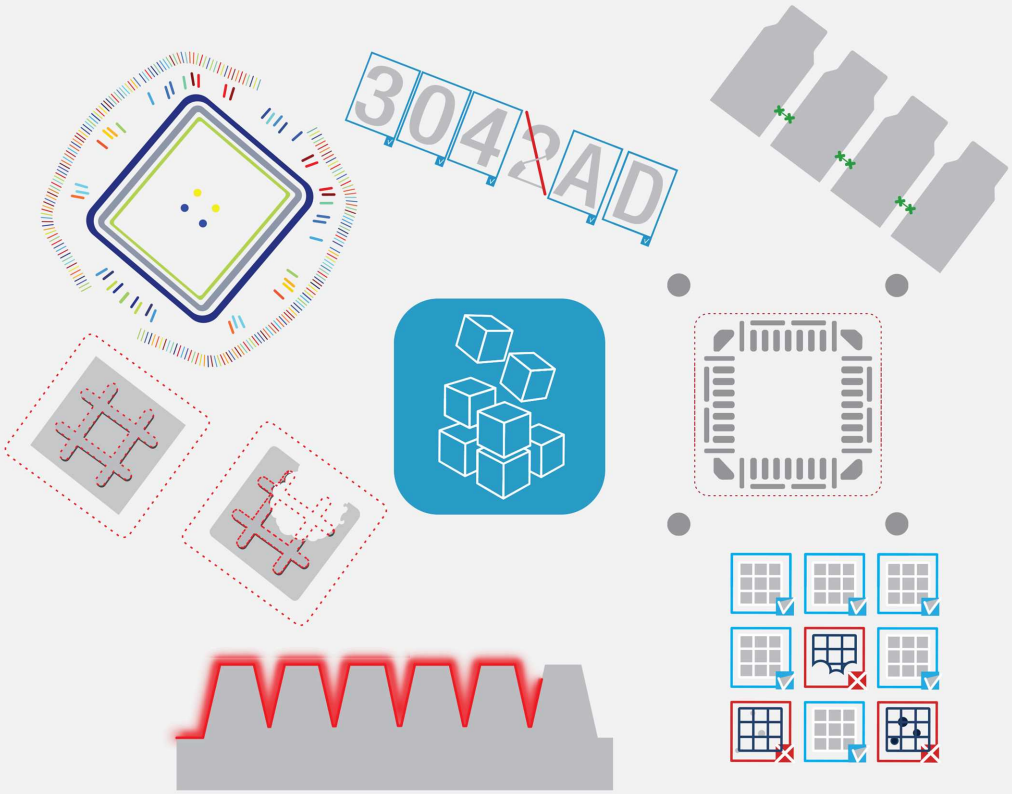
euresys
Empowering Computer Vision

REFERENCE MANUAL

Open eVision



This documentation is provided with Open eVision 2.16.1 (doc build 1155).
www.euresys.com

Contents

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	-

E

BW32PixelAccessor::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 -

E

C15PixelAccessor::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

2.2. Image and ROI Classes

Image Classes

EImageBW1
EImageBW8
EImageBW16
EImageBW32

EImageC15
EImageC16
EImageC24
EImageC24A
EImageC48

ROI Classes

EROIBW1
EROIBW8
EROIBW16
EROIBW32

EROIC15
EROIC16
EROIC24
EROIC24A
EROIC48

2.3. Region Classes

Classes

ERegion
ERectangleRegion
EPolygonRegion
ECircleRegion
EEllipseRegion

3. Libraries

3.1. Easy3D Library

Classes

Structs

E3DPoint
EDepth8
EDepth16
EDepth32f
ERenderStyle

Enumerations

"E3DAttribute Enum" on page 134
EAlignmentPolarity
EAttributeType
EAxisOriginMode
EColorRampMode
EMaximumAnalysisMode
ENoiseRemovalMethod
EObjectBasedCalibrationPrecisionVsSpeedTradeOff
EObjectBasedCalibrationType
EPlaneCropperType
EProjectionType
EZMapOrientationVectorMode
EZMapReferencePlaneMode

3.2. Easy3DLaserLine Library

Classes

EExplicitGeometricCalibrationModel
EObjectBasedCalibrationGenerator
EObjectBasedCalibrationModel
ELaserLineExtractor

3.3. Easy3DObject Library

Classes

E3DObject
E3DObjectExtractor

3.4. Easy3DMatch Library

Classes

E3DAligner
"E3DAlignment Class" on page 30
E3DAnomaly
E3DComparer
E3DMatch
E3DMatcher

Enumerations

EComparisonDistanceMode

3.5. EasyImage Library

Classes

EasyImage
EKernel
EMovingAverage

Enumerations

EArithmeticLogicOperation
EContourMode
EContourThreshold
EHistogramFeature
EKernelRectifier
EKernelRotation
EKernelType
EReferenceNoise
EThresholdMode

3.6. EasyClassify Library

Classes

EClassificationDataset Class
EClassificationMetrics Class
EClassificationResult Class
EClassifier Class
EDeepLearningTool Class

3.7. EasySegment Library

Classes

EClassificationDataset Class
EUnsupervisedSegmenterMetrics Class
EUnsupervisedSegmenterResult Class
EUnsupervisedSegmenter Class
ESupervisedSegmenter Class
ESupervisedSegmenterMetrics Class
ESupervisedSegmenterResult Class
EDeepLearningTool Class
EDeepLearningDefectDetectionMetrics Class

Structs

EROCPoint Struct

Enumerations

EUnsupervisedSegmenterCapacity Enum
ESupervisedSegmenterCapacity Enum
EConfusionMatrixElement Enum

3.8. EasyLocate Library

Classes

EClassificationDataset Class
ELocator Class
ELocatorResult Class
ELocatorMetrics Class
ELocatorObject Class
ELocatorPredictedObject Class

Structs

Enumerations

ELocatorCapacity Enum

4. Classes

4.1. E3DAlignment Class

Represents a 3D Alignment returned by E3DAligner.

Derived Class(es): E3DMatch

Namespace: Euresys::Open_eVision_2_16::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_2_16::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_2_16::Easy3D

[C++]

```
float GetError() const
```

E3DAlignment::operator=

Assignment operator.

Namespace: Euresys::Open_eVision_2_16::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_2_16::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_2_16::Easy3D

[C++]

```
int GetRefPoseMatchedIndex() const
```

4.2. E3DAxisDisplay Class

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

Namespace: Euresys::Open_eVision_2_16::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.
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_2_16::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_2_16::Easy3D

```
[C++]
Euresys::Open_eVision_2_16::Easy3D::EAxisOriginMode GetAxisOrigin() const
void SetAxisOrigin(Euresys::Open_eVision_2_16::Easy3D::EAxisOriginMode mode)
```

Remarks

The default mode is EAxisOriginMode.

E3DAxisDisplay::GetAxisOriginUserDefined

E3DAxisDisplay::SetAxisOriginUserDefined

Set and Get the axis origin.

Namespace: Euresys::Open_eVision_2_16::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_2_16::Easy3D

```
[C++]
E2DPoint GetAxisSize() const
void SetAxisSize(const E2DPoint& 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_2_16::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_2_16::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_2_16::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_2_16::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_2_16::Easy3D

```
[C++]
ERGBColor GetGridColor() const
void SetGridColor(ERGBColor color)
```

Remarks

The default color is grey.

E3DAxisDisplay::operator=

Assignment operator.

Namespace: Euresys::Open_eVision_2_16::Easy3D

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

Parameters

other
The E3DAxisDisplay object that should be copied.

E3DAxisDisplay::GetRenderAxis

E3DAxisDisplay::SetRenderAxis

Enables or disables the display of the axis.

Namespace: Euresys::Open_eVision_2_16::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_2_16::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_2_16::Easy3D

```
[C++]
E2DPoint GetRenderGridStep() const
void SetRenderGridStep(const E2DPoint& 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.

4.3. EZMap Class

Represents a generic ZMap type interface.

Derived Class(es): EZMap8 EZMap16 EZMap32f
Namespace: Euresys::Open_eVision_2_16::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	Returns an 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.

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	Returns an 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 grey values per pixel.
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.

ResetWorldTrans-formation	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.
SaveImageAndMetada-ta	Saves image format and Metadata JSON format.
SaveMetadata	Saves Metadata in JSON format.
Serialize	Serializes the EZMap object with all its attributes.
SerializeImage	Serializes the image associated to EZMap.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.
SetHeight	Access ZMap Height.
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.
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 grey values per pixel.
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.

E

ZMap::AddMetadata

Adds a metadata key (name) and value.
If the metadata key already exists, its value will be overwritten.

Namespace: Euresys::Open_eVision_2_16::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_2_16::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_2_16::Easy3D

```
[C++]
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_2_16::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.

4Vector

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.
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.)

EZMap::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision_2_16::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.
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.)

EZMap::GetBufferPtr

Retrieves the pointer to the internal pixel buffer.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void* GetBufferPtr(
 )
void* GetBufferPtr(
 OEV_INT32 x,
 OEV_INT32 y
 )
const void* GetBufferPtr(
 )
const void* GetBufferPtr(
 OEV_INT32 x,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
void* GetCheckedBufferPtr(
 OEV_INT32 x,
 OEV_INT32 y
 )
const void* GetCheckedBufferPtr(
 OEV_INT32 x,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key

The name of an exising 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_2_16::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_2_16::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::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_2_16::Easy3D

[C++]

```
E3DPoint GetWorldPositionFromPixelPosition(
 OEV_INT32 u,
 OEV_INT32 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_2_16::Easy3D

[C++]

```
E3DPoint GetZMapPositionFromPixelPosition(
 OEV_INT32 u,
 OEV_INT32 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_2_16::Easy3D

[C++]

```
OEV_INT32 GetHeight() const
void SetHeight(OEV_INT32 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_2_16::Easy3D

[C++]

```
void ImageToWorld(
 const E2DPoint& pixelPt,
 E2DPoint& 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_2_16::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_2_16::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_2_16::Easy3D

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

Parameters

path
Full path to the file.

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_2_16::Easy3D

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

Parameters

path
Full path to the file.

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_2_16::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_2_16::Easy3D

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

Parameters

path
Full path to the file.

EZMap::GetMapToWorldMatrix

Returns an 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_2_16::Easy3D

[C++]

```
const E2DTransformMatrix& GetMapToWorldMatrix() const
```

EZMap::ResetWorldTransformation

Reset the world transformation, Map to World and World to Map matrices become identity matrix.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
void ResetWorldTransformation(
 )
```

EZMap::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
OEV_INT2 GetRowPitch() const
```

EZMap::Save

Saves the EZMap object to the given Open eVision file.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

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

Parameters

path

The full path to the destination file.

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_2_16::Easy3D

[C++]

```
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision_2_16::EImageFileType type
 )
```

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.

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_2_16::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& pathImage,
 const std::string& pathMetadata,
 Euresys::Open_eVision_2_16::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_2_16::Easy3D

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

Parameters

path
The full path to the destination file.

EZMap::Serialize

Serializes the EZMap object with all its attributes.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer
The ESerializer object that is read from or writen to.

EZMap::SerializeImage

Serializes the image associated to EZMap.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void SerializeImage(
 ESerializer* serializer
 )
```

Parameters

serializer
The ESerializer object that is read from or writen to.

EZMap::SetBufferPtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void SetBufferPtr(
 OEV_INT32 width,
 OEV_INT32 height,
 void* imagePointer,
 OEV_INT32 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_2_16::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_2_16::Easy3D

```
[C++]
void SetSize(
 OEV_INT32 width,
 OEV_INT32 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_2_16::Easy3D

[C++]

```
Euresys::Open_eVision_2_16::EImageType GetType() const
```

EZMap::GetWidth

EZMap::SetWidth

Access ZMap Width.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
OEV_INT32 GetWidth() const
void SetWidth(OEV_INT32 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_2_16::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_2_16::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

Returns an 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_2_16::Easy3D

[C++]

```
const E3DTransformMatrix& GetWorldToMapMatrix() const
```

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_2_16::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_2_16::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_2_16::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_2_16::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_2_16::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 grey values per pixel.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
float GetZResolution() const
void SetZResolution(float resolution)
```

4.4. 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_2_16::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.
GetBufferPtr	Clears the ZMap: replaces all pixels with the undefined value.
GetCheckedBufferPtr	Retrieves the pointer to the pixel buffer.
GetHeight	Access ZMap Height.

GetMapToWorldMatrix	Returns an 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.
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	Returns an 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 mininum and maximum pixel values, excluding the undefined pixels.
GetZResolution	Resolution of the EZMap16 along the Z axis The resolution is the number of grey values per pixel.
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.
Serialize	Serializes the EZMap16 object with all its attributes.

SerializeImage	Serializes the image associated to EZMap16.
SetBufferPtr	Sets the pointer to an externally allocated image buffer.
SetHeight	Access ZMap Height.
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.
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 grey values per pixel.
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_2_16::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_2_16::Easy3D

[C++]

```
EImageBW16& AsEImage(
 )

const EImageBW16& AsEImage(
 )
```

EZMap16::Clear

Clears the ZMap: replaces all pixels with the undefined value.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
void Clear(
 )
```

EZMap16::ClearMetadata

Deletes all metadata.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
void ClearMetadata(
 )
```

EZMap16::ConvertCoordinatesMapToPixel

Converts 3D Map coordinates to Buffer coordinates

Namespace: Euresys::Open_eVision_2_16::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_2_16::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_2_16::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_2_16::Easy3D

```
[C++]
void DeleteMetadata(
 const std::string& Key
 )
```

Parameters

Key
-

EZMap16::Draw

Draws an EZMap16 in a device context.

Namespace: Euresys::Open_eVision_2_16::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.
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.)

EZMap16::DrawImage

Displays the internal image buffer

Namespace: Euresys::Open_eVision_2_16::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.
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.)

EZMap16::EZMap16

Creates a 16 bits EZMap.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void EZMap16(
 )
void EZMap16(
 OEV_INT32 width,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
void FillUndefinedPixels(
 EZMap16& outMap,
 Euresys::Open_eVision_2_16::Easy3D::EFillUndefinedPixelsDirection direction,
 Euresys::Open_eVision_2_16::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::GetBufferPtr

Clears the ZMap: replaces all pixels with the undefined value.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void* GetBufferPtr(
 )

void* GetBufferPtr(
 OEV_INT32 x,
 OEV_INT32 y
 )

const void* GetBufferPtr(
 )

const void* GetBufferPtr(
 OEV_INT32 x,
 OEV_INT32 y
 )
```

Parameters

x

-

y

-

EZMap16::GetCheckedBufferPtr

Retrieves the pointer to the pixel buffer.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void* GetCheckedBufferPtr(
 OEV_INT32 x,
 OEV_INT32 y
 )
const void* GetCheckedBufferPtr(
 OEV_INT32 x,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
std::string GetMetadata(
 const std::string& Key
 )
```

Parameters

Key
The name of an exising metadata.

EZMap16::GetPixel

Gets the value of a pixel .

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
EDepth16 GetPixel(
 OEV_INT32 x,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
bool GetPixelPositionFromWorldPosition(
 const E3DPoint& world_position,
 OEV_INT32& u,
 OEV_INT32& 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_2_16::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_2_16::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::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_2_16::Easy3D

```
[C++]
E?DPoint GetWorldPositionFromPixelPosition(
 OEV_INT?2 u,
 OEV_INT?2 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_2_16::Easy3D

[C++]

```
E3DPoint GetZMapPositionFromPixelPosition(
 OEV_INT32 u,
 OEV_INT32 v
 )
```

Parameters

u
Column of the pixel (bounds: [0,width[).

v
Row of the pixel (bounds: [0,height[).

EZMap16::GetZRange

Compute the mininum and maximum pixel values, excluding the undefined pixels.

Namespace: Euresys::Open_eVision_2_16::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_2_16::Easy3D

```
[C++]
float GetZValue(
 const OEV_INT32 x,
 const OEV_INT32 y
 )
```

Parameters

x
X Coordinate.

y
Y Coordinate.

EZMap16::GetHeight

EZMap16::SetHeight

Access ZMap Height.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
OEV_INT32 GetHeight() const
void SetHeight(OEV_INT32 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_2_16::Easy3D

[C++]

```
void ImageToWorld(
 const E2DPoint& pixelPt,
 E2DPoint& 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_2_16::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_2_16::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_2_16::Easy3D

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

Parameters

path

Full path to the file.

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_2_16::Easy3D

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

Parameters

path

Full path to the file.

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_2_16::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_2_16::Easy3D

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

Parameters

path

Full path to the file.

EZMap16::GetMapToWorldMatrix

Returns an 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_2_16::Easy3D

```
[C++]
const E2DTransformMatrix& GetMapToWorldMatrix() const
```

EZMap16::ModifyMetadata

Changes an existing metadata key and value. Throws an exception if it does not exist.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void ModifyMetadata(
 const std::string& Key,
 const std::string& value
 )
```

Parameters

Key

-

value

-

EZMap16::operator=

Assignment operator.

Namespace: Euresys::Open_eVision_2_16::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_2_16::Easy3D

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

EZMap16::GetRowPitch

Returns the buffer row pitch.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
OEV_INT32 GetRowPitch() const
```

EZMap16::Save

Saves the EZMap16 object to the given Open eVision file.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

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

Parameters

path

The full path to the destination file.

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_2_16::Easy3D

[C++]

```
void SaveImage(
 const std::string& path,
 Euresys::Open_eVision_2_16::EImageFileType type
 )
```

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.

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_2_16::Easy3D

```
[C++]
void SaveImageAndMetadata(
 const std::string& path,
 const std::string& pathMetadata,
 Euresys::Open_eVision_2_16::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_2_16::Easy3D

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

Parameters

path

The full path to the destination file.

EZMap16::Serialize

Serializes the EZMap16 object with all its attributes.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer
The ESerializer object that is read from or writen to.

EZMap16::SerializeImage

Serializes the image associated to EZMap16.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void SerializeImage(
 ESerializer* serializer
 )
```

Parameters

serializer
The ESerializer object that is read from or writen to.

EZMap16::SetBufferPtr

Sets the pointer to an externally allocated image buffer.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void SetBufferPtr(
 OEV_INT32 width,
 OEV_INT32 height,
 void* imagePointer,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
void SetPixel(
 EDepth16 value,
 OEV_INT32 x,
 OEV_INT32 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_2_16::Easy3D

```
[C++]
void SetResolution(
 float rx,
 float ry,
 float rz
 )
void SetResolution(
 E2DPoint 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_2_16::Easy3D

[C++]

```
void SetSize(
 OEV_INT32 width,
 OEV_INT32 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_2_16::Easy3D

[C++]

```
void SetZValue(
 const float value,
 const OEV_INT32 x,
 const OEV_INT32 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_2_16::Easy3D

[C++]

```
Euresys::Open_eVision_2_16::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_2_16::Easy3D

[C++]

```
EDepth16 GetUndefinedValue() const
```

EZMap16::GetWidth

EZMap16::SetWidth

Access ZMap Width.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
OEV_INT32 GetWidth() const
void SetWidth(OEV_INT32 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_2_16::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_2_16::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

Returns an 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_2_16::Easy3D

```
[C++]
const E3DTransformMatrix& GetWorldToMapMatrix() const
```

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_2_16::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_2_16::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_2_16::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_2_16::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_2_16::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 grey values per pixel.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
float GetZResolution() const
void SetZResolution(float resolution)
```

4.5. EZMapToPointCloudConverter Class

Generates an EPointCloud from a ZMap.

Namespace: Euresys::Open_eVision_2_16::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.
EZMapToPointCloudConverter	Creates an EZMapToPointCloudConverter object.

E

ZMapToPointCloudConverter::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.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

```
void Convert(
 const EZMap8& srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )

void Convert(
 const EZMap8& srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )

void Convert(
 const EZMap16& srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )

void Convert(
 const EZMap16& srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )
```

```
void Convert(
 const EZMap<2f& srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )
void Convert(
 const EZMap<2f& srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )
void Convert(
 const EZMap* srcZMap,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )
void Convert(
 const EZMap* srcZMap,
 ERegion& region,
 EPointCloud& pointCloud,
 bool inZMapSpace
 )
```

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).

region

-

Remarks

The destination point cloud will be cleared before being (re-)populated.

EZMapToPointCloudConverter::EZMapToPointCloudConverter

Creates an EZMapToPointCloudConverter object.

Namespace: Euresys::Open_eVision_2_16::Easy3D

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

Parameters

other

-

5. Structures

5.1. E3DPoint Struct

Represents a 3D point with floating point coordinates.

Namespace: Euresys::Open_eVision_2_16::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 euclidian distance between the E3DPoint and a segment represented by 2 other E3DPoint.
E3DPoint	Constructs a default E3DPoint object.
operator!=	Checks if two E3DPoint are strictly different.
operator==	Checks if two E3DPoint are strictly equal.
Serialize	Serializes the E3DPoint.
SquareDistanceTo	Returns the euclidean squared distance to another E3DPoint.

E3DPoint::DistanceTo

Returns the euclidean distance to another E3DPoint.

Namespace: Euresys::Open_eVision_2_16::Easy3D

[C++]

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

Parameters

point
The other point.

E3DPoint::DistanceToSegment

Returns the euclidian distance between the E3DPoint and a segment represented by 2 other E3DPoint.

Namespace: Euresys::Open_eVision_2_16::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_2_16::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::operator!=

Checks if two E3DPoint are strictly different.

Namespace: Euresys::Open_eVision_2_16::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_2_16::Easy3D

```
[C++]
bool operator==(
 const E3DPoint& point
 )
```

Parameters

point
The other point.

E3DPoint::Serialize

Serializes the E3DPoint.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
void Serialize(
 ESerializer* serializer
 )
```

Parameters

serializer
The ESerializer object that is read from or written to.

E3DPoint::SquareDistanceTo

Returns the euclidean squared distance to another E3DPoint.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
float SquareDistanceTo(
 const E3DPoint& point
 )
```

Parameters

point
The other point.

E3DPoint::X

X coordinate of the point.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
float X
```

E3DPoint::Y

Y coordinate of the point.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
float Y
```

E3DPoint::Z

Z coordinate of the point.

Namespace: Euresys::Open_eVision_2_16::Easy3D

```
[C++]
float Z
```

5.2. 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_2_16

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_2_16

```
[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_2_16

```
[C++]
OEV_INT32 GetSize()
```

EBW16::Value

The value of the pixel.

Namespace: Euresys::Open_eVision_2_16

```
[C++]
OEV_UINT16 Value
```

5.3. EMatrixCodeIso29158CalibrationParameters Struct

Holds all grading inputs pertaining to ISO/IEC 29158

Namespace: Euresys::Open_eVision_2_16

Properties

MLcal	Mean of the light from a histogram of the calibrated standard.
Rcal	Reported reflectance value, from a calibration 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

EMatrixCodeIso29158CalibrationParameters	-

EMatrixCodeIso29158CalibrationParameters::EMatrixCodeIso29158CalibrationParameters

-

Namespace: Euresys::Open_eVision_2_16

```
[C++]
void EMatrixCodeIso29158CalibrationParameters(
 )
```

EMatrixCodeIso29158CalibrationParameters::MLcal

Mean of the light from a histogram of the calibrated standard.

Namespace: Euresys::Open_eVision_2_16

```
[C++]
float MLcal
```

EMatrixCodeIso29158CalibrationParameters::Rcal

Reported reflectance value, from a calibration standard.

Namespace: Euresys::Open_eVision_2_16

```
[C++]
float Rcal
```

EMatrixCodeIso29158CalibrationParameters::SRcal

System response parameters(such as exposure and/or again) used to create an image of the calibration standard.

Namespace: Euresys::Open_eVision_2_16

[C++]

```
float SRcal
```

EMatrixCodeIso29158CalibrationParameters::SRtarget

System response parameters(such as exposure and/or again) used to create an image of the symbole under test.

Namespace: Euresys::Open_eVision_2_16

[C++]

```
float SRtarget
```

5.4. EQRCodeIso15415GradingParameters Struct

Holds all grading parameters pertaining to ISO/IEC 15415

Namespace: Euresys::Open_eVision_2_16

Properties

AdditionalParametersGrades	Additional Parameters Grades
AxialNonUniformity	Axial Non Uniformity

AxialNonUniformityGrade	Axial Non Uniformity Grade
DecodingGrade	Decoding Grade
FixedPatternDamageGrade	Fixed Pattern Damage Grade
GridNonUniformity	Grid Non Uniformity
GridNonUniformityGrade	Grid Non Uniformity Grade
HorizontalPrintGrowth	Horizontal Print Growth
ModulationGrade	Modulation Grade
OverallSymbolGrade	Overall Symbol Grade
ReflectanceMarginGrade	Reflectance Margin Grade
SymbolContrast	Symbol Contrast
SymbolContrastGrade	Symbol Contrast Grade
UnusedErrorCorrection	Unused Error Correction
UnusedErrorCorrectionGrade	Unused Error Correction Grade
VerticalPrintGrowth	Vertical Print Growth

Methods

EQRCodeIso15415GradingParameters	-

EQRCodeIso15415GradingParameters::AdditionalParametersGrades

Additional Parameters Grades

Namespace: Euresys::Open_eVision_2_16

```
[C++]
EQRCodeAdditionalParametersGrades AdditionalParametersGrades
```

EQRCodeIso15415GradingParameters::AxialNonUniformity

Axial Non Uniformity

Namespace: Euresys::Open_eVision_2_16

```
[C++]
float AxialNonUniformity
```

EQRCodeIso15415GradingParameters::AxialNonUniformityGrade

Axial Non Uniformity Grade

Namespace: Euresys::Open_eVision_2_16

[C++]

```
OEV_INT32 AxialNonUniformityGrade
```

EQRCodeIso15415GradingParameters::DecodingGrade

Decoding Grade

Namespace: Euresys::Open_eVision_2_16

[C++]

```
OEV_INT32 DecodingGrade
```

EQRCodeIso15415GradingParameters::EQRCodeIso15415GradingParameters

-

Namespace: Euresys::Open_eVision_2_16

[C++]

```
void EQRCodeIso15415GradingParameters(
 )
```

EQRCodeIso15415GradingParameters::FixedPatternDamageGrade

Fixed Pattern Damage Grade

Namespace: Euresys::Open_eVision_2_16

```
[C++]
OEV_INT32 FixedPatternDamageGrade
```

EQRCodeIso15415GradingParameters::GridNonUniformity

Grid Non Uniformity

Namespace: Euresys::Open_eVision_2_16

```
[C++]
float GridNonUniformity
```

EQRCodeIso15415GradingParameters::GridNonUniformityGrade

Grid Non Uniformity Grade

Namespace: Euresys::Open_eVision_2_16

```
[C++]
OEV_INT32 GridNonUniformityGrade
```

EQRCodeIso15415GradingParameters::HorizontalPrintGrowth

Horizontal Print Growth

Namespace: Euresys::Open_eVision_2_16

[C++]

```
float HorizontalPrintGrowth
```

EQRCodeIso15415GradingParameters::ModulationGrade

Modulation Grade

Namespace: Euresys::Open_eVision_2_16

[C++]

```
OEV_INT32 ModulationGrade
```

EQRCodeIso15415GradingParameters::OverallSymbolGrade

Overall Symbol Grade

Namespace: Euresys::Open_eVision_2_16

[C++]

```
OEV_INT32 OverallSymbolGrade
```

EQRCodeIso15415GradingParameters::ReflectanceMarginGrade

Reflectance Margin Grade

Namespace: Euresys::Open_eVision_2_16

[C++]

```
OEV_INT32 ReflectanceMarginGrade
```

EQRCodeIso15415GradingParameters::SymbolContrast

Symbol Contrast

Namespace: Euresys::Open_eVision_2_16

[C++]

```
float SymbolContrast
```

EQRCodeIso15415GradingParameters::SymbolContrastGrade

Symbol Contrast Grade

Namespace: Euresys::Open_eVision_2_16

[C++]

```
OEV_INT32 SymbolContrastGrade
```

EQRCodeIso15415GradingParameters::UnusedErrorCorrection

Unused Error Correction

Namespace: Euresys::Open_eVision_2_16

```
[C++]
float UnusedErrorCorrection
```

EQRCodeIso15415GradingParameters::UnusedErrorCorrectionGrade

Unused Error Correction Grade

Namespace: Euresys::Open_eVision_2_16

```
[C++]
OEV_INT?2 UnusedErrorCorrectionGrade
```

EQRCodeIso15415GradingParameters::VerticalPrintGrowth

Vertical Print Growth

Namespace: Euresys::Open_eVision_2_16

[C++]

float VerticalPrintGrowth

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_2_16::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 or INT32.
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.

Remarks

Numeric types are: UINT8, UINT16, UINT32, INT32, float, double.

6.2. E3DObjectFeature Enum

The list of possible extracted features for E3DObject

Namespace: Euresys::Open_eVision_2_16::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.

6.3. EAdaptiveThresholdMethod Enum

Adaptive thresholding modes.

Namespace: Euresys::Open_eVision_2_16

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_2_16::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_2_16

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_2_16

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. ECorrelationMode Enum

Allowed values for the EasyMatch correlation mode.

Namespace: Euresys::Open_eVision_2_16

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.

6.8. EDataSize Enum

Possible data sizes for an object feature.

Namespace: Euresys::Open_eVision_2_16

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

6.9. EDataType Enum

Possible data types for an object feature.

Namespace: Euresys::Open_eVision_2_16

EDataType_UnsignedInt	Unsigned integer.
EDataType_SignedInt	Signed integer.
EDataType_Float	Floating-point.

6.10. EDLDataAugmentationType Enum

EDLDataAugmentationType represents how the data augmentation transformation are generated.

Namespace: Euresys::Open_eVision_2_16::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.

6.11. EDongleType Enum

Dongle types.

Namespace: Euresys::Open_eVision_2_16

EDongleType_Legacy	Legacy Dongle
EDongleType_Neo	Neo Dongle
EDongleType_Unknown	All Dongles

6.12. EDoubleThresholdMode Enum

The double threshold mode for the selection of coded elements with respect to a given feature.

Namespace: Euresys::Open_eVision_2_16

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.

6.13. EDraggingMode Enum

Defines how the shape could be dragged

Namespace: Euresys::Open_eVision_2_16

EDraggingMode_Standard	Allows positioning the shape edges symmetrically.

EDraggingMode_ToEdges	Allows positioning each shape edge individually.

6.14. EDragHandle Enum

Allowed values for a handle identifier in the context of handle dragging.

Namespace: Euresys::Open_eVision_2_16

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_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.

6.15. EDrawableFeature Enum

The various features that can be drawn for coded elements.

Namespace: Euresys::Open_eVision_2_16

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°.
EDrawableFeature_FeretBox45	The Feret box oriented at 45°.
EDrawableFeature_FeretBox68	The Feret box oriented at 67.5°.
EDrawableFeature_GravityCenter	The gravity center.
EDrawableFeature_ MinimumEnclosingRectangle	The minimum enclosing rectangle.
EDrawableFeature_FeretBox	

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.	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.

6.16. EDrawingMode Enum

Allowed modes to draw the bounding box of a symbol.

Namespace: Euresys::Open_eVision_2_16

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.

6.17. EHarrisThresholdingMode Enum

The thresholding modes for the Harris corner detector.

Namespace: Euresys::Open_eVision_2_16

EHarrisThresholdingMode_Relative	Relative thresholding mode.
EHarrisThresholdingMode_ Absolute	Absolute thresholding mode.

6.18. EHistogramFeature Enum

The various parameters that can be extracted from a histogram.

Namespace: Euresys::Open_eVision_2_16

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.

6.19. EMatchingMode Enum

Allowed values for the matching mode of EasyOCR.

Namespace: Euresys::Open_eVision_2_16

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.

6.20. EMatrixCodeContrastMode Enum

-

Namespace: Euresys::Open_eVision_2_16

EMatrixCodeContrastMode_ BlackOnWhite	Dark cells on a light background.
EMatrixCodeContrastMode_ WhiteOnBlack	Light cells on a dark background.

6.21. EMaximumAnalysisMode Enum

This enumeration contains the possible values for the analysis mode of the ELaserLineExtractor object.

Namespace: Euresys::Open_eVision_2_16::Easy3D

EMaximumAnalysisMode_Peaks	Peak analysis mode.
EMaximumAnalysisMode_Max	Maximum analysis mode.
EMaximumAnalysisMode_COG	Center of gravity analysis mode.

6.22. EOCR2DetectionMethod Enum

This enumeration contains the possible methods for text detection in EOCR2.

Namespace: Euresys::Open_eVision_2_16

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.

6.23. EOCR2SegmentationMethod Enum

This enumeration contains the possible methods for image segmentation in EOCR2.

Namespace: Euresys::Open_eVision_2_16

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.

24. 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_2_16

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.

6.25. 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_2_16

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.

6.26. 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_2_16

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).

6.27. EQRCodeModel Enum

This enumeration contains the possible values for a QR code model. Used in EQRCode and EQRCodeReader.

Namespace: Euresys::Open_eVision_2_16

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.

6.28. EQRCodeScanPrecision Enum

This enumeration contains the possible values for the scanning precision used by an EQRCodeReader object.

Namespace: Euresys::Open_eVision_2_16

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.