrawOption_Center	Link the pixel of the path by the center of the pixels.
EPathVectorDrawOption_Fill	Fill the pixel that are part of the path.

6.101. EPatternStyle Enum

Allowed values for the nature of the pattern in EasyFind.

Namespace: Euresys::Open_eVision

EPatternStyle_Rasterized	Defines the rasterized pattern style.
EPatternStyle_Vectorized	Defines the vectorized pattern type.
EPatternStyle_Unknown	-

6.102. EPatternType Enum

Allowed values for the type of patterns in EasyFind.

Namespace: Euresys::Open_eVision

EPatternType_ ConsistentEdges	Defines the ConsistentEdges pattern type.
EPatternType_ ThinStructure	Defines the ThinStructure pattern type.
EPatternType_ Unknown	-

6.103. EPenStyle Enum

Pen style.

Namespace: Euresys::Open_eVision

EPenStyle_Solid	Solid pen.
EPenStyle_Dash	Dash pen.
EPenStyle_DashDot	Dash dot pen.
EPenStyle_DashDotDot	Dash dot dot pen.
EPenStyle_Dot	Dot pen.

6.104. EPhotometricStereoContrast Enum

This enumeration contains the possible ways to handle the contrast when retrieving Albedos, Gaussian curvatures or Mean curvatures. See EPhotometricStereoImager::GetAlbedos, EPhotometricStereoImager::ComputeMeanCurvatures and EPhotometricStereoImager::ComputeGaussianCurvatures.

Namespace: Euresys::Open_eVision::Easy3D

EPhotometricStereoContrast_FullRange	The full range of the image is linearly scaled. The image may need further post-processing in this case to remove outliers.
EPhotometricStereoContrast_HighContrast	Produce images with a high contrast, to do so, outliers will be clipped. Scaling is linear. This increases the contrast so images may seem noisy.
EPhotometricStereoContrast_FixedRange	Scaling is performed linearly using the specified range. Some default values are provided but user may need to specify them as they depend on the camera resolution and the object captured. This is interesting when several images of similar objects must have the same range.

6.105. EPickingMode Enum

-

Namespace: Euresys::Open_eVision

EPickingMode_All	-
EPickingMode_Begin	-
EPickingMode_End	-
EPickingMode_Central	-
EPickingMode_Score	-

6.106. EPlaneCropperType Enum

Type of crop to use in EPlaneCropper.

Namespace: Euresys::Open_eVision::Easy3D

EPlaneCropperType_KeepAbove	Only the points above the plane are selected.
EPlaneCropperType_KeepBelow	Only the points below the plane are selected.
EPlaneCropperType_KeepClose	Only the points closer to the plane than a specified distance ("maxDistance") are selected.
EPlaneCropperType_KeepFar	Only the points further away from the plane than a specified distance ("maxDistance") are selected.

6.107. EPlotItem Enum

Defines how the profile is drawed across a gauge.

Namespace: Euresys::Open_eVision

EPlotItem_Transitions	Displays the profile along a point location gauge.
EPlotItem_Peak	Displays the corresponding derivative curve.
EPlotItem_Thresholds	Displays the threshold and minimum amplitude levels.
EPlotItem_Points	Displays the valid transitions.

6.108. EPointCloudFilteringMethod Enum

Type of filtering method to use in EPointCloudFilter.

Namespace: Euresys::Open_eVision::Easy3D

EPointCloudFilteringMethod_DistanceFromMean	Removes points for which the distance from the average of the neighborhood is larger than for other points of the cloud. Points whose criterion is greater than mean + thresholdMultiplier * stddev are removed, mean and stddev being the mean and standard derivation of the criterion across all points of the cloud and thresholdMultiplier a factor set by EPointCloudFilter::ThresholdMultiplier.
EPointCloudFilteringMethod_HighStandardDeviation	Removes points for which the standard deviation calculated in the neighborhood is larger than for other points of the cloud. Points whose criterion is greater than mean + thresholdMultiplier * stddev are removed, mean and stddev being the mean and standard derivation of the criterion across all points of the cloud and thresholdMultiplier a factor set by EPointCloudFilter::ThresholdMultiplier. Unlike EPointCloudFilteringMethod_DistanceFromMean, this method does not filter points at the extremities of a surface. On the other hand, it is more sensitive to the value of the threshold.

6.109. EPolygonMeasurementMode Enum

PolygonGauge measurement mode

Namespace: Euresys::Open_eVision

EPolygonMeasurementMode_Global	The polygon is rigid, only the global position and the orientation are adjusted. If EPolygonGauge::EnableScaling is enabled, a scale factor is also computed.
EPolygonMeasurementMode_Edge	The edges of the polygon are measured independently. The result is a new polygon composed by these fitted edges. The measured polygon has the same number of vertices.
EPolygonMeasurementMode_Point	The polygon edges are sampled to fit closely to the measured points. The new polygon uses all the measured points of the contour.

6.110. EProjectionType Enum

This enumeration contains the possible values for the parameter of E3DViewer::ProjectionType method.

Namespace: Euresys::Open_eVision::Easy3D

EProjectionType_ Orthographic	Use an orthographic projection
EProjectionType_ Perspective	Use an perspective projection

6.111. EQRCodeCodingMode Enum

This enumeration contains the possible values for the coding mode of a QR code. Used by EQRCodeDecodedStream.

Namespace: Euresys::Open_eVision

EQRCodeCodingMode_ Basic	The QR code does not use a specific coding mode.
EQRCodeCodingMode_ Fnc1_Gs1	The QR code uses the FNC1/GS1 coding mode (FNC1 in first position).
EQRCodeCodingMode_ Fnc1_Aim	The QR code uses the FNC1/AIM coding mode (FNC1 in second position).
EQRCodeCodingMode_ ECI	The QR code uses Extended Channel Interpretation (ECI) coding mode.

6.112. EQRCodeEncoding Enum

This enumeration contains the possible values for the encoding used for parts of the bit stream of a QR code, contained by the EQRCodeDecodedStreamPart object.

Namespace: Euresys::Open_eVision

EQRCodeEncoding_ Numeric	The stream part is coded numerically.
EQRCodeEncoding_ Alphanumeric	The stream part is coded alphanumerically.
EQRCodeEncoding_ Byte	The stream part is coded as bytes.
EQRCodeEncoding_ Kanji	The stream part is coded in Kanji.

6.113. EQRCodeLevel Enum

This enumeration contains the possible values for the level of error correction of a QR code. Used in EQRCode.

Namespace: Euresys::Open_eVision

EQRCodeLevel_L	The QR code is level L (about 7% of error correction).
EQRCodeLevel_M	The QR code is level M (about 15% of error correction).
EQRCodeLevel_Q	The QR code is level Q (about 25% of error correction).
EQRCodeLevel_H	The QR code is level H (about 30% of error correction).
EQRCodeLevel_NoCorrection	The code has only error detection capacity (micro QR code (Version M1) only).

6.114. EQRCodeModel Enum

This enumeration contains the possible values for a QR code model. Used in EQRCode and EQRCodeReader.

Namespace: Euresys::Open_eVision

EQRCodeModel_Model1	The QR code is a model 1.
EQRCodeModel_Model2	The QR code is a model 2 or 2005.
EQRCodeModel_MicroQR	The QR code is a Micro QR.

6.115. EQRCodeScanPrecision Enum

This enumeration contains the possible values for the scanning precision used by an EQRCodeReader object.

Namespace: Euresys::Open_eVision

EQRCodeScanPrecision_Automatic	The QR code reader determines the scan precision automatically.
EQRCodeScanPrecision_Fine	The QR code reader scans finely the search field to find the QR codes. This value is recommended for small images or large images with small QR codes.

EQRCodeScanPrecision_Coarse	The QR code reader scans coarsely the search field to find the QR codes. This value is recommended for large images with medium to large QR codes.

6.116. EQRDetectionMethod Enum

This enumeration contains the possible detection methods the EQRCodeReader can use to detect QR codes. Combinations of the methods are allowed.

Namespace: Euresys::Open_eVision

EQRDetectionMethod_AdaptiveThreshold	This method detects finder patterns based on adaptive thresholding of the image.
EQRDetectionMethod_Gradient	This method detects finder patterns based on gradients in the image.
EQRDetectionMethod_PerspectiveLegacy	This selects the gradient-based detection algorithms with improved perspective mode developed for eVision 1.2.2.
EQRDetectionMethod_GradientLegacy	This selects the gradient-based detection algorithms with basic perspective mode developed for eVision 1.2.2.

6.117. EQRDetectionTradeOff Enum

This enumeration contains several settings for the trade-off between detection speed and reliability of the EasyQRCode methods.
Setting this parameter will overwrite the current settings for EQRDetectionMethod and EQRCodeScanPrecision.

Namespace: Euresys::Open_eVision

EQRDetectionTradeOff_FavorSpeed	This setting gives the fastest detection speed, but may reduce the detection accuracy. Sets EQRDetectionMethod_AdaptiveThreshold and EQRCodeScanPrecision_Coarse.
EQRDetectionTradeOff_Balanced	This setting gives a balance between detection speed and reliability. Sets EQRDetectionMethod_AdaptiveThreshold | EQRDetectionMethod_Gradient and EQRCodeScanPrecision_Automatic.
EQRDetectionTradeOff_FavorReliability	This setting gives the best detection reliability, at the cost of detection speed. Sets EQRDetectionMethod_AdaptiveThreshold | EQRDetectionMethod_Gradient and EQRCodeScanPrecision_Fine.

EQRDetectionTradeOff_Custom	This setting is returned when the current settings EQRDetectionMethod and EQRCodeScanPrecision do not match any of the EQRDetectionTradeOff presets. This choice should NOT be used to set a desired trade-off setting.

6.118. EReadingOrientation Enum

Represents the orientation to assume when decoding a barcode without start/stop patterns, for example those of PharmacodeOneTrack.

Namespace: Euresys::Open_eVision::EasyBarCode2

EReadingOrientation_LeftToRight	Barcodes without start/stop patterns will be decoded with the orientation that is closest from left to right.
EReadingOrientation_RightToLeft	Barcodes without start/stop patterns will be decoded with the orientation that is closest from right to left.
EReadingOrientation_TopToBottom	Barcodes without start/stop patterns will be decoded with the orientation that is closest from top to bottom.
EReadingOrientation_BottomToTop	Barcodes without start/stop patterns will be decoded with the orientation that is closest from bottom to top.

6.119. EReadMode Enum

This enumeration contains the possible operation modes for the EMatrixCodeReader::Read method

Namespace: Euresys::Open_eVision::EasyMatrixCode2

EReadMode_Speed	The EMatrixCodeReader::Read method will halt as soon as one of the following is true: (1) it has found the required number of codes given by EMatrixCodeReader::MaxNumCodes. (2) it reaches the timelimit given by EMatrixCodeReader::TimeOut. (3) it has completely finished its process. This mode will result in the shortest processing times.
EReadMode_Quality	The EMatrixCodeReader::Read method will keep trying to improve its detection until one of the following is true: (1) it reaches the timelimit given by EMatrixCodeReader::TimeOut. (2) it has completely finished its process. This mode will results in the best grading results.

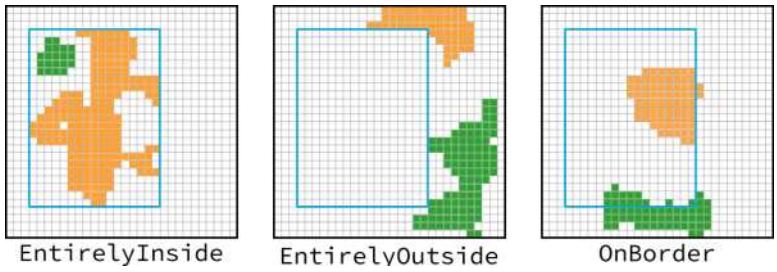
6.120. ERectangleMode Enum

The modes that specify how the selection of coded elements with a rectangle behaves.

Namespace: Euresys::Open_eVision

ERectangleMode_ EntirelyInside	Takes into consideration only the coded elements that entirely lie inside the given rectangle, not touching its borders
ERectangleMode_ EntirelyOutside	Takes into consideration only the coded elements that entirely lie outside the given rectangle, not touching its borders.
ERectangleMode_ InsideOrOnBorder	Takes into consideration only the coded elements that entirely lie inside the given rectangle, or that touch its borders.
ERectangleMode_ OutsideOrOnBorder	Takes into consideration only the coded elements that entirely lie outside the given rectangle, or that touch its borders.
ERectangleMode_ OnBorder	Takes into consideration only the coded elements that touch the borders of the rectangle.

Remarks



<caption>Examples of objects matching the categories EntirelyInside, EntirelyOutise, and OnBorder.</caption>

6.121. EReductionMode Enum

The reduction mode to be used when learning a Consistent Edges model.

Namespace: Euresys::Open_eVision

EReductionMode_Auto	Use the best-guess algorithm for selecting the reduction strength when learning a model.
EReductionMode_ Manual	Use a user-set value for the reduction strength when learning a model (cf. the property EPatternFinder::ReductionStrength).
EReductionMode_ Unknown	-

6.122. EReferenceNoise Enum

Enumeration for specifying how a reference image is affected by noise in EasyImage.

Namespace: Euresys::Open_eVision

EReferenceNoise_NoReference	The reference image is free from noise (synthetic image or noise source cancelled).
EReferenceNoise_SameAsImage	The reference image is contaminated by the same noise source as the source image.

6.123. ERgbStandard Enum

Allowed values for the RGB standard in EasyColor.

Namespace: Euresys::Open_eVision

ERgbStandard_Ntsc	NTSC primaries with the following CIE XYZ coordinates: Red: (0.607, 0.299, 0.000), Green: (0.174, 0.587, 0.066), Blue: (0.201, 0.114, 1.117). NTSC uses the white point "C". When used, color to gray equation is R * 0.299 + G * 0.587 + B * 0.114.
ERgbStandard_Pal	PAL primaries with the following CIE XYZ coordinates: Red: (0.4303, 0.2219, 0.0202), Green: (0.3416, 0.7068, 0.1296), Blue: (0.1784, 0.0713, 0.9393). PAL uses the white point "D65". When used, color to gray equation is R * 0.2219 + G * 0.7068 + B * 0.0713.
ERgbStandard_Smpte	SMPTE primaries with the following CIE XYZ coordinates: Red: (0.393, 0.212, 0.019), Green: (0.365, 0.701, 0.112), Blue: (0.192, 0.087, 0.958). SMPTE uses the white point "D65". When used, color to gray equation is R * 0.212 + G * 0.701 + B * 0.087.
ERgbStandard_SRGB	sRGB primaries with the following CIE XYZ coordinates: Red: (0.412, 0.212, 0.019), Green: (0.357, 0.715, 0.119), Blue: (0.180, 0.072, 0.950). sRGB uses the white point "D65". When used, color to gray equation is R * 0.212 + G * 0.715 + B * 0.072.

Remarks

The definition of the RGB primaries is not unique. In principle, there is one RGB system for each set of phosphors used in color monitors. Anyway, the CCIR has defined standard combinations for use in digital TV broadcast. Before performing a conversion, function EasyColor::RgbStandard can be used to specify the standard used.

6.124. ERoiHit Enum

Describes the ROI that was hit by the mouse cursor.

Namespace: Euresys::Open_eVision

ERoiHit_NoHit	No ROI.
ERoiHit_Learn_0	First learning ROI.
ERoiHit_Learn_1	Second learning ROI.
ERoiHit_Match_0	First matching ROI.
ERoiHit_Match_1	Second matching ROI.
ERoiHit_Inspect	Inspection ROI.

6.125. ERotationRightAngles Enum

Clockwise Right angles for rotation.

Namespace: Euresys::Open_eVision

ROTATION_90_CW	-
ROTATION_180_CW	-
ROTATION_270_CW	-

6.126. ESegmentationMethod Enum

The segmentation methods that are available to the image encoder.

Namespace: Euresys::Open_eVision

ESegmentationMethod_Custom	-
ESegmentationMethod_BinaryImage	Segmentation of binary images (cf. EBinaryImageSegmenter).
ESegmentationMethod_ColorRangeThreshold	Segments a color image by specifying a cube in the RGB space (cf. EColorRangeThresholdSegmenter).
ESegmentationMethod_ColorSingleThreshold	Segments a color image by specifying a single threshold (cf. EColorSingleThresholdSegmenter).
ESegmentationMethod_GrayscaleDoubleThreshold	Segments a grayscale image by specifying a double threshold (cf. EGrayscaleDoubleThresholdSegmenter).
ESegmentationMethod_GrayscaleSingleThreshold	Segments a grayscale image by specifying a single threshold (cf. EGrayscaleSingleThresholdSegmenter).

ESegmentationMethod_ImageRange	Segments an image by specifying a pixel-by-pixel double threshold (cf. EGrayscaleDoubleThresholdSegmenter).
ESegmentationMethod_ReferenceImage	Segments an image by specifying a pixel-by-pixel single threshold (cf. EGrayscaleSingleThresholdSegmenter).
ESegmentationMethod_LabeledImage	Segments an image by mapping the value of the pixels directly to a layer index (cf. ELabeledImageSegmenter).

Remarks

The parameters of the segmentation methods are configured through the getters finishing by "Segmenter" that are available in EImageEncoder.

6.127. ESegmentationMode Enum

Allowed values for the segmentation mode in EasyOCR.

Namespace: Euresys::Open_eVision

ESegmentationMode_KeepObjects	After segmentation, keep the blobs as they were found.
ESegmentationMode_RepasteObjects	After segmentation, group together the blobs believed to belong to the same character.

6.128. ESelectByPosition Enum

Allowed values for the selection mode of ECodedImage.

Namespace: Euresys::Open_eVision

ESelectByPosition_InsertIn	Insert the objects completely inside the given area.
ESelectByPosition_InsertTouch	Insert all the objects with a non-empty intersection with the given area.
ESelectByPosition_InsertOut	Insert the objects completely outside the given area.
ESelectByPosition_RemoveIn	Remove the objects completely inside the given area.
ESelectByPosition_RemoveTouch	Remove all the objects with a non-empty intersection with the given area.
ESelectByPosition_RemoveOut	Remove the objects completely outside the given area.
ESelectByPosition_RemoveBorder	Remove the objects outside the given area (including the objects touching the given area boundary).

Remarks

When specifying the position by means of an ROI, the minimum width and height of the ROI object must be at least 3 pixels. This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See ECodedImage2 for the new API.

6.129. ESelectionFlag Enum

Specifies to which subset of a selection an operation should be applied in EasyObject and EasyOCV.

Namespace: Euresys::Open_eVision

ESelectionFlag_Any	The operation applies to both selected and unselected items.
ESelectionFlag_True	The operation applies to selected items only.
ESelectionFlag_False	The operation applies to unselected items only.

6.130. ESelectOption Enum

Allowed values for the selection mode of ECodedImage. This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See ECodedImage2 for the new API.

Namespace: Euresys::Open_eVision

ESelectOption_InsertAll	Add all objects.
ESelectOption_ InsertGreaterOrEqual	Add all objects with feature value above the upper threshold.
ESelectOption_ InsertLesserOrEqual	Add all objects with feature value below the lower threshold.
ESelectOption_ InsertRange	Add all objects with feature value between or equal to the lower and upper thresholds.
ESelectOption_ RemoveAll	Delete all objects.
ESelectOption_ RemoveGreaterOrEqual	Delete all objects with feature value above the lower threshold.
ESelectOption_ RemoveLesserOrEqual	Delete all objects with feature value below the upper threshold.
ESelectOption_ RemoveRange	Delete all objects with feature value between or equal to the lower and upper thresholds.
ESelectOption_ InsertOutOfRange	Add all objects with feature value above the upper and below the lower threshold.

ESelectOption_ RemoveOutOfRange	Delete all objects with feature value above the upper and below the lower threshold.

6.131. ESerializerFileWriterMode Enum

Creation mode of the file.

Namespace: Euresys::Open_eVision

ESerializerFileWriterMode_Create	Creates the archive file on the hard disk. If the file already exists, the ESerializer::CreateFileWriter method returns NULL.
ESerializerFileWriterMode_Overwrite	Overwrites the previously created archive if it already exists, or creates it otherwise.
ESerializerFileWriterMode_Append	Appends the data at the end of the previously created archive, or creates the archive if it doesn't exist.

6.132. EShapeBehavior Enum

Allowed values for conditions on the behavior of a shape.

Namespace: Euresys::Open_eVision

EShapeBehavior_ Visible	Identifies a visible shape.
EShapeBehavior_ Selected	Identifies a selected shape.
EShapeBehavior_ Selectable	Identifies a selectable shape.
EShapeBehavior_ Dragable	Identifies a dragable shape.
EShapeBehavior_ Rotatable	Identifies a rotatable shape.
EShapeBehavior_ Resizable	Identifies a resizable shape.
EShapeBehavior_ Labeled	Identifies a labeled shape.
EShapeBehavior_Active	Identifies an active shape.
EShapeBehavior_ Passed	Identifies a non defective shape.

6.133. EShapeType Enum

Shape type.

Namespace: Euresys::Open_eVision

EShapeType_NoShape	-
EShapeType_PointShape	Defines a point shape.
EShapeType_LineShape	Defines a line shape.
EShapeType_CircleShape	Defines a circle shape.
EShapeType_WedgeShape	Defines a wedge shape.
EShapeType_RectangleShape	Defines a rectangle shape.
EShapeType_FrameShape	Defines a frame shape.
EShapeType_WorldShape	Defines a world shape.
EShapeType_PointGauge	Defines a point location gauge.
EShapeType_LineGauge	Defines a line fitting gauge.
EShapeType_CircleGauge	Defines a circle fitting gauge.
EShapeType_RectangleGauge	Defines a rectangle fitting gauge.
EShapeType_WedgeGauge	Defines a wedge fitting gauge.
EShapeType_PolygonGauge	Defines a polygon fitting gauge.
EShapeType_PolygonShape	Defines a polygon shape.

6.134. EShiftingMode Enum

Allowed values for the shifting mode of EasyOCR.

Namespace: Euresys::Open_eVision

EShiftingMode_Chars	Each character is moved individually.
EShiftingMode_Text	The all set of characters is moved together.

6.135. ESingleThresholdMode Enum

The single threshold mode for the selection of coded elements with respect to a given feature.

Namespace: Euresys::Open_eVision

ESingleThresholdMode_Less	The value of the feature must be strictly less than the threshold.
ESingleThresholdMode_LessEqual	The value of the feature must be less or equal to the threshold.
ESingleThresholdMode_Equal	The value of the feature must be equal to the threshold.
ESingleThresholdMode_GreaterEqual	The value of the feature must be greater or equal to the threshold.
ESingleThresholdMode_Greater	The value of the feature must be strictly greater than the threshold.
ESingleThresholdMode_Different	The value of the feature must be different to the threshold.

6.136. ESortDirection Enum

The sorting mode for selections of coded elements based on their features.

Namespace: Euresys::Open_eVision

ESortDirection_Ascending	Sorts the coded elements with respect to a given feature, in ascending order.
ESortDirection_Descending	Sorts the coded elements with respect to a given feature, in descending order.

6.137. ESortOption Enum

Allowed values for the sort mode of ECodedImage. This enumeration pertains to the EasyObject legacy API and should not be used for new developments. See ECodedImage2 for the new API.

Namespace: Euresys::Open_eVision

ESortOption_Ascending	Sort by increasing feature values.
ESortOption_Descending	Sort by decreasing feature values.

6.138. ESourceColorMode Enum

This enumeration contains the modes for the display of the point cloud colors.

Namespace: Euresys::Open_eVision::Easy3D

ESourceColorMode_Constant	Constant color for all the points (defined by E3DViewer::SetRenderSourceConstantColor).
ESourceColorMode_Ramp	Use a color ramp (defined by E3DViewer::GenerateColors).
ESourceColorMode_Attribute	Use the color attributes of the point cloud (if defined).

6.139. ESpotPolarity Enum

The possible polarities of an ESpot.

Namespace: Euresys::Open_eVision

ESpotPolarity_Black	The spot is darker than its surroundings.
ESpotPolarity_White	The spot is lighter than its surroundings.
ESpotPolarity_Any	The spot can be darker or lighter than its surroundings.

6.140. ESpotType Enum

The possible types of an ESpot.

Namespace: Euresys::Open_eVision

ESpotType_Particle	The spot is a compact defect, like dust, hole, pinch.
ESpotType_Scratch	The spot is an elongated defect, with the shape of a line.

6.141. EStockMeasurementUnit Enum

Allowed values for the type of Measurement Unit.

Namespace: Euresys::Open_eVision

EStockMeasurementUnit_None	Defines the None value type.
EStockMeasurementUnit_um	Defines the micron meter value type.
EStockMeasurementUnit_mm	Defines the milimeter value type.
EStockMeasurementUnit_cm	Defines the centimeter value type.
EStockMeasurementUnit_dm	Defines the decimeter value type.
EStockMeasurementUnit_m	Defines the meter value type.
EStockMeasurementUnit_dam	Defines the decameter value type.
EStockMeasurementUnit_hm	Defines the Hectometer value type.
EStockMeasurementUnit_km	Defines the Kilometer value type.
EStockMeasurementUnit_mil	Defines the milliliter value type.
EStockMeasurementUnit_inch	Defines the inch value type.
EStockMeasurementUnit_foot	Defines the foot value type.
EStockMeasurementUnit_yard	Defines the yard value type.
EStockMeasurementUnit_mile	Defines the mile value type.

6.142. ESupervisedSegmenterCapacity Enum

The capacity of the supervised segmentation network.
A larger capacity means that the underlying neural network is capable of learning more information but it will be slower.

Namespace: Euresys::Open_eVision::EasyDeepLearning

ESupervisedSegmenterCapacity_Small	Small capacity.
ESupervisedSegmenterCapacity_Normal	Normal capacity.
ESupervisedSegmenterCapacity_Large	Large capacity.

6.143. ESymbologies Enum

This enum is deprecated.

The symbologies supported by EasyBarCode.

Namespace: Euresys::Open_eVision

ESymbologies_Standard_Symbologies	Reserved for internal use
ESymbologies_Additional_Symbologies	Reserved for internal use
ESymbologies_Codabar	Codabar symbology.
ESymbologies_Code128	Code 128 symbology.
ESymbologies_Code25Interleaved	Code 25 Interleaved symbology.
ESymbologies_Code39	Code 39 symbology.
ESymbologies_Ean128	EAN 128 symbology.
ESymbologies_Ean13	EAN 13 symbology.
ESymbologies_Msi	MSI symbology.
ESymbologies_UpcA	UPC A symbology.
ESymbologies_UpcE	UPC E symbology.
ESymbologies_BinaryCode	Binary Code symbology.
ESymbologies_AdsAnker	Code ABC Anker symbology.
ESymbologies_Bc412	Code BC 412 symbology.
ESymbologies_Code11	Code 11 symbology.
ESymbologies_Code13	Code 13 symbology.
ESymbologies_Code25Datalogic	Code 25 DataLogic symbology.

ESymbologies_ Code25Matrix	Code 25 Matrix symbology.
ESymbologies_ Code25Iata	Code 25 IATA symbology.
ESymbologies_ Code25Industry	Code 25 Industry symbology.
ESymbologies_ Code25Compressed	Code 25 Compressed symbology.
ESymbologies_ Code25Inverted	Code 25 Inverted symbology.
ESymbologies_Code32	Code 32 symbology.
ESymbologies_ Code39Extended	Code 39 Extended symbology.
ESymbologies_ Code39Reduced	Code 39 Reduced symbology.
ESymbologies_Code93	Code 93 symbology.
ESymbologies_ Code93Extended	Code 93 Extended symbology.
ESymbologies_ CodeBcdMatrix	Code BCD Matrix symbology.
ESymbologies_CodeCip	Code CIP symbology.
ESymbologies_CodeStk	Code STK symbology.
ESymbologies_Ean8	EAN 8 symbology.
ESymbologies_ IbmDeltaDistanceA	IBM Delta Distance A symbology.
ESymbologies_Plessey	Plessey symbology.
ESymbologies_Telepen	Telepen symbology.
ESymbologies_Rss14	RSS-14 symbology.
ESymbologies_ Rss14Limited	RSS-14 Limited symbology.
ESymbologies_ Rss14Expanded	RSS-14 Expanded symbology.
ESymbologies_ Standard	Gathers all the symbologies belonging to the standard group.
ESymbologies_ Additional	Gathers all the symbologies belonging to the additional group.
ESymbologies_ Unknown	-

Remarks

Due to the large number of supported symbologies, they have been splitted into two groups. The most commonly used symbologies have been gathered under the name Standard symbologies. The remaining symbologies belong to the Additional symbologies group.

6.144. ESymbolPolarity Enum

Allowed values for the polarity of a code.

Namespace: Euresys::Open_eVision

ESymbolPolarity_BlackOnWhite	Dark cells on a light background.
ESymbolPolarity_WhiteOnBlack	Light cells on a dark background.

6.145. ETextLabelAlignment Enum

This enumeration contains the alignment of the text labels

Namespace: Euresys::Open_eVision::Easy3D

ETextLabelAlignment_FromPosition	Aligned towards the line between the anchor and the label (default)
ETextLabelAlignment_Left	Aligned left
ETextLabelAlignment_Right	Aligned right

6.146. EThinStructureMode Enum

Allowed values for the type of thin structures in EasyFind.

Namespace: Euresys::Open_eVision

EThinStructureMode_Auto	Lets EasyFind choose automatically the best contrast of thin elements.
EThinStructureMode_Dark	Favors thin elements darker than regions.
EThinStructureMode_Bright	Favors thin elements brighter than regions.

6.147. EThresholdMode Enum

The various modes for thresholding that are supported by Open eVision.

Namespace: Euresys::Open_eVision

EThresholdMode_Absolute	Reserved value. For absolute thresholding, use the threshold value itself, cast to EThresholdMode.
EThresholdMode_Relative	Relative threshold; determines the required threshold level so that a given fraction of the image pixels lie below it.
EThresholdMode_MinResidue	Selects a threshold value such that the quadratic difference between the source and thresholded image is minimized.
EThresholdMode_MaxEntropy	Selects a threshold value such that the entropy (i.e. the amount of information) of the resulting thresholded image is maximized.
EThresholdMode_Isodata	Selects a threshold value that lies halfway between the average dark gray value (i.e. gray levels below the threshold) and average light gray values (i.e. gray levels above the threshold).

6.148. ETrainingMode Enum

Allowed values for the training mode in EChecker.

Namespace: Euresys::Open_eVision

ETrainingMode_Precise	Precise training mode. This mode uses a two-pass training process to compute the threshold images. It gives better results at the cost of training time. Use this mode if you have harder to spot defects or lot of variation in the training images.
ETrainingMode_Quick	Quick training mode. This mode uses a quick, one-pass training process to compute the threshold images. Training time is lowered, but the results might be less reliable. Use this mode if you have easy to spot defects and few variation in the training images.

6.149. ETransitionChoice Enum

The transition selection method applied by the gauge measurement

Namespace: Euresys::Open_eVision

ETransitionChoice_NthFromBegin	N-th transition from the beginning (counting from 0).
ETransitionChoice_NthFromEnd	N-th transition from the end (counting from 0).

ETransitionChoice_LargestAmplitude	Transition whose peak has the largest amplitude value.
ETransitionChoice_LargestArea	Transition whose peak has the largest area value.
ETransitionChoice_Closest	Transition closest to the center.
ETransitionChoice_All	All transitions.

6.150. ETransitionType Enum

The type of transition to be retained by the gauge measurement

Namespace: Euresys::Open_eVision

ETransitionType_Bw	Black to white.
ETransitionType_Wb	White to black.
ETransitionType_BwOrWb	Black to white or white to black.
ETransitionType_Bwb	Black to white to black.
ETransitionType_Wbw	White to black to white.

6.151. EUIAPI Enum

This enumeration contains the various User Interface API supported by E3DViewer

Namespace: Euresys::Open_eVision::Easy3D

EUIAPI_Win32	The E3DViewer is used in a Windows Win32 application
EUIAPI_Qt	The E3DViewer is used in a Qt application (Windows or Linux)

6.152. EUnsupervisedScore Enum

Unsupervised scores for an unsupervised segmentation result.
These scores can be used for classification purposes.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EUnsupervisedScore_Absolute	Absolute score (normalized L1 distance between the input image and its reconstruction by the network).

EUnsupervisedScore_MeanSquared	Mean squared score (normalized L2 distance between the input image and its reconstruction by the network).
EUnsupervisedScore_Maximum	Maximum absolute score (normalized L-Inf distance between the input image and its reconstruction).
EUnsupervisedScore_BlobSize	Absolute score for which there is no contiguous area (blob) in the reconstruction error bigger than the smallest defect size of the unsupervised segmenter (See EUnsupervisedSegmenter).

6.153. EUnsupervisedSegmenterCapacity Enum

The capacity of the unsupervised network.
A larger capacity means that the underlying neural network is capable of learning more information but it will be slower.

Namespace: Euresys::Open_eVision::EasyDeepLearning

EUnsupervisedSegmenterCapacity_Small	Small capacity.
EUnsupervisedSegmenterCapacity_Normal	Normal capacity.
EUnsupervisedSegmenterCapacity_Large	Large capacity.

6.154. EVerbosity Enum

The verbosity level of a Memento message.

Namespace: Euresys::Open_eVision

EVerbosity_Critical	Critical (verbosity level 1). Highest Severity.
EVerbosity_Error	Error (verbosity level 2)
EVerbosity_Warning	Warning (verbosity level 3)
EVerbosity_Notice	Notice (verbosity level 4)
EVerbosity_Info	Info (verbosity level 5)
EVerbosity_Debug	Debug (verbosity level 6)
EVerbosity_Verbose	Verbose (verbosity level 7). Lowest Severity.

6.155. EViewDirection Enum

This enumeration contains the axis aligned view directions.

Namespace: Euresys::Open_eVision::Easy3D

EViewDirection_NegX	Negative X
EViewDirection_PosX	Positive X
EViewDirection_NegY	Negative Y
EViewDirection_PosY	Positive Y
EViewDirection_NegZ	Negative Z
EViewDirection_PosZ	Positive Z

6.156. EZMapGeneratorResolutionXYMode Enum

Mode used to specify the resolution of the ZMap generated on the x and y axes

Namespace: Euresys::Open_eVision::Easy3D

EZMapGeneratorResolutionXYMode_Absolute	resolution is specified in absolute units (e.g mm/pixel)
EZMapGeneratorResolutionXYMode_Relative	resolution is specified relative to the size of the object. A value of 1 means resolution is computed so as to get one point per pixel assuming points are uniformly distributed.

6.157. EZMapOrientationVectorMode Enum

The EZMap orientation, it's the direction of the X (width) axis of the ZMap.

Namespace: Euresys::Open_eVision::Easy3D

EZMapOrientationVectorMode_Auto	Chooses automatically the orientation of the ZMap.
EZMapOrientationVectorMode_XAxis	The X (width) axis of the ZMap is aligned with the world X axis.
EZMapOrientationVectorMode_YAxis	The X (width) axis of the ZMap is aligned with the world Y axis.
EZMapOrientationVectorMode_ZAxis	The X (width) axis of the ZMap is aligned with the world Z axis.
EZMapOrientationVectorMode_Explicit	The orientation vector is arbitrary and given by the method SetOrientationVector.

6.158. EZMapReferencePlaneMode Enum

The 3D reference plane used to build the EZMap.

Namespace: Euresys::Open_eVision::Easy3D

EZMapReferencePlaneMode_XPlane	The reference plane is orthogonal to the X axis (YZ domain).
EZMapReferencePlaneMode_YPlane	The reference plane is orthogonal to the Y axis (XZ domain).
EZMapReferencePlaneMode_ZPlane	The reference plane is orthogonal to the Z axis (XY domain). That's the default reference plane.
EZMapReferencePlaneMode_Explicit	The reference plane is arbitrary and given by the method SetReferencePlane.

6.159. Features Enum

Open eVision Features.

Namespace: Euresys::Open_eVision::LicenseFeatures

EasyGauge	-
EasyColor	-
EasyImage	-
EasyObject	-
EasyBarCode	-
EasyMatch	-
eVisionStudio	-
EasyFind	-
EasyMatrixCode	-
EasyOCR	-
EasyOCV	-
EasyQRCode	-
EasyOCR2	-
Easy3D	-
EasyClassify	-
Easy3DObject	-
Easy3DMatch	-
Easy3DLaserLine	-

EasySegment	-
EasyLocate	-
Gigelink	-
Recorder	-
EasySpotDetector	-