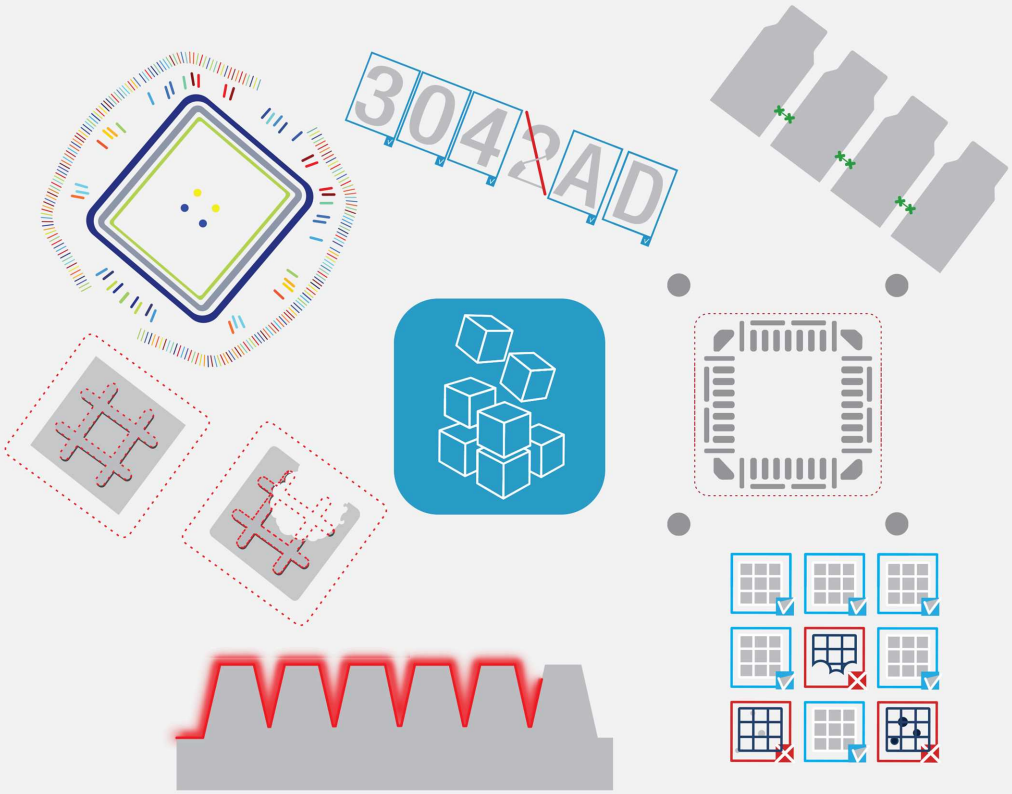
euresys
Empowering Computer Vision

CODE SNIPPETS

Open eVision



This documentation is provided with Open eVision 2.16.1 (doc build 1156).
www.euresys.com

Contents

1. Basic Types 6
1.1. Loading and Saving Images 6
1.2. Interfacing Third-Party Images 6
1.3. Retrieving Pixel Values 7
1.4. Importing Bitmap from the Resources 7
1.5. ROI Placement 8
1.6. Vector Management 8
1.7. Exception Management 8
2. ERegion 10
2.1. Basic Usage 10
2.2. Prepare Once, Use Multiple Times 10
2.3. Combine Regions 11
2.4. Tool Chain 11
3. EGrabberBridge 12
3.1. Using EGrabberBridge 12
3.2. Using EGrabberBridge with Format Conversion 12
3.3. Managing EGrabber Parameters 13
4. EasyImage 15
4.1. Thresholding 15
Single Thresholding 15
Double Thresholding 15
Histogram-Based Single Thresholding 16
Histogram-Based Double Thresholding 16
4.2. Arithmetic and Logic Operations 17
4.3. Convolution 17
Pre-Defined Kernel Filtering 17
User-Defined Kernel Filtering 18
4.4. Non-Linear Filtering 18
Morphological Filtering 18
Hit-and-Miss Transform 19
4.5. Vector Operations 20
Path Sampling 20
Profile Sampling 20
4.6. Statistics 21
Image Statistics 21
Sliding Windows Statistics 21
Histogram-Based Statistics 22
4.7. Noise Reduction by Integration 22
Temporal Noise Reduction 22
Recursive Average 23
4.8. Feature Point Detectors 23
Harris Corner Detector 23
Canny Edge Detector 24
4.9. Using Flexible Masks 24
5. EasyColor 26
5.1. Colorimetric Systems Conversion 26
5.2. Color Components 26
5.3. White Balance 27
5.4. Pseudo-Coloring 27
5.5. Bayer Pattern Decoding 28
6. Deep Learning Tools 29
6.1. Creating a Dataset and Training a Classifier 29

6.2. Loading a Classifier and Classifying a New Image 29
6.3. Using Multithreading for Classification 30
6.4. Loading an Unsupervised Segmenter and Segmenting an Image 31
7. EasyObject 33
7.1. Constructing the Blobs 33
Image Encoder 33
Image Segmenter 33
Holes Extraction 34
Continuous Mode 35
7.2. Computing Blobs Features 35
7.3. Selecting and Sorting Blobs 36
7.4. Using Flexible Masks 37
Constructing Blobs 37
Generating a Flexible Mask from an Encoded Image 37
Generating a Flexible Mask from a Blob Selection 38
7.5. Using the Object Template Matcher 38
8. EasyMatch 40
8.1. Pattern Learning 40
8.2. Setting Search Parameters 40
8.3. Pattern Matching and Retrieving Results 41
9. EasyFind 42
9.1. Pattern Learning 42
9.2. Setting Search Parameters 42
9.3. Pattern Finding and Retrieving Results 43
10. EasyGauge 44
10.1. Point Location 44
10.2. Line Fitting 44
10.3. Circle Fitting 45
10.4. Rectangle Fitting 46
10.5. Wedge Fitting 46
10.6. Gauge Grouping 47
Gauge Hierarchy 47
Complex Measurement 47
10.7. Calibration using EWorldShape 48
Calibration by Guesswork 48
Landmark-Based Calibration 49
Dot Grid-Based Calibration 49
Coordinates Transform 50
Image Unwarping 50
11. EasyOCR 52
11.1. Learning Characters 52
11.2. Recognizing Characters 52
12. EasyOCR2 54
12.1. Detecting Characters 54
12.2. Learning Characters 55
12.3. Reading Characters 56
Reading Using TrueType Fonts 56
Reading Using EOCR2 Character Database 56
Reading Using EOCR2 Model File 57
12.4. View Bitmap 58
13. EasyBarCode 59
13.1. Reading a Bar Code 59
13.2. Reading a Bar Code Following a Given Symbology 59
13.3. Reading a Bar Code of Known Location 60

13.4. Reading a Mail Bar Code 60
14. EasyMatrixCode 62
14.1. Automatic Reading 62
14.2. Reading with Prior Learning 62
14.3. Advanced Tuning of the Search Parameters 63
14.4. Retrieving Print Quality Grading 64
15. EasyMatrixCode2 65
15.1. Reading Matrix Codes from an Image 65
15.2. Reading with Prior Learning 65
15.3. Inspecting Print Quality Grades 66
15.4. Asynchronous Processing 66
16. EasyQRCode 68
16.1. Automatic Reading of a QR Code 68
16.2. Retrieving Information of a QR Code 68
16.3. Tuning the Search Parameters 69
16.4. Retrieving the Decoded String (Simple) 69
16.5. Retrieving the Decoded String (Safe) 70
16.6. Retrieving the Decoded Data (Advanced) 70
17. Easy3DObject 72
17.1. Extracting 3D Objects with a Selection Criterion 72
17.2. Inspecting a Feature from the List of E3DObjects 72
17.3. Drawing a 2D Feature from the List of E3DObjects 72
17.4. Drawing 3D Features from a List of E3DObjects 73
18. Easy3DMatch 74
18.1. E3DAligner Minimal Code 74
18.2. E3DAligner Reprojection Plane 74
18.3. E3DAlignment Align Sample 75
18.4. E3DComparer Minimal Sample 75
18.5. E3DComparer Advanced Sample 76
18.6. E3DMatcher Minimal Sample 76
18.7. E3DMatcher Advanced Sample 77

1. Basic Types

1.1. Loading and Saving Images

Functional Guide | Reference: Load, Save, SaveJpeg

```
////////////////////////////////////////////////////////////
// This code snippet shows how to load and save an image. //
////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// Load an image file
srcImage.Load("mySourceImage.bmp");

// ...

// Save the destination image into a file
dstImage.Save("myDestImage.bmp");

// Save the destination image into a jpeg file
// The default compression quality is 75
dstImage.Save("myDestImage.jpg");

// Save the destination image into a jpeg file
// set the compression quality to 50
dstImage.SaveJpeg("myDestImage50.jpg", 50);
```

1.2. Interfacing Third-Party Images

Functional Guide | Reference: SetImagePtr

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to link an Open eVision image //
// to an externally allocated buffer.                        //
////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();

// Size of the third-party image
int sizeX = bufferSizeX;
int sizeY = bufferSizeY;

//Pointer to the third-party image buffer
IntPtr imgPtr = bufferPointer;

// ...

// Link the Open eVision image to the third-party image
// Assuming the corresponding  buffer is aligned on 4 bytes
```

```
srcImage.SetImagePtr(sizeX, sizeY, imgPtr);
```

1.3. Retrieving Pixel Values

Functional Guide | Reference: GetImagePtr

```
//////////////////////////////////////////////////////////
// This code snippet shows the recommended method to access  //
// the pixel values in a BW8 image.                          //
//////////////////////////////////////////////////////////

using System.RunTime.InteropServices;

IntPtr pixAddr;
byte pix;

//...

for(int y = 0; y < height; ++y)
{
    pixAddr = bw8Image.GetImagePtr(0,y);
    for(int x = 0; x < width; ++x)
        pix = Marshal.ReadByte(pixAddr,x);
}
```

1.4. Importing Bitmap from the Resources

Functional Guide | Reference: SetImagePtr

```
//////////////////////////////////////////////////////////
// This code snippet shows how to import a bitmap from       //
// the resources.                                            //
//////////////////////////////////////////////////////////

// Use this function to fasten the copy of the bitmap
[DllImport("kernel32.dll", EntryPoint = "CopyMemory", SetLastError = false)]
public static extern void CopyMemory(IntPtr dest, IntPtr src, uint count);

// Get the bitmap
Bitmap bitmap = new Bitmap(WindowsFormsApp►.Properties.Resources.Image►);

int width = bitmap.Width;
int height = bitmap.Height;

BitmapData bmd = bitmap.LockBits(new Rectangle(0, 0, width, height), ImageLockMode.ReadOnly, bitmap.PixelFormat);

EImageC24 image = new EImageC24(width, height);

// Number of bytes per row to copy
uint strideWidth = (uint)(3 * width);

for (int y = 0; y < height; ++y)
{
  CopyMemory(image.GetImagePtr(0, y), bmd.Scan0 + y * bmd.Stride, strideWidth);
}

bitmap.UnlockBits(bmd);
bitmap.Dispose();
```

```
// Process the image

image.Dispose();
```

1.5. ROI Placement

Functional Guide | Reference: Attach, SetPlacement

```
////////////////////////////////////////////////////////////
// This code snippet shows how to attach an ROI to an image  //
// and set its placement.                                    //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 parentImage= new EImageBW8();

// ROI constructor
EROIBW8 myROI= new EROIBW8();

// Attach the ROI to the image
myROI.Attach(parentImage);

//Set the ROI position
myROI.SetPlacement(50, 50, 200, ↵00);
```

1.6. Vector Management

Functional Guide | Reference: Empty, AddElement

```
////////////////////////////////////////////////////////////
// This code snippet shows how to create a vector, fill it   //
// and retrieve the value of a given element.                //
////////////////////////////////////////////////////////////

// EBW8Vector constructor
EBW8Vector ramp= new EBW8Vector();
EBW8 bw8 = new EBW8();

// Clear the vector
ramp.Empty();

// Fill the vector with increasing values
for(int i= 0; i < ↵28; i++)
{
    bw8.Value = (byte)i;
    ramp.AddElement(bw8);
}

// Retrieve the ↵0th element value
EBW8 value = ramp.GetElement(9);
```

1.7. Exception Management

Functional Guide | Reference: GetPixel, What

```
//////////////////////////////////////////
// This code snippet shows how to manage  //
// Open eVision exceptions.               //
//////////////////////////////////////////

try
{
    // Image constructor
    EImageC24 srcImage= new EImageC24();

    // ...

    // Retrieve the pixel value at coordinates (56, 73)
    EC24 value= srcImage.GetPixel(56, 73);
}

catch(EException exc)
{
    // Retrieve the exception description
    string error = exc.What();
}
```

2. ERegion

See also: Arbitrary-Shaped ROI (ERegion) / example: Inspecting Pads Using Regions

2.1. Basic Usage

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a threshold on a //
// circular region in an image.                            //
////////////////////////////////////////////////////////////

// Image constructors
EImageBW8 srcImage = new EImageBW8();
EImageBW8 dstImage = new EImageBW8();

//...

// Create the region
ECircleRegion circleRegion = new ECircleRegion(center, radius);

// Threshold the image
EasyImage.Threshold(srcImage, circleRegion, dstImage);
```

2.2. Prepare Once, Use Multiple Times

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a threshold on a //
// circular region in multiple image while preparing it    //
// only once.                                              //
////////////////////////////////////////////////////////////

// Image constructors
EImageBW8 [] srcImage = new EImageBW8[↵0];
EImageBW8 [] dstImage = new EImageBW8[↵0];

//...

// Create the region
ECircleRegion circleRegion = new ECircleRegion(center, radius);

// Prepare the region
circleRegion.Prepare(srcImage[0]);

// Threshold the images
for (int i = 0; i < ↵0; i++)
  EasyImage.Threshold(srcImage[i], circleRegion, dstImage[i]);
```

2.3. Combine Regions

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a threshold on a //
// combined region in an image                             //
////////////////////////////////////////////////////////////

// Image constructors
EImageBW8 srcImage = new EImageBW8();
EImageBW8 dstImage = new EImageBW8();

//...

// Create first region
ECircleRegion circleRegion = new ECircleRegion(center, radius);

// Create second region
ERectangleRegion rectangleRegion = new ERectangleRegion(center, width, height, angle);

// Combine regions
ERegion combinedRegion = ERegion.Union(circleRegion, rectangleRegion);

// Threshold the image
EasyImage.Threshold(srcImage, combinedRegion, dstImage);
```

2.4. Tool Chain

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a threshold on a //
// region coming for a previous EasyFind process           //
////////////////////////////////////////////////////////////

// Image constructors
EImageBW8 findImage = new EImageBW8();
EImageBW8 srcImage = new EImageBW8();
EImageBW8 dstImage = new EImageBW8();

// EPatternFinder constructor
EPatternFinder finder = new EPatternFinder();

//...

// Use EasyFind
EFoundPattern [] patterns = finder.Find(findImage);

// Create region from found pattern
ERegion foundRegion = new ERegion(patterns[0]);

// Threshold the image
EasyImage.Threshold(srcImage, foundRegion, dstImage);
```

3. EGrabberBridge

See also: EGrabberBridge - Using Images from Coaxlink

3.1. Using EGrabberBridge

```
//////////////////////////////////////////////////////////////////////
// This code snippet shows how to go from an EGrabber buffer to an //
// EGrabberImageBW8, compatible with Open eVision processing       //
//////////////////////////////////////////////////////////////////////

// Construct the EGrabber objects.
// The FormatConverter is optional and will automatically convert the EGenTL buffer to
// the chosen Open eVision image type.
// WARNING: EGrabberCallbackOnDemand and FormatConverter are using an EGenTL instance,
// you must dispose them before disposing it.
EGenTL genTL = new EGenTL();
EGrabberCallbackOnDemand grabber = new EGrabberCallbackOnDemand(genTL);

// Allocate one buffer
grabber.reallocBuffers(1);

//....

// Start the grabber acquisition of one buffer
grabber.start(1);

// Get the acquired buffer
using (ScopedBuffer buffer = new ScopedBuffer(grabber))

{
  // Convert the ScopedBuffer to an Open eVision data container
  using (EGrabberImageBW8 image = new EGrabberImageBW8(buffer.getInfo()))
  {
    // Use the EGrabberImageBW8 as an Open eVision EImage Object
    // Here an inversion to the image is performed
    EImageBW8 invertedImage = new EImageBW8(image.Width, image.Height);
    EasyImage.Oper(EArithmeticLogicOperation.Invert, image, invertedImage);
  }
}
```

3.2. Using EGrabberBridge with Format Conversion

```
//////////////////////////////////////////////////////////////////////
// This code snippet shows how to go from an EGrabber buffer to an //
// EGrabberImageBW8, compatible with Open eVision processing using //
// format conversion                                               //
//////////////////////////////////////////////////////////////////////

// Construct the EGrabber objects.
// The FormatConverter is optional and will automatically convert the EGenTL buffer to
```

```
// the chosen Open eVision image type.
// WARNING: EGrabberCallbackOnDemand and FormatConverter are using an EGenTL instance,
// you must dispose them before disposing it.
EGenTL genTL = new EGenTL();
EGrabberCallbackOnDemand grabber = new EGrabberCallbackOnDemand(genTL);
FormatConverter converter = new FormatConverter(genTL);

// Allocate one buffer
grabber.reallocBuffers(1);

//....

// Start the grabber acquisition of one buffer
grabber.start(1);

// Get the acquired buffer
using (ScopedBuffer buffer = new ScopedBuffer(grabber))
{
  // Convert the ScopedBuffer to an Open eVision data container
  using (EGrabberImageBW8 image = new EGrabberImageBW8(converter, buffer.getInfo()))
  {
    // Use the EGrabberImageBW8 as an Open eVision EImage Object
    // Here an inversion to the image is performed
    EImageBW8 invertedImage = new EImageBW8(image.Width, image.Height);
    EasyImage.Oper(EArithmeticLogicOperation.Invert, image, invertedImage);
  }
}
```

3.3. Managing EGrabber Parameters

```
////////////////////////////////////////////////////////////////////
// This code snippet shows how to go from an EGrabber buffer to an //
// EGrabberImageBW8, compatible with Open eVision processing using //
// format conversion                                               //
////////////////////////////////////////////////////////////////////

// Construct the EGrabber objects.
// The FormatConverter is optional and will automatically convert the EGenTL buffer to
// the chosen Open eVision image type.
// WARNING: EGrabberCallbackOnDemand and FormatConverter are using an EGenTL instance,
// you must dispose them before disposing it.
EGenTL genTL = new EGenTL();
EGrabberCallbackOnDemand grabber = new EGrabberCallbackOnDemand(genTL);
FormatConverter converter = new FormatConverter(genTL);

// Allocate one buffer
grabber.reallocBuffers(1);

// ...

// Manage EGrabber features
// Get/set camera (RemoteModule) features of various types:
// string - integer - float.
// WARNING: The features might be specific to each camera.
string pixelFormat = grabber.getStringRemoteModule("PixelFormat");
grabber.setStringRemoteModule("PixelFormat", "Mono8");

int width = (int) grabber.getIntegerRemoteModule("Width");
grabber.setIntegerRemoteModule("Width", 1024);

float exposureTime = (float) grabber.getFloatRemoteModule("ExposureTime");
```

```
grabber.setFloatRemoteModule("ExposureTime", 60.0f);

// ...

// Start the grabber acquisition of one buffer
grabber.start(1);

// Get the acquired buffer
using (ScopedBuffer buffer = new ScopedBuffer(grabber))
{
  // Convert the ScopedBuffer to an Open eVision data container
  using (EGrabberImageBW8 image = new EGrabberImageBW8(converter, buffer.getInfo()))
  {
    // ...
  }
}
```

4. EasyImage

4.1. Thresholding

Single Thresholding

Functional Guide | Reference: SetSize, Threshold

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to perform minimum residue     //
// thresholding, absolute thresholding and relative           //
// thresholding operations.                                   //
////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Minimum residue thresholding (default method)
EasyImage.Threshold(srcImage, dstImage);

// Absolute thresholding (threshold = ↦0)
EasyImage.Threshold(srcImage, dstImage, ↦0);

// Relative thresholding (70% black, 30% white)
EasyImage.Threshold(srcImage, dstImage, unchecked((uint)EThresholdMode.Relative), 0, 255, 0.7f);
```

Double Thresholding

Functional Guide | Reference: DoubleThreshold

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to perform a thresholding      //
// operation based on low and high threshold values.          //
////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Double thresholding, low threshold = 50, high threshold = ↦50,
// pixels below 50 become black, pixels above ↦50 become white,
// pixels between thresholds become gray
```

```
EasyImage.DoubleThreshold(srcImage, dstImage, 50, ⬅50, 0, ⬅28, 255);
```

Histogram-Based Single Thresholding

Functional Guide | Reference: Histogram, HistogramThreshold

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to perform a minimum residue   //
// thresholding operation based on an histogram.              //
////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// Histogram constructor
EBWHistogramVector histo= new EBWHistogramVector();

// Variables
uint thresholdValue= , unchecked((uint)EThresholdMode.MinResidue);
float avgBelowThr, avgAboveThr;

// ...

// Compute the histogram
EasyImage.Histogram(srcImage, histo);

// Compute the single threshold (and the average pixel values below and above the threshold)
EasyImage.HistogramThreshold(histo, ref thresholdValue, out avgBelowThr, out avgAboveThr);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform the single thresholding
EasyImage.Threshold(srcImage, dstImage, thresholdValue);
```

Histogram-Based Double Thresholding

Functional Guide | Reference: Histogram, ThreeLevelsMinResidueThreshold, DoubleThreshold

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to perform a double thresholding   //
// operation. The low and high threshold values are computed      //
// according to the minimum residue method based on an histogram. //
////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// Histogram constructor
EBWHistogramVector histo= new EBWHistogramVector();

// Variables
EBW8 lowThr= new EBW8();
EBW8 highThr= new EBW8();
float avgBelowThr, avgBetweenThr, avgAboveThr;

// ...

// Compute the histogram
```

```
EasyImage.Histogram(srcImage, histo);

// Compute the low and high threshold values automatically
// (and the average pixel values below, between and above the threshold)
EasyImage.ThreeLevelsMinResidueThreshold(histo, out lowThr, out highThr, out avgBelowThr, out avgBetweenThr, out
avgAboveThr);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform the double thresholding
EasyImage.DoubleThreshold(srcImage, dstImage, lowThr.Value , highThr.Value);
```

4.2. Arithmetic and Logic Operations

Functional Guide | Reference: Oper

```
//////////////////////////////////////////////////////
// This code snippet shows how to apply miscellaneous //
// arithmetic and logic operations to images.         //
//////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcGray0= new EImageBW8();
EImageBW8 srcGray1= new EImageBW8();
EImageBW8 dstGray= new EImageBW8();
EImageC24 srcColor= new EImageC24();
EImageC24 dstColor= new EImageC24();

EBW8 bw8Constant = new EBW8(2);
EC24 c24Constant = new EC24(28, 64, 96);

// ...

// All images must have the same size
dstGray.SetSize(srcGray0);
dstColor.SetSize(srcColor);

// Subtract srcGray1 from srcGray0
EasyImage.Oper(EArithmeticLogicOperation.Subtract, srcGray0, srcGray1, dstGray);

// Multiply srcGray0 by a constant value
EasyImage.Oper(EArithmeticLogicOperation.Multiply, srcGray0, bw8Constant, dstGray);

// Add a constant value to srcColor
EasyImage.Oper(EArithmeticLogicOperation.Add, srcColor, c24Constant, dstColor);

// Erase (blacken) the destination image where the source image is black
bw8Constant.Value = (byte)0;
EasyImage.Oper(EArithmeticLogicOperation.SetZero, srcGray0, bw8Constant, dstGray);
```

4.3. Convolution

Pre-Defined Kernel Filtering

```
//////////////////////////////////////////////////////
// This code snippet shows how to apply miscellaneous    //
```

```
// convolution operations based on pre-defined kernels.  //
//////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform a Uniform filtering (5x5 kernel)
EasyImage.ConvolUniform(srcImage, dstImage, 2);

// Perform a Highpass filtering
EasyImage.ConvolHighpass►(srcImage, dstImage);

// Perform a Gradient filtering
EasyImage.ConvolGradient(srcImage, dstImage);

// Perform a Sobel filtering
EasyImage.ConvolSobel(srcImage, dstImage);
```

User-Defined Kernel Filtering

```
//////////////////////////////////////////////////////
// This code snippet shows how to apply a convolution  //
// operation based on a user-defined kernel.           //
//////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Create and define a user-defined kernel
// (Frei-Chen row gradient, positive only)
EKernel kernel= new EKernel();
kernel.SetKernelData(0.2929f, 0, -0.2929f,
                     0.4►42f, 0, -0.4►42f,
                     0.2929f, 0, -0.2929f);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Apply the convolution kernel
EasyImage.ConvolKernel(srcImage, dstImage, kernel);
```

4.4. Non-Linear Filtering

Functional Guide | Reference

Morphological Filtering

Functional Guide | Reference: ErodeBox, DilateBox, OpenDisk

```
//////////////////////////////////////////////////////
// This code snippet shows how to apply miscellaneous  //
// morphological filtering operations.                 //
//////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform an erosion (3x3 square kernel)
EasyImage.ErodeBox(srcImage, dstImage, ⌙);

// Perform a dilation (5x3 rectangular kernel)
EasyImage.DilateBox(srcImage, dstImage, 2, ⌙);

// Perform an Open operation (5x5 circular kernel)
EasyImage.OpenDisk(srcImage, dstImage, 2);
```

Hit-and-Miss Transform

Functional Guide | Reference: SetValue, HitAndMiss

```
//////////////////////////////////////////////////////////
// This code snippet shows how to highlight the left corner //
// of a rhombus by means of a Hit-and-Miss operation.       //
//////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Create and define a Hit-and-Miss kernel
// corresponding to the left corner of a rhombus
EHitAndMissKernel leftCorner= new EHitAndMissKernel(-⌙, -⌙, ⌙, ⌙);

// Left column of the kernel
leftCorner.SetValue(-⌙, 0, EHitAndMissValue.Background);

// Middle column of the kernel
leftCorner.SetValue(0, -⌙, EHitAndMissValue.Background);
leftCorner.SetValue(0, 0, EHitAndMissValue.Foreground);
leftCorner.SetValue(0, ⌙, EHitAndMissValue.Background);

// Right column of the kernel
leftCorner.SetValue(⌙, -⌙, EHitAndMissValue.Foreground);
leftCorner.SetValue(⌙, 0, EHitAndMissValue.Foreground);
leftCorner.SetValue(⌙, ⌙, EHitAndMissValue.Foreground);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Apply the Hit-and-Miss kernel
EasyImage.HitAndMiss(srcImage, dstImage, leftCorner);
```

4.5. Vector Operations

Functional Guide | Reference

Path Sampling

Functional Guide | Reference: Empty, AddElement, ImageToPath

```
////////////////////////////////////////////////////////////
// This code snippet shows how to retrieve and store the    //
// pixel values along a given path together with the        //
// corresponding pixel coordinates.                         //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ...

// Vector constructor
EBW8PathVector path= new EBW8PathVector();
EBW8 bw8= new EBW8(►28);

// Path definition
path.Empty();
for (int i = 0; i < ►00; i++)
{
  EBW8Path p;
  p.X = (short)i;
  p.Y = (short)i;
  p.Pixel = bw8;
  path.AddElement(p);
}

// Get the image data along the path
EasyImage.ImageToPath(srcImage, path);
int pixel = path.GetElement(20).Pixel.UINT32Value;
```

Profile Sampling

Functional Guide | Reference: ImageToLineSegment, LineSegmentToImage

```
////////////////////////////////////////////////////////////
// This code snippet shows how to set, retrieve and store   //
// the pixel values along a given line segment.             //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ...

// Vector constructor
EBW8Vector profile= new EBW8Vector();

// Get the image data along segment (►0,5►2)-(500,40)
EasyImage.ImageToLineSegment(srcImage, profile, ►0, 5►2, 500, 40);
```

```
// Set all these points to white (255) in the image
EBW8 white = new EBW8(255);
EasyImage.LineSegmentToImage(srcImage, white, ↵0, 5↵2, 500, 40);
```

4.6. Statistics

Image Statistics

```
////////////////////////////////////////////////////////////////////
// This code snippet shows how to compute basic image statistics. //
////////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ...

// Count the number of pixels above the threshold (↵28)
int count;
EBW8 threshold = new EBW8(↵28);
EasyImage.Area(srcImage, threshold, out count);

// Compute the pixels' average and standard deviation values
float stdDev, average;
EasyImage.PixelStdDev(srcImage, out stdDev, out average);

// Compute the image gravity center (pixels above threshold)
float x, y;
EasyImage.GravityCenter(srcImage, ↵28, out x, out y);
```

Sliding Windows Statistics

Functional Guide | Reference: LocalAverage, LocalDeviation

```
////////////////////////////////////////////////////////////////////////
// This code snippet shows how to perform sliding windows statistics. //
////////////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage0= new EImageBW8();
EImageBW8 dstImage↵= new EImageBW8();

// ...

// All images must have the same size
dstImage0.SetSize(srcImage);
dstImage↵.SetSize(srcImage);

// Local average in a ↵↵x↵↵ window
EasyImage.LocalAverage(srcImage, dstImage0, 5, 5);

// Local deviation in a ↵↵x↵↵ window
EasyImage.LocalDeviation(srcImage, dstImage↵, 5, 5);
```

Histogram-Based Statistics

Functional Guide | Reference: Histogram, AnalyseHistogram

```
////////////////////////////////////////////////////
// This code snippet shows how to compute statistics  //
// based on an histogram.                             //
////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ...

// Histogram constructor
EBWHistogramVector histo= new EBWHistogramVector();

// Compute the histogram
EasyImage.Histogram(srcImage, histo);

// Compute the average gray-level value
float average = EasyImage.AnalyseHistogram(histo, EHistogramFeature.AveragePixelValue, 0, 255);

// Compute the gray-level standard deviation
float deviation = EasyImage.AnalyseHistogram(histo, EHistogramFeature.PixelValueStdDev, 0, 255);
```

4.7. Noise Reduction by Integration

Functional Guide | Reference

Temporal Noise Reduction

Functional Guide | Reference: Oper

```
////////////////////////////////////////////////
// This code snippet shows how to perform noise  //
// reduction by temporal averaging.              //
////////////////////////////////////////////////

// Images constructor
EImageBW►6 noisyImage= new EImageBW►6();
EImageBW►6 cleanImage= new EImageBW►6();

// ►6 bits work image used as an accumulator
EImageBW►6 store= new EImageBW►6();

// ...

// All images must have the same size
cleanImage.SetSize(noisyImage);
store.SetSize(noisyImage);

// Clear the accumulator image
EBW►6 bw►6= new EBW►6(0);
EasyImage.Oper(EArithmeticLogicOperation.Copy, bw►6, store);

// Accumulation loop
int n;
```

```
for (n = 0; n < ►0; n++)
{
    // Acquire a new image into noisyImage
    // ...

    // Add this new noisy image into the accumulator
    EasyImage.Oper(EArithmeticLogicOperation.Add, noisyImage, store, store);
}

// Perform noise reduction
bw►6.Value= (byte)n;
EasyImage.Oper(EArithmeticLogicOperation.Divide, store, bw►6, cleanImage);
```

Recursive Average

Functional Guide | Reference: Oper, SetRecursiveAverageLUT, RecursiveAverage

```
////////////////////////////////////////////////
// This code snippet shows how to perform noise  //
// reduction by recursive averaging.             //
////////////////////////////////////////////////

// Images constructor
EImageBW8 noisyImage= new EImageBW8();
EImageBW8 cleanImage= new EImageBW8();

// ►6 bits work image used as an accumulator
EImageBW►6 store= new EImageBW►6();

// ...

// All images must have the same size
cleanImage.SetSize(noisyImage);
store.SetSize(noisyImage);

// Clear the accumulator image
EBW►6 bw►6= new EBW►6(0);
EasyImage.Oper(EArithmeticLogicOperation.Copy, bw►6, store);

// Prepare the transfer lookup table (reduction factor = 3)
EBW►6Vector lut= new EBW►6Vector();
EasyImage.SetRecursiveAverageLUT(lut, 3.0f);

// Perform the noise reduction
EasyImage.RecursiveAverage(noisyImage, store, cleanImage, lut);
```

4.8. Feature Point Detectors

Harris Corner Detector

Functional Guide | Reference: GetPointCount, GetPoint

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to retrieve corners' coordinates //
// by means of the Harris corner detector algorithm.           //
////////////////////////////////////////////////////////////////

// Image constructor
```

```
EImageBW8 srcImage= new EImageBW8();

// ...

// Harris corner detector
EHarrisCornerDetector harris= new EHarrisCornerDetector();
EHarrisInterestPoints interestPoints= new EHarrisInterestPoints();
harris.IntegrationScale= 2.0f;

// Perform the corner detection
harris.Apply(srcImage, interestPoints);

// Retrieve the number of corners
uint index = interestPoints.PointCount;

// Retrieve the first corner coordinates
EPoint point = interestPoints.GetPoint(0);
float x = point.X;
float y = point.Y;
```

Canny Edge Detector

Functional Guide | Reference: Apply

```
////////////////////////////////////////////////////
// This code snippet shows how to highlight edges  //
// by means of the Canny edge detector algorithm.  //
////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// ...

// Canny edge detector
ECannyEdgeDetector canny= new ECannyEdgeDetector();

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform the edges detection
canny.Apply(srcImage, dstImage);
```

4.9. Using Flexible Masks

Functional Guide | Reference

Computing Pixels Average

Functional Guide | Reference: PixelAverage

```
////////////////////////////////////////////////////
// This code snippet shows how to compute statistics   //
// inside a region defined by a flexible mask.         //
////////////////////////////////////////////////////

// Images constructor
```

```
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask= new EImageBW8();

// ...

// Compute the average value of the source image pixels
// corresponding to the mask do-care areas only
float average;
EasyImage.PixelAverage(srcImage, mask, out average);
```

5. EasyColor

5.1. Colorimetric Systems Conversion

Functional Guide | Reference: ConvertFromRgb, Transform

```
//////////////////////////////////////////////////////
// This code snippet shows how to convert a color image //
// from the RGB to the Lab colorimetric system.         //
//////////////////////////////////////////////////////

// Images constructor
EImageC24 srcImage= new EImageC24();
EImageC24 dstImage= new EImageC24();

// ...

// Prepare a lookup table for
// the RGB to La*b* conversion
EColorLookup lookup= new EColorLookup();
lookup.ConvertFromRgb(EColorSystem.Lab);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform the color conversion
EasyColor.Transform(srcImage, dstImage, lookup);
```

5.2. Color Components

Functional Guide | Reference: Compose, ConvertFromRgb, GetComponent

```
//////////////////////////////////////////////////////
// This code snippet shows how to create a color image  //
// from 3 grayscale images and extract the luminance    //
// component from a color image.                        //
//////////////////////////////////////////////////////

// Images constructor
EImageBW8 red= new EImageBW8();
EImageBW8 green= new EImageBW8();
EImageBW8 blue= new EImageBW8();
EImageC24 colorImage= new EImageC24();
EImageBW8 luminance= new EImageBW8();

// ...

// Source and destination images must have the same size
colorImage.SetSize(red);

// Combine the color planes into a color image
EasyColor.Compose(red, green, blue, colorImage);

// Prepare a lookup table for
```

```
// the RGB to LSH conversion
EColorLookup lookup= new EColorLookup();
lookup.ConvertFromRgb(EColorSystem.Lsh);

// Source and destination images must have the same size
luminance.SetSize(colorImage);

// Get the Luminance component
EasyColor.GetComponent(colorImage, luminance, 0, lookup);
```

5.3. White Balance

Functional Guide | Reference: PixelAverage, WhiteBalance, Transform

```
////////////////////////////////////////////////////////
// This code snippet shows how to perform white balancing. //
////////////////////////////////////////////////////////

// Images constructor
EImageC24 srcImage= new EImageC24();
EImageC24 dstImage= new EImageC24();
EImageC24 whiteRef= new EImageC24();

// ...

// Create a lookup table
EColorLookup lut= new EColorLookup();

// Measure the calibration values from a white reference image
float r, g, b;
EasyImage.PixelAverage(whiteRef, out r, out g, out b);

// Prepare the lookup table for
// a white balance operation
lut.WhiteBalance(►.00f, EasyColor.CompensateNtscGamma, r, g, b);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Perform the white balance operation
lut.Transform(srcImage, dstImage);
```

5.4. Pseudo-Coloring

Functional Guide | Reference: SetShading, PseudoColor

```
////////////////////////////////////////////////////////
// This code snippet shows how to perform pseudo-coloring. //
////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageC24 dstImage= new EImageC24();

// ...

// Create a pseudo-color lookup table
EPseudoColorLookup pcLut= new EPseudoColorLookup();
```

```
// Define a shade of pure tints, from red to blue
EC24 red= new EC24(255, 0, 0);
EC24 blue= new EC24(0, 0, 255);
pcLut.SetShading(red, blue, EColorSystem.Ish);

// Source and destination images must have the same size
dstImage.SetSize(srcImage);

// Generate the pseudo-colored image
EasyColor.PseudoColor(srcImage, dstImage, pcLut);
```

5.5. Bayer Pattern Decoding

Functional Guide | Reference: BayerToC24

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to perform Bayer pattern decoding. //
////////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 bayerImage= new EImageBW8();
EImageC24 dstImage= new EImageC24();

// ...

// Source and destination images must have the same size
dstImage.SetSize(bayerImage);

// Convert to true color with simple interpolation, default parity assumed
EasyColor.BayerToC24(bayerImage, dstImage);
```

6. Deep Learning Tools

6.1. Creating a Dataset and Training a Classifier

```
////////////////////////////////////////////////////////////
// This code snippet shows how to create a dataset, train a //
// classifier and get the best performance metrics obtained //
// during the training.                                     //
////////////////////////////////////////////////////////////

// Creating dataset and classifier objects
EClassificationDataset dataset= new EClassificationDataset();
EClassificationDataset trainingDataset= new EClassificationDataset();
EClassificationDataset validationDataset= new EClassificationDataset();
EClassifier classifier= new EClassifier();

// Adding images using a glob pattern
dataset.AddImages("*good*.png", "good");
dataset.AddImages("*defective*.png", "defective");

// Enabling data augmentation on the dataset
dataset.EnableDataAugmentation= true;

// Rotation of up to 90°
dataset.MaxRotationAngle= 90.0F;

// Enabling horizontal flips
dataset.EnableHorizontalFlip= true;

// Splitting the dataset with 80% of images for the training dataset
// and 20% for the validation dataset
dataset.SplitDataset(trainingDataset, validationDataset, 0.8F);

// Training the classifier for 50 epochs
classifier.Train(trainingDataset, validationDataset, 50);
classifier.WaitForTrainingCompletion();

// Get the best metrics obtained on the validation dataset
EClassificationMetrics bestMetrics = classifier.GetValidationMetrics(classifier.BestEpoch);

// Dispose of objects
dataset.Dispose();
trainingDataset.Dispose();
validationDataset.Dispose();
classifier.Dispose();
```

6.2. Loading a Classifier and Classifying a New Image

```
////////////////////////////////////////////////////////////
// This code snippet shows how load a trained classifier and //
// classify a new image.                                    //
```

```
////////////////////////////////////////////////////////////

// Image and classifier constructor
EClassifier classifier= new EClassifier();
EImageBW8 srcImage= new EImageBW8();

// String and probability for the most probable result
string label;
float probability;

// Load classifier and image
classifier.Load(...);
srcImage.Load(...);

// Classify image
EClassificationResult result = classifier.Classify(srcImage);

// Get the most probable label
label = result.BestLabel;
probability = result.BestProbability;

// Dispose of objects
classifier.Dispose();
srcImage.Dispose();
```

6.3. Using Multithreading for Classification

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to parallelize the            //
// classification of new images on the CPU.                  //
// This code snippet requires the .NET Framework 4.0         //
////////////////////////////////////////////////////////////////

using System.Collections.Threading;
using System.Collections.Concurrent;

...

static void ClassificationLoop(Object obj)
{
       BlockingCollection<EImageC24> queue = obj as BlockingCollection<EImageC24>;

    EClassifier c = new EClassifier();
    c.Load("classifier.ecl");

    while (!queue.IsCompleted)
    {
        EImageC24 image = queue.Take();

        EClassificationResult result = c.Classify(image);
        // Get the most probable label
        string label = result.BestLabel;
        float probability = result.BestProbability;

        // Perform other actions based on the result
               ...
    }
}

...

int NUM_THREADS = 2;
```

```
// Queue holding the image to classify
BlockingCollection<EImageC24> imageQueue = new BlockingCollection<EImageC24>(new ConcurrentQueue<EImageC24>(), 2
* NUM_THREADS);

// Create and start the thread pool
Thread[] threads = new Thread[NUM_THREADS];
for (int i = 0; i < NUM_THREADS; i++)
{
    threads[i] = new Thread(ClassificationLoop);
    threads[i].Start(imageQueue);
}

bool hasImage = true;
while (hasImage)
{
       EImageC24 image = new EImageC24();

       // Load or set the data pointer of the image
       ...

       // Add the image to the queue
       imageQueue.Add(image);

       // Check that we still have an image to process and change the status
       // of "hasImage" if necessary.
       ...
}

// Tell the threads that they won't have any new image coming.
imageQueue.CompleteAdding();

// Wait for the threads to finish
for (int i = 0; i < NUM_THREADS; i++)
       threads[i].Join();
```

6.4. Loading an Unsupervised Segmenter and Segmenting an Image

```
////////////////////////////////////////////////////////////
// This code snippet shows how to load a trained          //
// unsupervised segmenter and how to segment a new image. //
////////////////////////////////////////////////////////////

// Image and segmenter constructor
EUnsupervisedSegmenter segmenter = new EUnsupervisedSegmenter();
EImageBW8 image = new EImageBW8();

// Load segmenter and image
segmenter.Load(...);
image.Load(...);

// Apply the segmenter on the image
EUnsupervisedSegmenterResult result = segmenter.Apply(image);

// Retrieve the segmentation map
EImageBW8 segmentationMap = result.SegmentationMap() ;

// Dispose of objects
```

```
segmenter.Dispose();
image.Dispose();
```

7. EasyObject

7.1. Constructing the Blobs

Image Encoder

Functional Guide | Reference: Encode, SetBlackLayerEncoded, SetWhiteLayerEncoded, SetMode, SetAbsoluteThreshold, GetGrayscaleSingleThresholdSegmenter

```
////////////////////////////////////////////////////////////
// This code snippet shows how to build blobs belonging to  //
// the white layer according to the minimum residue method  //
// and how to build blobs belonging to the black layer      //
// according to an absolute threshold.                      //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Build the blobs belonging to the white layer,
// the segmentation is based on the Minimum Residue method
encoder.Encode(srcImage, codedImage);

// Build the blobs belonging to the black layer,
// the segmentation is based on an absolute threshold (↵0)
Euresys.Open_eVision_↵_↵.Segmenters.EGrayscaleSingleThresholdSegmenter segmenter=
encoder.GrayscaleSingleThresholdSegmenter;
segmenter.BlackLayerEncoded= true;
segmenter.WhiteLayerEncoded= false;

segmenter.Mode= EGrayscaleSingleThreshold.Absolute;
segmenter.AbsoluteThreshold= ↵0;

encoder.Encode(srcImage, codedImage);
```

Image Segmenter

Functional Guide | Reference: SetSegmentationMethod, GetGrayscaleDoubleThresholdSegmenter, SetHighThreshold, SetLowThreshold

```
////////////////////////////////////////////////////////////
// This code snippet shows how to build blobs according to  //
// a user-defined image segmenter.                          //
////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Set the segmentation method to GrayscaleDoubleThreshold
encoder.SegmentationMethod= ESegmentationMethod.GrayscaleDoubleThreshold;

// Retrieve the segmenter object
Euresys.Open_eVision_►_►.Segmenters.EGrayscaleDoubleThresholdSegmenter segmenter=
encoder.GrayscaleDoubleThresholdSegmenter;

// Set the high and low threshold values
segmenter.HighThreshold= ►50;
segmenter.LowThreshold= 50;

// Specify the layers to be encoded (neutral layer only)
segmenter.BlackLayerEncoded= false;
segmenter.NeutralLayerEncoded= true;
segmenter.WhiteLayerEncoded= false;

// Encode the image
encoder.Encode(srcImage, codedImage);
```

Holes Extraction

Functional Guide | Reference: GetHoleCount, GetHole, GetObjCount, GetObj

```
////////////////////////////////////////////////////////////
// This code snippet shows how to retrieve blobs' holes. //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Encode the image
encoder.Encode(srcImage, codedImage);

// Retrieve holes for all the blobs
for (uint blobIndex = 0; blobIndex < codedImage.GetObjCount(); blobIndex++)
{
       EObject blob = codedImage.GetObj(blobIndex);

       // Browse the holes of the current object
       for (uint holeIndex = 0; holeIndex < blob.HoleCount; holeIndex++)
       {
              // Retrieve a given hole
```

```
                EHole hole = blob.GetHole(holeIndex);
        }
}
```

Continuous Mode

Functional Guide | Reference: SetContinuousModeEnabled, FlushContinuousMode

```
////////////////////////////////////////////////
// This code snippet shows how to build blobs  //
// in the continuous mode context.             //
////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Enable the continuous mode
encoder.ContinuousModeEnabled= true;

// Loop to acquire 50 different chunks
for (int count = 0; count < 50 ; count++)
{
        // Store the new chunk into srcImage
        // ...

        // Encode the current chunk
        encoder.Encode(srcImage, codedImage);
}

// Flush the continuous mode
encoder.FlushContinuousMode(codedImage);
```

7.2. Computing Blobs Features

Functional Guide | Reference: GetGravityCenter, GetObj

```
////////////////////////////////////////////////////////////
// This code snippet shows how to retrieve blobs' features. //
////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Encode the source image
```

```
encoder.Encode(srcImage, codedImage);

for (uint index = 0; index < codedImage.GetObjCount(); index++)
{
       // Retrieve the selected blob gravity center
       EObject blob = codedImage.GetObj(index);
       float centerX = blob.GravityCenter.X;
       float centerY = blob.GravityCenter.Y;
}
```

7.3. Selecting and Sorting Blobs

Functional Guide | Reference: AddObjects, ElementCount, RemoveUsingUnsignedIntegerFeature, Sort

```
//////////////////////////////////////////////////////
// This code snippet shows how to build blobs, select  //
// some of them and sort the selected ones.            //
//////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Encode the source image
encoder.Encode(srcImage, codedImage);

// Create a blob selection
EObjectSelection selection= new EObjectSelection();
selection.AddObjects(codedImage);

// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature.Area, 100, ESingleThresholdMode.Less);

// Retrieve the number of remaining blobs
uint numBlobs= selection.ElementCount;

// Sort the remaining blobs based on their area
selection.Sort(EFeature.Area, ESortDirection.Ascending);

// Retrieve the selected blobs
for (uint index = 0; index < numBlobs; index++)
{
    float centerX= selection.GetElement(index).GravityCenterX;
    float centerY= selection.GetElement(index).GravityCenterY;
}
```

7.4. Using Flexible Masks

Constructing Blobs

Functional Guide | Reference: Encode

```
////////////////////////////////////////////////////
// This code snippet shows how to build blobs inside  //
// a region defined by a flexible mask.               //
////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask = new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Encode the source image regions
// corresponding to the mask do care areas
encoder.Encode(srcImage, mask, codedImage);
```

Generating a Flexible Mask from an Encoded Image

Functional Guide | Reference: RenderMask

```
////////////////////////////////////////////////////
// This code snippet shows how to generate a flexible  //
// mask from an encoded image.                         //
////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Encode the source image
encoder.Encode(srcImage, codedImage);

// The source image and the mask must have the same size
mask.SetSize(srcImage);

// Create the mask based on the white layer
// of the coded image
codedImage.RenderMask(mask, ↵);
```

Generating a Flexible Mask from a Blob Selection

Functional Guide | Reference: RenderMask

```
////////////////////////////////////////////////////////
// This code snippet shows how to generate a flexible  //
// mask from a selection of blobs.                     //
////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 mask= new EImageBW8();

// Image encoder
EImageEncoder encoder= new EImageEncoder();

// Coded image
ECodedImage2 codedImage= new ECodedImage2();

// ...

// Encode the source image
encoder.Encode(srcImage, codedImage);

// The source image and the mask must have the same size
mask.SetSize(srcImage);

// Create a blob selection
EObjectSelection selection= new EObjectSelection();
selection.AddObjects(codedImage);

// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature.Area, ↵00, ESingleThresholdMode.Less);

// Create the mask based on the blob selection
selection.RenderMask(mask);

// Sort the remaining blobs based on their area
selection.Sort(EFeature.Area, ESortDirection.Descending);

// Create the mask corresponding to the largest blob
selection.GetElement(0).RenderMask(mask);
```

7.5. Using the Object Template Matcher

Functional Guide | Reference: EObjectTemplateMatcher

```
////////////////////////////////////////////////////////////
// This code snippet shows how to use EObjectTemplateMatcher //
// for alignment and template matching                       //
////////////////////////////////////////////////////////////

// Encode the template image
EImageEncoder encoder = new EImageEncoder();
ECodedImage2 coded_img = new ECodedImage2();

EImageBW8 template_img = new EImageBW8();
encoder.Encode(template_img, coded_img);
```

```
EObjectSelection object_select = new EObjectSelection();
object_select.AddObjects(coded_img);

// Initialize EObjectTemplateMatcher
EObjectTemplateMatcher object_matcher = new EObjectTemplateMatcher();
object_matcher.EnableAlignment = true; // optional
object_matcher.MaximumDistance = 60;   // optional

// set the template
object_matcher.BuildTemplate(object_select);

// Encode the test image
EImageBW8 test_img = new EImageBW8();
encoder.Encode(test_img, coded_img);

// Build a selection of test objects
object_select.Clear();
object_select.AddObjects(coded_img);
object_select.RemoveUsingUnsignedIntegerFeature(EFeature.Area, 0, ESingleThresholdMode.Less);  // optional
filter

// Perform the alignment and the matching
object_matcher.SortSelection(object_select);

// Get the number of matches
int num = object_matcher.NumberOfPairedObjects;

// Retrieve the template indexes for each selection object
int[] template_indexes = object_matcher.TemplateIndexes;
```

8. EasyMatch

8.1. Pattern Learning

Functional Guide | Reference: LearnPattern

```
////////////////////////////////////////////////////
// This code snippet shows how to learn a pattern  //
// defined by a region of interest (ROI).          //
////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ROI constructor
EROIBW8 pattern= new EROIBW8();

// EMatcher constructor
EMatcher matcher= new EMatcher();

// ...

// Attach the ROI to the source image
// and set its position
pattern.Attach(srcImage);
pattern.SetPlacement(24, 52, 200, 200);

// Learn the pattern
matcher.LearnPattern(pattern);
```

8.2. Setting Search Parameters

Functional Guide | Reference: SetMaxPositions, SetMinAngle, SetMaxAngle, SetMinScore, SetInterpolate, Save

```
////////////////////////////////////////////////////////
// This code snippet shows how to tune pattern matching //
// search parameters and save them into a file.         //
////////////////////////////////////////////////////////

// Image constructor
EImageBW8 pattern= new EImageBW8();

// EMatcher constructor
EMatcher matcher= new EMatcher();

// ...

// Learn the pattern
matcher.LearnPattern(pattern);

// Set the maximum number of occurrences
matcher.MaxPositions= 5;
```

```
// Set the rotation tolerances
matcher.MinAngle= -20.0f;
matcher.MaxAngle= 20.0f;

// Enable sub-pixel accuracy
matcher.Interpolate= true;

// Set the minimum score
matcher.MinScore= 0.70f;

// Save the matching context into a model file
matcher.Save("myModel.mch");
```

8.3. Pattern Matching and Retrieving Results

Functional Guide | Reference: Load, Match, GetNumPositions, GetPosition

```
//////////////////////////////////////////////////
// This code snippet shows how to perform pattern  //
// matching operations and retrieve the results.   //
//////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EMatcher constructor
EMatcher matcher= new EMatcher();

// ...

// Load a model file
matcher.Load("myModel.mch");

// Perform the matching
matcher.Match(srcImage);

// Retrieve the number of occurrences
uint numOccurrences= matcher.NumPositions;

// Retrieve the first occurrence
EMatchPosition myOccurrence= matcher.GetPosition(0);

// Retrieve its score and position
float score= myOccurrence.Score;
float centerX= myOccurrence.CenterX;
float centerY= myOccurrence.CenterY;
```

9. EasyFind

9.1. Pattern Learning

Functional Guide | Reference: Learn

```
////////////////////////////////////////////////////
// This code snippet shows how to learn a pattern  //
// defined by a region of interest (ROI).          //
////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ROI constructor
EROIBW8 pattern= new EROIBW8();

// EPatternFinder constructor
EPatternFinder finder= new EPatternFinder();

// ...

// Attach the ROI to the source image
// and set its position
pattern.Attach(srcImage);
pattern.SetPlacement(24, 52, 200, 200);

// Learn the pattern
finder.Learn(pattern);
```

9.2. Setting Search Parameters

Functional Guide | Reference: SetMaxInstances, SetAngleTolerance, SetMinScore, Save

```
////////////////////////////////////////////////////////
// This code snippet shows how to tune pattern finding  //
// search parameters and save them into a file.         //
////////////////////////////////////////////////////////

// Image constructor
EImageBW8 pattern= new EImageBW8();

// EPatternFinder constructor
EPatternFinder finder= new EPatternFinder();

// ...

// Learn the pattern
finder.Learn(pattern);

// Set the maximum number of occurrences
finder.MaxInstances= 5;

// Set the rotation tolerances
```

```
finder.AngleTolerance= 20.0f;

// Set the minimum score
finder.MinScore= 0.70f;

// Save the finding context into a model file
finder.Save("myModel.fnd");
```

9.3. Pattern Finding and Retrieving Results

Functional Guide | Reference: Load, Find, GetScore, GetCenter

```
//////////////////////////////////////////////////
// This code snippet shows how to perform pattern  //
// finding operations and retrieve the results.    //
//////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EPatternFinder constructor
EPatternFinder finder= new EPatternFinder();

// EFoundPattern constructor
EFoundPattern[] foundPattern= null;

// ...

// Load a model file
finder.Load("myModel.fnd");

// Perform the pattern finding
foundPattern= finder.Find(srcImage);

// Retrieve the number of instances
int numInstances= foundPattern.Length;

// Retrieve the score and the
// position of the first instance
float score= foundPattern[0].Score;
float centerX= foundPattern[0].Center.X;
float centerY= foundPattern[0].Center.Y;
```

10. EasyGauge

10.1. Point Location

Functional Guide | Reference: SetTransitionType, SetTransitionChoice, SetCenterXY, SetTolerance, Measure, GetMeasuredPoint, GetX, GetY

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to create a point location tool, //
// adjust the transition parameters, set the nominal gauge      //
// position, perform the measurement and retrieve the result.   //
////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EPointGauge constructor
EPointGauge pointGauge= new EPointGauge();

// Adjust the transition parameters
pointGauge.TransitionType= ETransitionType.Wb;
pointGauge.TransitionChoice= ETransitionChoice.Closest;

// Set the gauge nominal position
pointGauge.SetCenterXY(256.0f, 256.0f);

// Set the gauge length to ↵0 units and the angle to 45°
pointGauge.SetTolerances(↵0.0f, 45.0f);

// Measure
pointGauge.Measure(srcImage);

// Get the measured point coordinates
float measuredX = pointGauge.GetMeasuredPoint().X;
float measuredY = pointGauge.GetMeasuredPoint().Y;

// Save the point gauge measurement context
pointGauge.Save("myPointGauge.gge");
```

10.2. Line Fitting

Functional Guide | Reference: SetTransitionType, SetTransitionChoice, SetTransitionIndex, SetLine, Measure, GetMeasuredLine, GetOrg, GetEnd

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to create a line measurement tool, //
// adjust the transition parameters, set the nominal gauge        //
// position, perform the measurement and retrieve the result.     //
////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ELineGauge constructor
```

```
ELineGauge lineGauge= new ELineGauge();

// Adjust the transition parameters
lineGauge.TransitionType= ETransitionType.Bw;
lineGauge.TransitionChoice= ETransitionChoice.NthFromEnd;
lineGauge.TransitionIndex= 2;

// Set the line fitting gauge position,
// length (50 units) and orientation (20°)
EPoint center= new EPoint(256.0f, 256.0f);
ELine line= new ELine(center, 50.0f, 20.0f);
lineGauge.Line= line;

// Measure
lineGauge.Measure(srcImage);

// Get the origin and end point coordinates of the fitted line
EPoint originPoint = lineGauge.MeasuredLine.Org;
EPoint endPoint = lineGauge.MeasuredLine.End;

// Save the point gauge measurement context
lineGauge.Save("myLineGauge.gge");
```

10.3. Circle Fitting

Functional Guide | Reference: SetTransitionType, SetTransitionChoice, SetCircle, Measure, GetMeasuredCircle, GetCenter, GetRadius

```
////////////////////////////////////////////////////////////////////////
// This code snippet shows how to create a circle measurement tool, //
// adjust the transition parameters, set the nominal gauge          //
// position, perform the measurement and retrieve the result.       //
////////////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ECircleGauge constructor
ECircleGauge circleGauge= new ECircleGauge();

// Adjust the transition parameters
circleGauge.TransitionType= ETransitionType.Bw;
circleGauge.TransitionChoice= ETransitionChoice.LargestAmplitude;

// Set the Circle fitting gauge position, diameter (50 units),
// starting angle (↩0°), and amplitude (270°)
EPoint center= new EPoint(256.0f, 256.0f);
ECircle circle= new ECircle(center, 50.0f, ↩0.0f, 270.0f);
circleGauge.Circle = circle;

// Measure
circleGauge.Measure(srcImage);

// Get the center point coordinates and the radius of the fitted circle
float centerX = circleGauge.MeasuredCircle.Center.X;
float centerY = circleGauge.MeasuredCircle.Center.Y;
float radius = circleGauge.MeasuredCircle.Radius;

// Save the point gauge measurement context
circleGauge.Save("myCircleGauge.gge");
```

10.4. Rectangle Fitting

Functional Guide | Reference: SetTransitionType, SetTransitionChoice, SetRectangle, Measure, GetMeasuredRectangle, GetSizeX, GetSizeY, GetAngle

```
////////////////////////////////////////////////////////////////////////
// This code snippet shows how to create a rectangle measurement tool, //
// adjust the transition parameters, set the nominal gauge position,   //
// perform the measurement and retrieve the result.                    //
////////////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// ERectangleGauge constructor
ERectangleGauge rectangleGauge= new ERectangleGauge();

// Adjust the transition parameters
rectangleGauge.TransitionType= ETransitionType.Bw;
rectangleGauge.TransitionChoice= ETransitionChoice.LargestAmplitude;

// Set the rectangle fitting gauge position,
// size (50x30 units) and orientation (↦5°)
rectangleGauge.SetCenterXY(256.0f, 256.0f);
rectangleGauge.SetSize(50.0f, 30.0f);
rectangleGauge.Angle = ↦5.0f;

// Measure
rectangleGauge.Measure(srcImage);

// Get the size and the rotation angle of the fitted rectangle
float sizeX = rectangleGauge.MeasuredRectangle.SizeX;
float sizeY = rectangleGauge.MeasuredRectangle.SizeY;
float angle = rectangleGauge.MeasuredRectangle.Angle;

// Save the point gauge measurement context
rectangleGauge.Save("myRectangleGauge.gge");
```

10.5. Wedge Fitting

Functional Guide | Reference: SetTransitionType, SetTransitionChoice, SetWedge, Measure, GetMeasuredWedge, GetInnerRadius, GetOuterRadius

```
////////////////////////////////////////////////////////////////////////
// This code snippet shows how to create a wedge measurement tool, //
// adjust the transition parameters, set the nominal gauge         //
// position, perform the measurement and retrieve the result.      //
////////////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EWedgeGauge constructor
EWedgeGauge wedgeGauge= new EWedgeGauge();

// Adjust the transition parameters
wedgeGauge.TransitionType= ETransitionType.Bw;
wedgeGauge.TransitionChoice= ETransitionChoice.NthFromBegin;
wedgeGauge.TransitionIndex= 0;
```

```
// Set the wedge fitting gauge position, diameter (50 units),
// breadth (-25 units), starting angle (0°) and amplitude (270°)
EPoint center= new EPoint(256.0f, 256.0f);
EWedge wedge= new EWedge(center, 50.0f, -25.0f, 0.0f, 270.0f);
wedgeGauge.Wedge= wedge;

// Measure
wedgeGauge.Measure(srcImage);

// Get the inner and outer radius of the fitted wedge
float innerRadius = wedgeGauge.MeasuredWedge.InnerRadius;
float outerRadius = wedgeGauge.MeasuredWedge.OuterRadius;

// Save the point gauge measurement context
wedgeGauge.Save("myWedgeGauge.gge");
```

10.6. Gauge Grouping

Gauge Hierarchy

Functional Guide | Reference: Attach, SetName, Save

```
////////////////////////////////////////////////////////////
// This code snippet shows how to create a gauge hierarchy  //
// and save it into a file.                                 //
////////////////////////////////////////////////////////////

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// Gauges constructor
ERectangleGauge rectangleGauge= new ERectangleGauge();
ECircleGauge circleGauge►= new ECircleGauge();
ECircleGauge circleGauge2= new ECircleGauge();

// ...

// Attach the rectangle gauge to the EWorldShape
rectangleGauge.Attach(worldShape);

// Attach the circle gauges to the rectangle gauge
circleGauge►.Attach(rectangleGauge);
circleGauge2.Attach(rectangleGauge);

// Set the first circle gauge name
circleGauge►.Name= "myCircleGauge►";

// ...

// Save worldShape together with its daughters
worldShape.Save("myWorldShape.gge", true);
```

Complex Measurement

Functional Guide | Reference: Load, GetNumDaughters, Process, GetDaughter, GetShapeNamed

```
////////////////////////////////////////////////////////
// This code snippet shows how to trigger the measurement //
// of a whole gauge hierarchy and retrieve the results.   //
////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// Load the EWorldShape together with its daughters
worldShape.Load("myWorldShape.gge", true);

// Retrieve the number of worldShape's daughters
uint numDaughters= worldShape.NumDaughters;

// ...

// Trigger the measurement of all the
// gauges attached to the EWorldShape
worldShape.Process(srcImage, true);

// Retrieve the measurement result of
// the first daughter (a rectangle gauge)
ERectangleGauge rectangleGauge= (ERectangleGauge)worldShape.GetDaughter(0);
float sizeX= rectangleGauge.MeasuredRectangle.SizeX;

// Retrieve the measurement result of a
// daughter gauge called "myCircleGauge►"
ECircleGauge circleGauge= (ECircleGauge)worldShape.GetShapeNamed("myCircleGauge►");
EPoint center= circleGauge.MeasuredCircle.Center;
```

10.7. Calibration using EWorldShape

Functional Guide | Reference

Calibration by Guesswork

Functional Guide | Reference: SetSensor, GetXResolution, GetYResolution

```
////////////////////////////////////////////////////////
// This code snippet shows how to perform a calibration //
// by guesswork.                                        //
////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// ...

// Compute the calibration coefficients
// Field of view: 32x24 mm
worldShape.SetSensor(srcImage.Width, srcImage.Height, 32.0f, 24.0f);

// Retrieve the spatial resolution
float resolutionX= worldShape.XResolution;
```

```
float resolutionY= worldShape.YResolution;
```

Landmark-Based Calibration

Functional Guide | Reference: EmptyLandmarks, AddLandmark, Calibrate

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a landmark-based //
// calibration.                                            //
////////////////////////////////////////////////////////////

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// ...

// Reset the calibration context
worldShape.EmptyLandmarks();

// Loop on the landmarks
for(int index= 0; index < numLandmarks; index++)
{
    // Get the I-th landmark as a pair of EPoint(x, y)
    EPoint sensorPoint, worldPoint;

    // Retrieve and store the relevant data into worldPoint and sensorPoint
    sensorPoint = myIthLandmark_Sensor;
    worldPoint = myIthLandmark_World;

    // Add the I-th pair
    worldShape.AddLandmark(sensorPoint, worldPoint);
}

// Perform the calibration
worldShape.Calibrate((int)ECalibrationMode.Skewed);
```

Dot Grid-Based Calibration

Functional Guide | Reference: EmptyLandmarks, AddPoint, RebuildGrid, AutoCalibrate

```
////////////////////////////////////////////////////////////
// This code snippet shows how to perform a dot grid-based //
// calibration.                                            //
////////////////////////////////////////////////////////////

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// ...

// Reset the calibration context
worldShape.EmptyLandmarks();

// Loop on the dots
for(int index= 0; index < numDots; index++)
{
    // Get the I-th dot as an EPoint(x, y)
    EPoint dotPoint;

    // Retrieve and store the relevant data into dotPoint
```

```
    dotPoint = myIthDot;

    // Add the I-th dot
    worldShape.AddPoint(dotPoint);
}

// Reconstruct the grid topology
// pitch X and Y = 5 units
worldShape.RebuildGrid(5, 5);

// Perform the calibration
// the calibration modes are computed automatically
worldShape.AutoCalibrate(true);
```

Coordinates Transform

Functional Guide | Reference: SensorToWorld, WorldToSensor

```
////////////////////////////////////////////////////////////
// This code snippet shows how to convert coordinates from //
// the Sensor space to the World space and conversely.     //
////////////////////////////////////////////////////////////

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// EPoint constructor
EPoint sensor= new EPoint();
EPoint world= new EPoint();

// ...

// Perform the calibration
worldShape.Calibrate((int)ECalibrationMode.Scaled | (int)ECalibrationMode.Skewed);

// Retrieve the world coordinates of a point, knowing its sensor coordinates
world= worldShape.SensorToWorld(sensor);

// Retrieve the sensor coordinates of a point, knowing its world coordinates
sensor= worldShape.WorldToSensor(world);
```

Image Unwarping

Functional Guide | Reference: SetupUnwarp, Unwarp

```
////////////////////////////////////////////////////////////
// This code snippet shows how to unwarp an image based //
// of the computed calibration coefficients.            //
////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 srcImage= new EImageBW8();
EImageBW8 dstImage= new EImageBW8();

// EWorldShape constructor
EWorldShape worldShape= new EWorldShape();

// Lookup table constructor
EUnwarpingLut lut= new EUnwarpingLut();

// ...
```

```
// Perform the calibration
worldShape.Calibrate((int)ECalibrationMode.Tilted | (int)ECalibrationMode.Radial);

// Setup the lookup table for unwarping
worldShape.SetupUnwarp(lut, srcImage, true);

// Perform the image unwarping
worldShape.Unwarp(lut, srcImage, dstImage, true);
```

11. EasyOCR

11.1. Learning Characters

Functional Guide | Reference: NewFont, SetTextColor, SetMinCharWidth, SetMaxCharWidth, SetMinCharHeight, SetMaxCharHeight, SetNoiseArea, LearnPatterns, BuildObjects, FindAllChars, Save

```
////////////////////////////////////////////////////
// This code snippet shows how to learn characters  //
// based on an image featuring a known text and     //
// save the corresponding font file.                //
////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EOCR constructor
EOCR ocr= new EOCR();

// Text to be learned (all digits)
// Assuming the image contains this text
string text= "0►23456789";

// ...

// Create a new font
ocr.NewFont(8, ►►);

// Adjust the segmentation parameters
ocr.TextColor= EOCRColor.BlackOnWhite;
ocr.MinCharWidth= ►5;
ocr.MaxCharWidth= 50;
ocr.MinCharHeight= ►5;
ocr.MaxCharHeight= 75;
ocr.NoiseArea= ►5;

// Segment the characters
ocr.BuildObjects(srcImage);
ocr.FindAllChars(srcImage);

// Learn the characters
ocr.LearnPatterns(srcImage, text, (int)EOCRClass.Digit);

// Save the font into a file
ocr.Save("myFont.ocr");
```

11.2. Recognizing Characters

Functional Guide | Reference: Load, Recognize

```
////////////////////////////////////////////////////
// This code snippet shows how to load a font file,  //
// perform a default character recognition operation //
```

```
// and perform a character recognition operation     //
// using a class filter.                             //
/////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// EOCR constructor
EOCR ocr= new EOCR();

// Load the font file
ocr.Load("myFont.ocr");

// ...

// Recognize the characters
string text= ocr.Recognize(srcImage, ↵0, (int)EOCRClass.AllClasses);

// Alternatively
// Define the character filter (2 letters and 3 digits)
uint[] charFilter = new uint[5];
charFilter[0] = (uint)EOCRClass.UpperCase;
charFilter[↵] = (uint)EOCRClass.UpperCase;
charFilter[2] = (uint)EOCRClass.Digit;
charFilter[3] = (uint)EOCRClass.Digit;
charFilter[4] = (uint)EOCRClass.Digit;

// Recognize the characters with class filtering
text = ocr.Recognize(srcImage, ↵0, charFilter);
```

12. EasyOCR2

12.1. Detecting Characters

```
////////////////////////////////////////////////////
// This code snippet shows how to detect characters   //
// in an image, using a few parameters and a topology //
////////////////////////////////////////////////////
// Load an Image
EImageBW8 image = new EImageBW8();
image.Load("image.tif");

// Attach a ROI to the image
EROIBW8 roi = new EROIBW8();
roi.Attach(image, 50, 224, 340, 96);

// Create an EOCR2 instance
EOCR2 ocr2 = new EOCR2();

// Set the expected character sizes
ocr2.CharsWidthRange = new EIntegerRange(25,25);
ocr2.CharsHeight = 37;

// Set the text polarity, in this case WhiteOnBlack
ocr2.TextPolarity = EasyOCR2TextPolarity.WhiteOnBlack;

// Set the topology
ocr2.Topology = ".{↵0}\n.{3} .{4}";

// Detect the text in the image. The output Text structure contains:
// - an individual textbox for each character
// - an individual bitmap image for each character
// - a threshold value to binarize the bitmap image for each character
// All structured in a hierarchy with Lines -> Words -> Characters
EOCR2Text text = ocr2.Detect(roi);

// Cleanup
text.Dispose();
ocr2.CharsWidthRange.Dispose();
ocr2.Dispose();
roi.Dispose();
image.Dispose();
```

The image used in this code snippet

12.2. Learning Characters

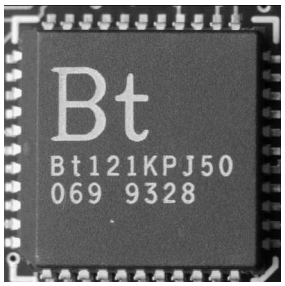
```
////////////////////////////////////////////////////
// This code snippet shows how to learn characters  //
// based on an image featuring a known text and     //
// save the corresponding character database        //
////////////////////////////////////////////////////
// Load an Image
EImageBW8 image = new EImageBW8();
image.Load("image.tif");

// Attach a ROI to the image
EROIBW8 roi = new EROIBW8();
roi.Attach(image, 50, 224, 340, 96);

// Create an EOCR2 instance
EOCR2 ocr2 = new EOCR2();

// Set the required parameters
ocr2.CharsWidthRange = new EIntegerRange(25,25);
ocr2.CharsHeight = 37;
ocr2.TextPolarity = EasyOCR2TextPolarity.WhiteOnBlack;
ocr2.Topology = ".{►0}\n.{3} .{4}";

// Learn from the reference image:
//   ►) Detect the text in the image
EOCR2Text text = ocr2.Detect(roi);
//   2) Set the true values of the text
text.Text = "Bt►2►KPJ80\n786 95►2";
//   3) Add the characters to the character database
ocr2.Learn(text);

// Save the character database
ocr2.SaveCharacterDatabase("myDB.o2d");

// Alternatively, save the model file.
// This will store the character database and the parameter settings
ocr2.Save("myModel.o2m");

// Cleanup
text.Dispose();
ocr2.CharsWidthRange.Dispose();
ocr2.Dispose();
roi.Dispose();
image.Dispose();
```

The image used in this code snippet

12.3. Reading Characters

Reading Using TrueType Fonts

```
////////////////////////////////////////////////////
// This code snippet shows how to                 //
// - create a character database from TrueType fonts  //
// - read the text in an image                    //
////////////////////////////////////////////////////
// Load an image
EImageBW8 image = new EImageBW8();
image.Load("image.tif");

// Attach an ROI
EROIBW8 roi = new EROIBW8();
roi.Attach(image, 50, 224, 340, 96);

// Create an EOCR2 instance
EOCR2 ocr2 = new EOCR2();

// Set the required parameters
ocr2.CharsWidthRange = new EIntegerRange(25,25);
ocr2.CharsHeight = 37;
ocr2.Topology = "[LN]{►0}\nN{3} N{4}";
ocr2.TextPolarity = EasyOCR2TextPolarity.WhiteOnBlack;

// Add TrueType character to the character database
ocr2.AddCharactersToDatabase("C:\\Windows\\Fonts\\calibrib.ttf");
ocr2.AddCharactersToDatabase("C:\\Windows\\Fonts\\yugothb.ttc");

// Read text from the image
string result = ocr2.Read(roi);

// Cleanup
ocr2.CharsWidthRange.Dispose();
ocr2.Dispose();
roi.Dispose();
image.Dispose();
```



The image used in this code snippet

Reading Using EOCR2 Character Database

```
////////////////////////////////////////////////////
// This code snippet shows how to                 //
```

```
// - load a pre-made character database               //
// - read the text in an image                        //
//////////////////////////////////////////////////////////
// Load an image
EImageBW8 image = new EImageBW8();
image.Load("image.tif");

// Attach an ROI
EROIBW8 roi = new EROIBW8();
roi.Attach(image, 50, 224, 340, 96);

// Create an EOCR2 instance
EOCR2 ocr2 = new EOCR2();

// Set the required parameters
ocr2.CharsWidthRange = new EIntegerRange(25,25);
ocr2.CharsHeight = 37;
ocr2.Topology = "[LN]{►0}\nN{3} N{4}";
ocr2.TextPolarity = EasyOCR2TextPolarity.WhiteOnBlack;

// Add a pre-made character database to the EOCR2 instance
ocr2.AddCharactersToDatabase("myDB.o2d");

// Read text from the image
string result = ocr2.Read(roi);

// Cleanup
ocr2.CharsWidthRange.Dispose();
ocr2.Dispose();
roi.Dispose();
image.Dispose();
```

Reading Using EOCR2 Model File

```
//////////////////////////////////////////////////////////
// This code snippet shows how to                     //
// - load a pre-made model file                       //
// - read the text in an image                        //
//////////////////////////////////////////////////////////
// Load an image
EImageBW8 image = new EImageBW8();
image.Load("image.tif");

// Attach an ROI
EROIBW8 roi = new EROIBW8();
roi.Attach(image, 50, 224, 340, 96);

// Create an EOCR2 instance
EOCR2 ocr2 = new EOCR2();

// Load a pre-made model file, this will:
// - (re)set all parameters
// - add the character database in the model file to the EOCR2 instance
ocr2.Load("myModel.o2m");

// Read text from the image
string result = ocr2.Read(roi);

// Cleanup
ocr2.Dispose();
roi.Dispose();
image.Dispose();
```

12.4. View Bitmap

```
////////////////////////////////////////////
// This code snippet shows how to inspect the //
// characters in a character database         //
////////////////////////////////////////////
// Create an EOCR2 instance
EOCR2 ocr2 = new EOCR2();
// Load the character database
ocr2.AddCharactersToDatabase("database.o2d");
// Extract the character database
EOCR2CharacterDatabase db = ocr2.CharacterDatabase;
// Select the character that we are interested in (e.g. the third one)
EOCR2DatabaseCharacter chr = db.GetCharacter(2);
// Extract the bitmap for that character
EImageBW8 img = chr.Bitmap;
```

13. EasyBarCode

13.1. Reading a Bar Code

Functional Guide | Reference: Read

```
//////////////////////////////////////////////////
// This code snippet shows how to read a bar code  //
//////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Bar code reader constructor
EBarCode reader= new EBarCode();

// String for the decoded bar code
string result;

// ...

// Read the source image
result = reader.Read(srcImage);
```

13.2. Reading a Bar Code Following a Given Symbology

Functional Guide | Reference: SetAdditionalSymbologies, SetVerifyChecksum, Detect, Decode, GetNumDecodedSymbologies

```
//////////////////////////////////////////////////////////
// This code snippet shows how to enable a given symbology,  //
// enable the checksum verification, perform the bar code    //
// detection and retrieve the decoded string.                //
//////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Bar code reader constructor
EBarCode reader= new EBarCode();

// String for the decoded bar code
string result;

// ...

// Disable all standard symbologies
reader.StandardSymbologies= 0;

// Enable the Code32 symbology only
reader.AdditionalSymbologies= (int)ESymbologies.Code32;
```

```
// Enable checksum verification
reader.VerifyChecksum= true;

// Detect all possible meanings of the bar code
reader.Detect(srcImage);

// Retrieve the number of symbologies for
// which the decoding process was successful
uint numDecoded = reader.NumDecodedSymbologies;

if(numDecoded > 0)
{
    // Decode the bar code according to the Code32 symbology
    result = reader.Decode(ESymbologies.Code32);
}
```

13.3. Reading a Bar Code of Known Location

Functional Guide | Reference: SetKnownLocation, SetCenterXY, SetReadingSize

```
////////////////////////////////////////////////////
// This code snippet shows how to specify the bar code  //
// position and perform the bar code reading.           //
////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Bar code reader constructor
EBarCode reader= new EBarCode();

// String for the decoded bar code
string result;

// ...

// Disable automatic bar code detection
reader.KnownLocation = true;

// Set the bar code position
reader.SetCenterXY(450.0f, 400.0f);
reader.SetSize(250.0f, ▸▸0.0f);
reader.SetReadingSize(▸.▸5f, 0.5f);

// Read the bar code at the specified location
result = reader.Read(srcImage);
```

13.4. Reading a Mail Bar Code

Functional Guide | Reference: Read

```
////////////////////////////////////////////////////
// This code snippet shows how to read Mail Barcodes  //
// and retrieve the decoded data.                     //
////////////////////////////////////////////////////
// Image constructor
EImageBW8 srcImage = new EImageBW8();
// Mail barcode reader constructor
```

```
EMailBarcodeReader reader = new EMailBarcodeReader();
// Select expected symbologies and orientations (optional)
reader.ExpectedSymbologies = ...;
reader.ExpectedOrientations = ...;
// ...
// Read
EMailBarcode [] codes = reader.Read(srcImage);
// Retrieve the data included in found mail barcodes
for (int index= 0; index < codes.Length; index++)
{
    string text = codes[index].Text;
    EStringPair [] components = codes[index]. ComponentStrings;
}
```

14. EasyMatrixCode

14.1. Automatic Reading

Functional Guide | Reference: Read, GetDecodedString

```
//////////////////////////////////////////////////////////
// This code snippet shows how to read a data matrix code  //
// and retrieve the decoded string.                        //
//////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Matrix code reader constructor
EMatrixCodeReader reader= new EMatrixCodeReader();

// Matrix code constructor
EMatrixCode mxCode= new EMatrixCode();

// String for the decoded information
string result;

// ...

// Read the source image
mxCode = reader.Read(srcImage);

// Retrieve the decoded string
result = mxCode.DecodedString;
```

14.2. Reading with Prior Learning

Functional Guide | Reference: SetLearnMaskElement, Learn, Read, GetDecodedString

```
//////////////////////////////////////////////////////////////
// This code snippet shows how to learn a given data matrix //
// code type (except its flipping status), perform the      //
// reading and retrieve the decoded string.                 //
//////////////////////////////////////////////////////////////

// Images constructor
EImageBW8 model= new EImageBW8();
EImageBW8 srcImage= new EImageBW8();

// Matrix code reader constructor
EMatrixCodeReader reader= new EMatrixCodeReader();

// Matrix code constructor
EMatrixCode mxCode= new EMatrixCode();

// String for the decoded information
string result;
```

```
// ...

// Tell the reader not to take the flipping into account when learning
reader.SetLearnMaskElement(ELearnParam.Flipping, false);

// Learn the model
reader.Learn(model);

// Read the source image
mxCode = reader.Read(srcImage);

// Retrieve the decoded string
result = mxCode.DecodedString;
```

14.3. Advanced Tuning of the Search Parameters

Functional Guide | Reference: GetSearchParams, ClearLogicalSize, AddLogicalSize, ClearFamily, AddFamily, Read, GetDecodedString

```
////////////////////////////////////////////////////////////////
// This code snippet shows how to explicitly specify the data //
// matrix code logical size and family, perform the reading   //
// and retrieve the decoded string.                           //
////////////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Matrix code reader constructor
EMatrixCodeReader reader= new EMatrixCodeReader();

// Matrix code constructor
EMatrixCode mxCode= new EMatrixCode();

// String for the decoded information
string result;

// ...

// Remove the default logical sizes
reader.SearchParams.ClearLogicalSize();

// Add the ►5x►5 and ►7x►7 logical sizes
reader.SearchParams.AddLogicalSize(ELogicalSize._►5x►5);
reader.SearchParams.AddLogicalSize(ELogicalSize._►7x►7);

// Remove the default families
reader.SearchParams.ClearFamily();

// Add the ECC050 family
reader.SearchParams.AddFamily(EFamily.ECC050);

// Read the source image
mxCode = reader.Read(srcImage);

// Retrieve the decoded string
result = mxCode.DecodedString;
```

14.4. Retrieving Print Quality Grading

Functional Guide | Reference: SetComputeGrading, GetAxialNonUniformityGrade, GetContrastGrade, GetPrintGrowthGrade, GetUnusedErrorCorrectionGrade

```
////////////////////////////////////////////////////////
// This code snippet shows how to read a data matrix code //
// and retrieve its print quality grading.                //
////////////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// Matrix code reader constructor
EMatrixCodeReader reader= new EMatrixCodeReader();

// Matrix code constructor
EMatrixCode mxCode= new EMatrixCode();

// ...

// Enable grading computation
reader.ComputeGrading= true;

// Read the source image
mxCode = reader.Read(srcImage);

// Retrieve the print quality grading
int axialNonUniformityGrade= mxCode.AxialNonUniformityGrade;
int contrastGrade= mxCode.ContrastGrade;
int printGrowthGrade= mxCode.PrintGrowthGrade;
int unusedErrorCorrectionGrade= mxCode.UnusedErrorCorrectionGrade;
```

15. EasyMatrixCode2

15.1. Reading Matrix Codes from an Image

```
////////////////////////////////////////////////////////
// This code snippet shows how to read data matrix codes //
// and retrieve the decoded string.                      //
////////////////////////////////////////////////////////
using EMC2 = Euresys.Open_eVision_x_x.EasyMatrixCode2;
// Load an image
EImageBW8 image = new EImageBW8();
image.Load("image.bmp");
// Prepare a matrix code reader
EMC2.EMatrixCodeReader reader = new EMC2.EMatrixCodeReader();
// Let the reader know that there are no more than 3 codes in the image
reader.MaxNumCodes = 3;
// Read the source image
reader.Read(image);
// Retrieve the detected codes
EMC2.EMatrixCode[] codes = reader.ReadResults;
// Retrieve the decoded string for the first code
string result = codes[0].DecodedString;
```

15.2. Reading with Prior Learning

```
////////////////////////////////////////////////////////////
// This code snippet shows how to learn from a given image, //
// perform the reading and retrieve the decoded string.     //
////////////////////////////////////////////////////////////

using EMC2 = Euresys.Open_eVision_x_x.EasyMatrixCode2;

// Load an image
EImageBW8 image = new EImageBW8();
image.Load("image.bmp");

// Prepare a matrix code reader
EMC2.EMatrixCodeReader reader = new EMC2.EMatrixCodeReader();

// Learn from this image
reader.Learn(image);

// Read the codes in this image
reader.Read(image);

// Retrieve the detected codes
EMC2.EMatrixCode[] codes = reader.ReadResults;

// Retrieve the decoded string of the first code
string result = codes[0].DecodedString;
```

15.3. Inspecting Print Quality Grades

```
////////////////////////////////////////////////////////
// This code snippet shows how to read a data matrix code //
// and retrieve its print quality grades.                 //
////////////////////////////////////////////////////////
using EMC2 = Euresys.Open_eVision_x_x.EasyMatrixCode2;
// Load an image
EImageBW8 image = new EImageBW8();
image.Load("image.bmp");
// Prepare a matrix code reader
EMC2.EMatrixCodeReader reader = new EMC2.EMatrixCodeReader();
// Tell the reader to compute grades for the read codes
reader.ComputeGrading = true;
// Read the codes in this image
reader.Read(image);
// Retrieve the detected codes
EMC2.EMatrixCode[] codes = reader.ReadResults;
// Retrieve the SemiT►O grades of the first code
EMatrixCodeSemiT►OGradingParameters semiT►OGrades = codes[0].SemiT►OGradingParameters;
// Retrieve specific grade values
float cellDefects = semiT►OGrades.CellDefects;
float symbolContrast = semiT►OGrades.SymbolContrast;
float unusedErrorCorrection = semiT►OGrades.UnusedErrorCorrection;
```

15.4. Asynchronous Processing

```
//////////////////////////////////////////////////////////////////////////
// This code snippet shows how to read data matrix codes asynchronously //
// from three separate images.                                          //
// The code in this snippet is valid for C++►► and newer.               //
//////////////////////////////////////////////////////////////////////////
using System.Threading;
using EMC2 = Euresys.Open_eVision_x_x.EasyMatrixCode2;
// create a subroutine that reads the codes from an image
void Read(ref EImageBW8 image, ref EMC2.EMatrixCodeReader reader, ref EMC2.EMatrixCode[] codes, ref bool
finished)
{
  // read the codes in this image
  reader.Read(image);

  // extract the results
  codes = reader.GetReadResults();

  // notify that the reader has finished
  finished = true;
}
void main()
{
  // Prepare three images
  EImageBW8 img► = new EImageBW8();
  EImageBW8 img2 = new EImageBW8();
  EImageBW8 img3 = new EImageBW8();

  // Prepare three matrix code readers
  EMC2.EMatrixCodeReader reader► = new EMC2.EMatrixCodeReader();
  EMC2.EMatrixCodeReader reader2 = new EMC2.EMatrixCodeReader();
  EMC2.EMatrixCodeReader reader3 = new EMC2.EMatrixCodeReader();
```

```
  // Prepare three vectors of matrix code instances
  EMC2.EMatrixCode[] codes► = null;
  EMC2.EMatrixCode[] codes2 = null;
  EMC2.EMatrixCode[] codes3 = null;

   // Prepare three Booleans to track the thread progress
  bool finished► = false;
  bool finished2 = false;
  bool finished3 = false;

  // load the images
  img►.Load("image►.bmp");
  img2.Load("image2.bmp");
  img3.Load("image3.bmp");

  // Launch three threads to read the codes in each image
  Thread thr► = new Thread(() => Read(ref img►, ref reader►, ref codes►, ref finished►));
  Thread thr2 = new Thread(() => Read(ref img2, ref reader2, ref codes2, ref finished2));
  Thread thr3 = new Thread(() => Read(ref img3, ref reader3, ref codes3, ref finished3));

  // Start the threads, they will run in the background.
  thr►.Join();
  thr►.Join();
  thr►.Join();

  // Wait until one of the threads has finished
  while (!(finished► || finished2 || finished3))
    Thread.Sleep(5);

  // Here, we manually stop all code readers, they will stop processing
  // even if they have not yet found the codes in the image
  reader►.StopProcess();
  reader2.StopProcess();
  reader3.StopProcess();

  // wait for the threads to completely finish before continuing
  thr►.Join();
  thr2.Join();
  thr3.Join();
}
```

16. EasyQRCode

16.1. Automatic Reading of a QR Code

Functional Guide | Reference: Read, EQRCodeDecodedStream Class

```
//////////////////////////////////////////////////
// This code snippet shows how to read a QR code  //
// and retrieve the decoded data.                 //
//////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// QR code reader constructor
EQRCodeReader reader= new EQRCodeReader ();

// ...

// Set the source image
reader.SearchField = srcImage;

// Read
EQRCode [] qrCodes = reader.Read();
```

16.2. Retrieving Information of a QR Code

Functional Guide | Reference: Read, EQRCode Class

```
//////////////////////////////////////////////////
// This code snippet shows how to read a QR code  //
// and retrieve the associated information.       //
//////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// QR code reader constructor
EQRCodeReader reader= new EQRCodeReader ();

// ...
// Set the source image
reader.SearchField = srcImage;

// Read
EQRCode [] qrCodes = reader.Read();

// Retrieve version, model and position information
// of the first QR code found, if one was found
if (qrCodes.Length > 0)
{
    uint version = qrCodes[0].Version;
    EQRCodeModel model = qrCodes[0].Model;
```

```
    EQRCodeGeometry geometry = qrCodes[0].Geometry;
}
```

16.3. Tuning the Search Parameters

Functional Guide | Reference: Read, GetDecodedString

```
////////////////////////////////////////////////
// This code snippet shows how to read a QR code  //
// and retrieve the decoded data after setting a  //
// number of search parameters.                   //
////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage= new EImageBW8();

// QR code reader constructor
EQRCodeReader reader= new EQRCodeReader ();

// ...

// Set the source image
reader.SearchField = srcImage;

// Set the search parameters
reader.MaximumVersion = 7;
reader.MinimumIsotropy = 0.9f;

// Set the searched models
reader.SearchedModels = new EQRCodeModel[] {EQRCodeModel.Model2};

// Read
EQRCode [] qrCodes = reader.Read();

// Retrieve the decoded string in best guess mode of the first QR code found
string decodedString = qrCodes[0].GetDecodedString(EByteInterpretationMode.Auto);
```

16.4. Retrieving the Decoded String (Simple)

Functional Guide | Reference: Read, EQRCode Class

```
////////////////////////////////////////////////
// This code snippet shows how to read a QR code  //
// and retrieve the decoded string.               //
////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage = new EImageBW8();

// QR code reader constructor
EQRCodeReader reader = new EQRCodeReader();

// ...

// Set the source image
reader.SearchField = srcImage;

// Read
```

```
EQRCode [] qrCodes = reader.Read();

// Retrieve the data of the first QR code found in best guess mode
string decodedString = qrCodes[0].GetDecodedString(EByteInterpretationMode.Auto);
```

16.5. Retrieving the Decoded String (Safe)

Functional Guide | Reference: Read, EQRCode Class

```
////////////////////////////////////////////////
// This code snippet shows how to read a QR code  //
// and retrieve safely the decoded string         //
////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage = new EImageBW8();

// QR code reader constructor
EQRCodeReader reader = new EQRCodeReader();

// ...

// Set the source image
reader.SearchField = srcImage;

// Read
EQRCode [] qrCodes = reader.Read();

// Retrieve the data of the first QR code found
string decodedString = "";
try
{
  // The QR Code can be fully decoded without user input
  decodedString = qrCodes[0].GetDecodedString();
}
catch (EException exc)
{
  // Handle the exception
  // ...
  // The QR Code cannot be fully decoded without user input
  // use hexadecimal byte interpretation
  decodedString = qrCodes[0].GetDecodedString(EByteInterpretationMode.Hexadecimal);
}
```

16.6. Retrieving the Decoded Data (Advanced)

Functional Guide | Reference: Read, EQRCode Class

```
////////////////////////////////////////////////
// This code snippet shows how to read a QR code  //
// and retrieve its coding mode,                  //
// the raw bit stream and the data part by part   //
////////////////////////////////////////////////

// Image constructor
EImageBW8 srcImage = new EImageBW8();

// QR code reader constructor
```

```
EQRCodeReader reader = new EQRCodeReader();

// ...

// Set the source image
reader.SearchField = srcImage;

// Read
EQRCode [] qrCodes = reader.Read();

// Retrieve the data stream of the first QR code found
EQRCodeDecodedStream stream = qrCodes[0].DecodedStream;

// Retrieve the coding mode and the raw bit stream of the first QR code found
EQRCodeCodingMode codingMode = stream.CodingMode;
byte [] bitstream = stream.RawBitstream;

// Retrieve the encoding and the decoded data of each part of the first QR code found
EQRCodeDecodedStreamPart [] parts = stream.DecodedStreamParts;
for (uint i = 0; i < parts.Length; ++i)
{
  // Retrieve encoding
  EQRCodeEncoding encoding = parts[i].Encoding;

  // Retrieve the decoded data
  byte [] decodedData = parts[i].DecodedData;

  // Interpret the decoded data based on the retrieved encoding
  ...
}
```

17. Easy3DObject

17.1. Extracting 3D Objects with a Selection Criterion

```
// EZmap constructor
EZMap8 zMap = new EZMap8();

// Extractor constructor
E3DObjectExtractor extractor = new E3DObjectExtractor();

// Setting a selection criterion
extractor.WidthRange = new EFloatRange(↵0, 500);

// Extracts the objects from the EZMap
int regionNB = extractor.Extract(zMap);

// Retrieve the extracted objects
E3DObject[] objects = extractor.Objects;
```

17.2. Inspecting a Feature from the List of E3DObjects

```
// Get the list of E3DObjects
E3DObject[] objects = extractor.Objects;

// Get the volume of the first object
float volume = objects[0].Volume;

// Get the ERectangleRegion of the last (the largest) object
ERectangleRegion region = objects[objects.Length - ↵].RectangleRegion;
```

17.3. Drawing a 2D Feature from the List of E3DObjects

```
// Get the list of E3DObjects
E3DObject[] objects = extractor.Objects;

// The GDI drawing surface
Graphics drawGDI;

// Draw the ERegion of each object
int nObjects = objects.Length;
for (int i = 0; i < nObjects; i++)
  objects[i].Draw(drawGDI, E3DObjectFeature.ERegion, new ERGBColor(0, 255, 0));
```

17.4. Drawing 3D Features from a List of E3DObjects

```
// Get the list of E3DObjects
E3DObject[] objects = extractor.Objects;

// Register the list of E3DObject to the 3D viewer
E3DViewer viewer3D = new E3DViewer(orgX, orgY, width, height);
viewer3D.Register3DObjects(objects);

// Define and use a render style for the ReferenceTopPosition feature
ERenderStyle renderStyle = new ERenderStyle();
renderStyle.pointRGB = new EC24A(►00, 0, 0);
viewer3D.SetFeatureStyleForAll3DObjects(renderStyle, E3DObjectFeature.ReferenceTopPosition);

// Set a different rendering color for the first object
ERenderStyle selectedRenderStyle = new ERenderStyle();
selectedRenderStyle.pointRGB = new EC24A(255, 255, 0);
viewer3D.SetFeatureStyleFor3DObject(0, selectedRenderStyle, E3DObjectFeature.ReferenceTopPosition);

// Enable the display of the ReferenceTopPosition feature
viewer3D.ShowFeatureForAll3DObjects(E3DObjectFeature.ReferenceTopPosition);
```

18. Easy3DMatch

18.1. E3DAligner Minimal Code

```
////////////////////////////////////////////////
// This code snippet shows how to compute the     //
// alignment between a sample and a cad reference.//
////////////////////////////////////////////////

// load the reference mesh and define the pose
E3DAligner aligner = new E3DAligner();
EMesh cad = new EMesh();
cad.Load("...");
float azimuthReference = 0.0f, elevationReference = 90.0f;
aligner.SetReference(cad, azimuthReference, elevationReference);

// load the sample
EPointCloud sample = new EPointCloud();
sample.Load("...");
float azimuthSample = 0.0f, elevationSample = -90.0f;

// perform alignment
E3DAlignment alignment = aligner.Align(sample, azimuthSample, elevationSample);
```

18.2. E3DAligner Reprojection Plane

```
////////////////////////////////////////////////
// This code snippet shows how to set the         //
// reprojection plane when performing alignment.  //
////////////////////////////////////////////////

// load the reference mesh and define the pose
E3DAligner aligner = new E3DAligner();
EMesh cad = new EMesh();
cad.Load("...");
float azimuthReference = 0.0f, elevationReference = 90.0f;
aligner.SetReference(cad, azimuthReference, elevationReference);

// define the reprojection plane
bool userKnowsPlaneEquation = false; // depending on the user
if (userKnowsPlaneEquation)
{
  E3DPlane reprojectionPlane = new E3DPlane(new E3DPoint(0, 0, -►), -►5);
  aligner.ScanReprojectionPlane = reprojectionPlane;
}
else
{
  EPointCloud cloud = new EPointCloud();
  cloud.Load("...");
  bool objectAbovePlane = true; // is the object above the plane on the cloud
  aligner.SetFlatScan(cloud, objectAbovePlane);
}
```

```
// load the sample
EPointCloud sample = new EPointCloud();
sample.Load("...");
float azimuthSample = 0.0f, elevationSample = 90.0f;

// perform alignment
E3DAlignment alignment = aligner.Align(sample, azimuthSample, elevationSample);
```

18.3. E3DAlignment Align Sample

```
////////////////////////////////////////////////
// This code snippet shows how to apply the   //
// transformation of the E3DAlignment to the  //
// sample to overlap it on the reference      //
////////////////////////////////////////////////

// perform alignment (see previous examples)
E3DAlignment alignment = new E3DAlignment(); // obtained with E3DAligner
EPointCloud sample = new EPointCloud(); // same pointcloud as the input of the E3DAligner

// align sample on reference
EPointCloud alignedSample = new EPointCloud();
EAffineTransformer.ApplyMatrix(alignment.Pose, sample, alignedSample);
```

18.4. E3DComparer Minimal Sample

```
////////////////////////////////////////////////
// This code snippet shows how to compare a sample //
// with a golden scan reference.                //
////////////////////////////////////////////////

// load the reference golden scan and set reference
E3DComparer comparer = new E3DComparer();
EPointCloud reference = new EPointCloud();
reference.Load("...");
comparer.PointCloudReference = reference;

// set thresholds
float distanceThresh = 0.2f, areaThresh = ⌙.0f;
comparer.SetAnomalyThresholds(distanceThresh, areaThresh);
// Prepare data structures (optional)
comparer.PrepareReference();

// load the sample and perform comparison
EPointCloud sample = new EPointCloud();
sample.Load("...");
comparer.Compare(sample);

// compute anomalies
E3DAnomaly[] anomalies = comparer.ComputesAnomalies();

// TODO: if (anomalies.Length != 0): an anomaly was detected: inspect the sample manually? throw it away?

// get cloud to inspect it manually
EPointCloud visualisationCloud = new EPointCloud();
comparer.GetComparisonPointCloud(visualisationCloud);
```

18.5. E3DComparer Advanced Sample

```
////////////////////////////////////////////////
// This code snippet shows how to set the options //
// when comparing two elements with E3DComparer.  //
////////////////////////////////////////////////

// load the reference cad and set reference
E3DComparer comparer = new E3DComparer();
EMesh cad = new EMesh();
cad.Load("...");
comparer.MeshReference = cad;

// set thresholds
float distanceThresh = 0.2f, areaThresh = ⌙.0f;
float hystDistanceThresh = ⌙.5f, hystAreaThresh = 0.5f;
comparer.SetAnomalyThresholds(distanceThresh, areaThresh);
comparer.SetAnomalyHysteresis(hystDistanceThresh, hystAreaThresh);  // defined relatively to base thresholds

// set ROIs
E3DBox[] rois = new E3DBox[⌙];
rois[0] = new E3DBox(⌙5, ⌙5, ⌙5);
comparer.ROI = rois;
E3DBox[] dontCare = new E3DBox[⌙];
dontCare[0] = new E3DBox(5, 5, 5);
comparer.DontCare = dontCare;
E3DBox[] noExtraMaterial = new E3DBox[⌙];
noExtraMaterial[0] = new E3DBox(new E3DPoint(⌙0, ⌙5, 20), 0, 0, 0, 5, 5, 5);
comparer.NoExtraMaterial = noExtraMaterial;

// prepare data structures (optional)
comparer.PrepareReference();

// load the sample and perform comparison
EPointCloud sample = new EPointCloud();
sample.Load("...");
comparer.Compare(sample);

// compute anomalies
E3DAnomaly[] anomalies = comparer.ComputesAnomalies();

// TODO: if (anomalies.Length) != 0): an anomaly was detected: inspect the sample manually? throw it away?

// get cloud to inspect it manually
EPointCloud visualisationCloud = new EPointCloud();
comparer.GetComparisonPointCloud(visualisationCloud);
```

18.6. E3DMatcher Minimal Sample

```
////////////////////////////////////////////////
// This code snippet shows how to match a sample  //
// with a golden scan reference.                  //
////////////////////////////////////////////////

// load the reference golden scan and set reference
E3DMatcher matcher = new E3DMatcher();
EPointCloud reference = new EPointCloud();
float azimuthReference = 0.0f, elevationReference = 90.0f;
reference.Load("...");
```

```
matcher.SetReference(reference, azimuthReference, elevationReference);

// set thresholds
float distanceThresh = 0.2f, areaThresh = ⅕.0f;
matcher.SetAnomalyThresholds(distanceThresh, areaThresh);
// prepare data structures (optional)
matcher.PrepareReference();

// load the sample and perform comparison
EPointCloud sample = new EPointCloud();
float azimuthSample = 0.0f, elevationSample = -90.0f;
sample.Load("...");
E3DMatch match = matcher.Match(sample, azimuthSample, elevationSample);
E3DAnomaly[] anomalies = match.Anomalies;
// TODO: if (anomalies.Length != 0): an anomaly was detected: inspect the sample manually? throw it away?

// get cloud to inspect it manually
EPointCloud visualisationCloud = new EPointCloud();
matcher.GetComparisonPointCloud(visualisationCloud);
```

18.7. E3DMatcher Advanced Sample

```
////////////////////////////////////////////////////
// This code snippet shows how to set the options //
// when matching two elements with E3DMatcher.    //
////////////////////////////////////////////////////

// load the reference golden scan and set reference
E3DMatcher matcher = new E3DMatcher();
EPointCloud reference = new EPointCloud();
float azimuthReference = 0.0f, elevationReference = 90.0f;
reference.Load("...");
matcher.SetReference(reference, azimuthReference, elevationReference);

// use advanced comparison mode
matcher.ComparisonDistanceMode = EComparisonDistanceMode.Advanced;

// ignore shadows
matcher.EnableMissingPointAsAnomaly = false;

// set thresholds
float distanceThresh = 0.2f, areaThresh = ⅕.0f;
float hystDistanceThresh = ⅕.5f, hystAreaThresh = 0.5f;
matcher.SetAnomalyThresholds(distanceThresh, areaThresh);
matcher.SetAnomalyHysteresis(hystDistanceThresh, hystAreaThresh);  // defined relatively to base thresholds

// retrieve reference poses (reference must have been set)
EZMap8[] referencePoseProjections;
matcher.RetrieveReferencePosesProjections(out referencePoseProjections);

// set ROI on the left half of the object
float originX = 0.0f, originY = 0.0f, width = referencePoseProjections[0].Width / 2, height =
referencePoseProjections[0].Height / 2;
ERectangleRegion roiRegion = new ERectangleRegion(originX, originY, width, height);
matcher.SetComparisonROI(roiRegion);

// set No Extra material on the whole object
originX = 0.0f; originY = 0.0f; width = referencePoseProjections[0].Width / 2; height = referencePoseProjections
[0].Height / 2;
ERectangleRegion noExtraMatRegion = new ERectangleRegion(originX, originY, width, height);
matcher.SetComparisonNoExtraMaterial(roiRegion);
```

```
// prepare data structures (optional)
matcher.PrepareReference();

// load the sample and perform comparison
EPointCloud sample = new EPointCloud();
float azimuthSample = 0.0f, elevationSample = -90.0f;
sample.Load("...");
E3DMatch match = matcher.Match(sample, azimuthSample, elevationSample);
E3DAnomaly[] anomalies = match.Anomalies;

// TODO: if (anomalies.Length != 0):an anomaly was detected: inspect the sample manually? throw it away?

// get cloud to inspect it manually
EPointCloud visualisationCloud = new EPointCloud();
matcher.GetComparisonPointCloud(visualisationCloud);
```