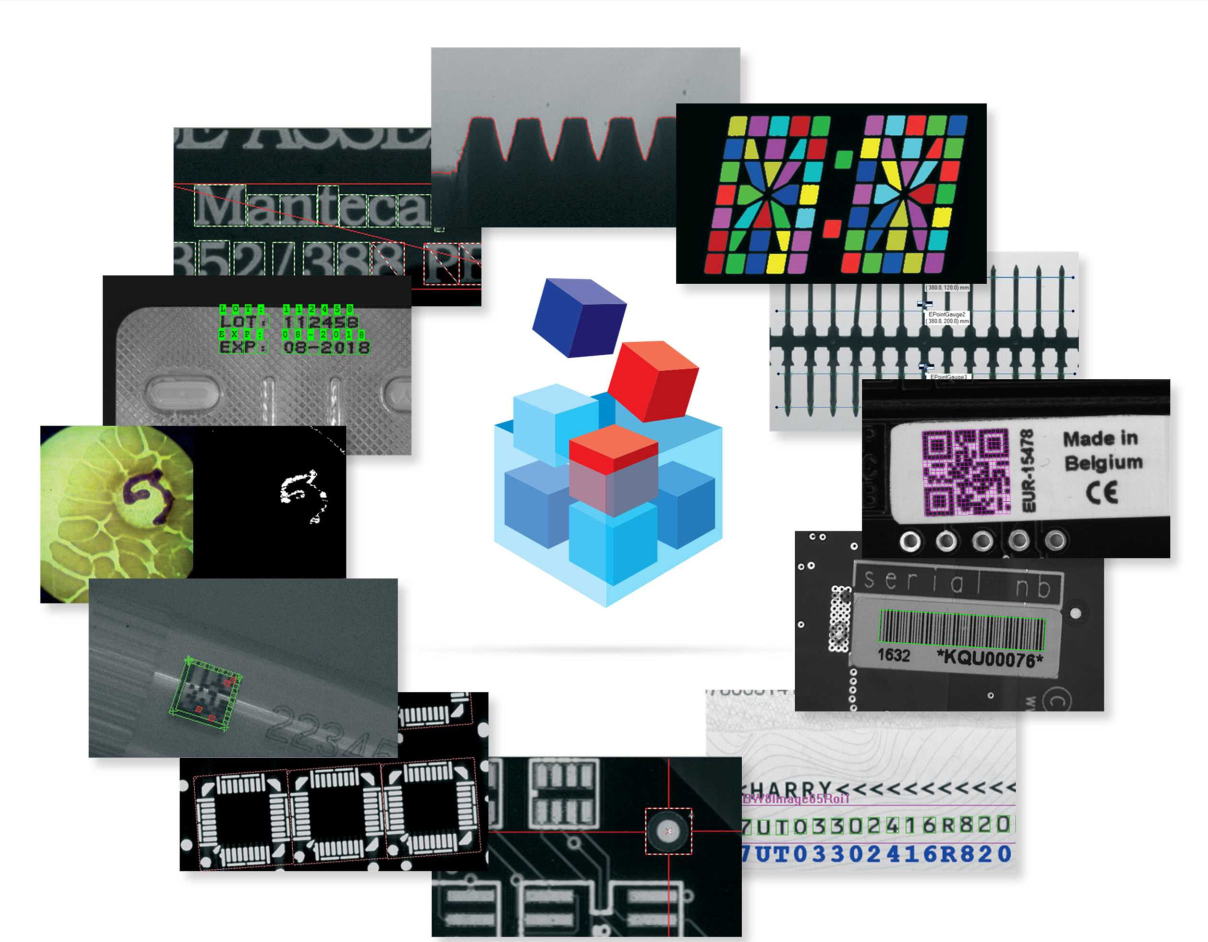
CODE SNIPPETS

Open eVision



Terms of Use

EURESYS s.a. shall retain all property rights, title and interest of the documentation of the hardware and the software, and of the trademarks of EURESYS s.a.

All the names of companies and products mentioned in the documentation may be the trademarks of their respective owners.

The licensing, use, leasing, loaning, translation, reproduction, copying or modification of the hardware or the software, brands or documentation of EURESYS s.a. contained in this book, is not allowed without prior notice.

EURESYS s.a. may modify the product specification or change the information given in this documentation at any time, at its discretion, and without prior notice.

EURESYS s.a. shall not be liable for any loss of or damage to revenues, profits, goodwill, data, information systems or other special, incidental, indirect, consequential or punitive damages of any kind arising in connection with the use of the hardware or the software of EURESYS s.a. or resulting of omissions or errors in this documentation.

This documentation is provided with Open eVision 2.6.1 (doc build 1110).

Contents

1. Basic Types

1.1. Loading and Saving Images

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// 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

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

```
// Images constructor
EImageBW8 srcImage;
```

```
// Size of the third-party image
int sizeX;
int sizeY;
```

```
//Pointer to the third-party image buffer
EBW8* imgPtr;
```

```
// ...
// 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

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

```
EImageBW8 img;
```

```
OEV_UINT8* pixelPtr;
OEV_UINT8* rowPtr;
OEV_UINT8  pixelValue;
OEV_UINT32 rowPitch;
OEV_UINT32 x, y;
```

```
rowPtr = reinterpret_cast <OEV_UINT8*>(img.GetImagePtr());
rowPitch = img.GetRowPitch();
```

```
for (y = 0; y < height; y++)
{
    pixelPtr = rowPtr;

    for (x = 0; x < width; x++)
    {
        pixelValue = *pixelPtr;

        // Add your pixel computation code here

        *pixelPtr = pixelValue;
        pixelPtr++;
    }

    rowPtr += rowPitch;
}
```

1.4. ROI Placement

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

```
// Image constructor
EImageBW8 parentImage;
```

```
// ROI constructor
EROIBW8 myROI;
```

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

```
//Set the ROI position
myROI.SetPlacement(50, 50, 200, 100);
```

1.5. Vector Management

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

```
// EBW8Vector constructor
EBW8Vector ramp;
```

```
// Clear the vector
ramp.Empty();
```

```
// Fill the vector with increasing values
for(int i= 0; i < 128; i++)
{
        ramp.AddElement((EBW8)i);
}
```

```
// Retrieve the 10th element value
EBW8 value= ramp[9];
```

1.6. Exception Management

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

```
try
{
        // Image constructor
        EImageC24 srcImage;
```

```
        // ...
```

```
        // Retrieve the pixel value at coordinates (56, 73)
        EC24 value= srcImage.GetPixel(56, 730);
}
```

```
catch(Euresys::Open_eVision_1_1::EException exc)
{
        // Retrieve the exception description
        std::string error = exc.What();
}
```

2. EasyImage

2.1. Thresholding

Single Thresholding

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// ...
```

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

```
// Minimum residue thresholding (default method)
EasyImage::Threshold(&srcImage, &dstImage);
```

```
// Absolute thresholding (threshold = 110)
EasyImage::Threshold(&srcImage, &dstImage, 110);
```

```
// Relative thresholding (70% black, 30% white)
EasyImage::Threshold(&srcImage, &dstImage, EThresholdMode_Relative, 0, 255, 0.7f);
```

Double Thresholding

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// ...
```

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

```
// Double thresholding, low threshold = 50, high threshold = 150,
// pixels below 50 become black, pixels above 150 become white,
// pixels between thresholds become gray
EasyImage::DoubleThreshold(&srcImage, &dstImage, 50, 150, 0, 128, 255);
```

Histogram-Based Single Thresholding

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

```
// Images constructor
```

```
EImageBW8 srcImage;
```

```
EImageBW8 dstImage;
```

```
// Histogram constructor
EBWHistogramVector histo;
```

```
// Variables
unsigned int thresholdValue;
float avgBelowThr, avgAboveThr;
```

```
// ...
```

```
// Compute the histogram
EasyImage::Histogram(&srcImage, &histo);
```

```
// Compute the single threshold (and the average pixel values below and above the
threshold)
thresholdValue= EThresholdMode_MinResidue;
EasyImage::HistogramThreshold(&histo, thresholdValue, avgBelowThr, 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

```
////////////////////////////////////////////////////////////////////
// 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;
EImageBW8 dstImage;
```

```
// Histogram constructor
EBWHistogramVector histo;
```

```
// Variables
EBW8 lowThr;
EBW8 highThr;
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, lowThr, highThr, avgBelowThr,
avgBetweenThr, 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);
```

2.2. Arithmetic and Logic Operations

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

```
// Images constructor
EImageBW8 srcGray0, srcGray1, dstGray;
EImageC24 srcColor, dstColor;
```

```
// ...
```

```
// All images must have the same size
dstGray.SetSize(&srcGray0);
// ...
```

```
// Subtract srcGray1 from srcGray0
EasyImage::Oper(EArithmeticLogicOperation_Subtract, &srcGray0, &srcGray1, &dstGray);
```

```
// Multiply srcGray0 by a constant value
EasyImage::Oper(EArithmeticLogicOperation_Multiply, &srcGray0, (EBW8)2, &dstGray);
```

```
// Add a constant value to srcColor
EasyImage::Oper(EArithmeticLogicOperation_Add, &srcColor, EC24(128,64,196),
&dstColor);
```

```
// Erase (blacken) the destination image where the source image is black
EasyImage::Oper(EArithmeticLogicOperation_SetZero, &srcGray0, (EBW8)0, &dstGray);
```

2.3. Convolution

Pre-Defined Kernel Filtering

```
///////////////////////////////////////////////////////////
// This code snippet shows how to apply miscellaneous    //
// convolution operations based on pre-defined kernels.  //
///////////////////////////////////////////////////////////
```

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// ...
```

```
// 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::ConvolHighpass1(&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;
EImageBW8 dstImage;
```

```
// ...
```

```
// Create and define a user-defined kernel
// (Frei-Chen row gradient, positive only)
EKernel kernel;
kernel.SetKernelData(0.2929f, 0, -0.2929f,
                     0.4142f, 0, -0.4142f,
                     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);
```

2.4. Non-Linear Filtering

Morphological Filtering

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// ...
```

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

```
// Perform an erosion (3x3 square kernel)
EasyImage::ErodeBox(&srcImage, &dstImage, 1);
```

```
// Perform a dilation (5x3 rectangular kernel)
EasyImage::DilateBox(&srcImage, &dstImage, 2, 1);
```

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

Hit-and-Miss Transform

```
//////////////////////////////////////////////////////////////
// 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;
EImageBW8 dstImage;
```

```
// ...
```

```
// Create and define a Hit-and-Miss kernel
// corresponding to the left corner of a rhombus
EHitAndMissKernel leftCorner(-1, -1, 1, 1);
```

```
// Left column of the kernel
```

```
leftCorner.SetValue(-1, 0, EHitAndMissValue_Background);
```

```
// Middle column of the kernel
leftCorner.SetValue(0, -1, EHitAndMissValue_Background);
leftCorner.SetValue(0, 0, EHitAndMissValue_Foreground);
leftCorner.SetValue(0, 1, EHitAndMissValue_Background);
```

```
// Right column of the kernel
leftCorner.SetValue(1, -1, EHitAndMissValue_Foreground);
leftCorner.SetValue(1, 0, EHitAndMissValue_Foreground);
leftCorner.SetValue(1, 1, 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);
```

2.5. Vector Operations

Path Sampling

```
//////////////////////////////////////////////////////////////
// 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;
```

```
// ...
```

```
// Vector constructor
EBW8PathVector path;
```

```
// Path definition
path.Empty();
for (int i = 0; i < 100; i++)
{
```

```
  EBW8Path p;
  p.X = i;
  p.Y = i;
  p.Pixel = 128;
  path.AddElement(p);
}
```

```
// Get the image data along the path
EasyImage::ImageToPath(&srcImage, &path);
```

Profile Sampling

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// ...
```

```
// Vector constructor
EBW8Vector profile;
```

```
// Get the image data along segment (10,510)-(500,40)
EasyImage::ImageToLineSegment(&srcImage, &profile, 10, 510, 500, 40);
```

```
// Set all these points to white (255) in the image
EasyImage::LineSegmentToImage(&srcImage, 255, 10, 510, 500, 40);
```

2.6. Statistics

Image Statistics

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// ...
```

```
// Count the number of pixels above the threshold (128)
INT32 count;
EasyImage::Area(&srcImage, 128, count);
```

```
// Compute the pixels' average and standard deviation values
float stdDev, average;
EasyImage::PixelStdDev(&srcImage, stdDev, average);
```

```
// Compute the image gravity center (pixels above threshold)
float x, y;
EasyImage::GravityCenter(&srcImage, 128, x, y);
```

Sliding Windows Statistics

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage0, dstImage1;
```

```
// ...
```

```
// All images must have the same size
dstImage0.SetSize(&srcImage);
dstImage1.SetSize(&srcImage);
```

```
// Local average in a 11x11 window
EasyImage::LocalAverage(&srcImage, &dstImage0, 5, 5);
```

```
// Local deviation in a 11x11 window
EasyImage::LocalDeviation(&srcImage, &dstImage1, 5, 5);
```

Histogram-Based Statistics

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// ...
```

```
// Histogram constructor
EBWHistogramVector histo;
```

```
// 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);
```

2.7. Noise Reduction by Integration

Temporal Noise Reduction

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

```
// Images constructor
EImageBW16 noisyImage, cleanImage;
```

```
// 16 bits work image used as an accumulator
EImageBW16 store;
```

```
// ...
```

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

```
// Clear the accumulator image
EasyImage::Oper(EArithmeticLogicOperation_Copy, (EBW16)0, &store);
```

```
// Accumulation loop
int n;
for (n=0; n < 10; 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
EasyImage::Oper(EArithmeticLogicOperation_Divide, &store, (EBW16)n, &cleanImage);
```

Recursive Average

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

```
// Images constructor
EImageBW8 noisyImage, cleanImage;
```

```
// 16 bits work image used as an accumulator
EImageBW16 store;
```

```
// ...
```

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

```
// Clear the accumulator image
EasyImage::Oper(EArithmeticLogicOperation_Copy, (EBW16)0, &store);
```

```
// Prepare the transfer lookup table (reduction factor = 3)
EBW16Vector lut;
EasyImage::SetRecursiveAverageLUT(&lut, 3.f);
```

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

2.8. Feature Point Detectors

Harris Corner Detector

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// ...
```

```
// Harris corner detector
EHarrisCornerDetector harris;
EHarrisInterestPoints interestPoints;
harris.SetIntegrationScale(2.f);
```

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

```
// Retrieve the number of corners
unsigned int index = interestPoints.GetPointCount();
```

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

Canny Edge Detector

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// ...
```

```
// Canny edge detector
ECannyEdgeDetector canny;
```

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

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

2.9. Using Flexible Masks

Computing Pixels Average

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// ...
```

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

3. EasyColor

3.1. Colorimetric Systems Conversion

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

```
// Images constructor
EImageC24 srcImage;
EImageC24 dstImage;
```

```
// ...
```

```
// Prepare a lookup table for
// the RGB to La*b* conversion
EColorLookup lookup;
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);
```

3.2. Color Components

```
//////////////////////////////////////////////////////////
// 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, green, blue;
EImageC24 colorImage;
EImageBW8 luminance;
```

```
// ...
```

```
// 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;
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);
```

3.3. White Balance

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

```
// Images constructor
EImageC24 srcImage, dstImage;
EImageC24 whiteRef;
```

```
// ...
```

```
// Create a lookup table
EColorLookup lut;
```

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

```
// Prepare the lookup table for
// a white balance operation
lut.WhiteBalance(1.00f, EasyColor::GetCompensateNtscGamma(), r, g, b);
```

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

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

3.4. Pseudo-Coloring

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

```
// Images constructor
EImageBW8 srcImage;
EImageC24 dstImage;
```

```
// ...
```

```
// Create a pseudo-color lookup table
EPseudoColorLookup pcLut;
```

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

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

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

3.5. Bayer Pattern Decoding

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

```
// Images constructor
EImageBW8 bayerImage;
EImageC24 dstImage;
```

```
// ...
```

```
// 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);
```

4. EasyDeepLearning

4.1. 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;
EClassificationDataset trainingDataset;
EClassificationDataset validationDataset;
EClassifier classifier;
```

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

```
// Enabling data augmentation on the dataset
dataset.SetEnableDataAugmentation(true);
```

```
// Rotation of up to 90°
dataset.SetMaxRotationAngle(90);
```

```
// Enabling horizontal flips
dataset.SetEnableHorizontalFlip(true);
```

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

```
// 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.GetBestEpoch());
```

4.2. 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;
EImageBW8 srcImage;
```

```
// String and probability for the most probable result
std::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.GetBestLabel();
probability = result.GetBestProbability();
```

5. EasyObject

5.1. Constructing the Blobs

Image Encoder

```
///////////////////////////////////////////////////////////////
// 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;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// 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 (110)
Segmenters::EGrayscaleSingleThresholdSegmenter& segmenter=
encoder.GetGrayscaleSingleThresholdSegmenter();
segmenter.SetBlackLayerEncoded(true);
segmenter.SetWhiteLayerEncoded(false);
```

```
segmenter.SetMode(EGrayscaleSingleThreshold_Absolute);
segmenter.SetAbsoluteThreshold(110);
```

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

Image Segmenter

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Set the segmentation method to GrayscaleDoubleThreshold
encoder.SetSegmentationMethod(ESegmentationMethod_GrayscaleDoubleThreshold);
```

```
// Retrieve the segmenter object
Segmenters::EGrayscaleDoubleThresholdSegmenter& segmenter=
encoder.GetGrayscaleDoubleThresholdSegmenter();
```

```
// Set the high and low threshold values
segmenter.SetHighThreshold(150);
segmenter.SetLowThreshold(50);
```

```
// Specify the layers to be encoded (neutral layer only)
segmenter.SetBlackLayerEncoded(false);
segmenter.SetNeutralLayerEncoded(true);
segmenter.SetWhiteLayerEncoded(false);
```

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

Holes Extraction

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

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

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

```
        // Browse the holes of the current object
        for (unsigned int holeIndex = 0; holeIndex < blob.GetHoleCount(); holeIndex++)
        {
                // Retrieve a given hole
                EHole& hole = blob.GetHole(holeIndex);
        }
}
```

Continuous Mode

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// Enable the continuous mode
encoder.SetContinuousModeEnabled(true);
```

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

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

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

5.2. Computing Blobs Features

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

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

```
for (unsigned int index = 0; index < codedImage.GetObjCount(); index++)
{
        // Retrieve the selected blob gravity center
        EObject& blob = codedImage.GetObj(index);
        float centerX = blob.GetGravityCenter().GetX();
        float centerY = blob.GetGravityCenter().GetY();
}
```

5.3. Selecting and Sorting Blobs

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

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

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

```
// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature_Area, 100, ESingleThresholdMode_
Less);
```

```
// Retrieve the number of remaining blobs
unsigned int numBlobs= selection.GetElementCount();
```

```
// Sort the remaining blobs based on their area
selection.Sort(EFeature_Area, ESortDirection_Ascending);
```

```
// Retrieve the selected blobs
for (unsigned int index = 0; index < numBlobs; index++)
{
    float centerX= selection.GetElement(index).GetGravityCenterX();
    float centerY= selection.GetElement(index).GetGravityCenterY();
}
```

5.4. Using Flexible Masks

Constructing Blobs

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// 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

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// 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, 1);
```

Generating a Flexible Mask from a Blob Selection

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 mask;
```

```
// Image encoder
EImageEncoder encoder;
```

```
// Coded image
ECodedImage2 codedImage;
```

```
// ...
```

```
// 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;
selection.AddObjects(codedImage);
```

```
// Remove the Small blobs
selection.RemoveUsingUnsignedIntegerFeature(EFeature_Area, 100, 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);
```

6. EasyMatch

6.1. Pattern Learning

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// ROI constructor
EROIBW8 pattern;
```

```
// EMatcher constructor
EMatcher matcher;
```

```
// ...
```

```
// Attach the ROI to the source image
// and set its position
pattern.Attach(&srcImage);
pattern.SetPlacement(214, 52, 200, 200);
```

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

6.2. Setting Search Parameters

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

```
// Image constructor
EImageBW8 pattern;
```

```
// EMatcher constructor
EMatcher matcher;
```

```
// ...
```

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

```
// Set the maximum number of occurrences
matcher.SetMaxPositions(5);
```

```
// Set the rotation tolerances
matcher.SetMinAngle(-20.f);
matcher.SetMaxScale(20.f);
```

```
// Enable sub-pixel accuracy
matcher.SetInterpolate(true);
```

```
// Set the minimum score
matcher.SetMinScore(0.70f);
```

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

6.3. Pattern Matching and Retrieving Results

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// EMatcher constructor
EMatcher matcher;
```

```
// ...
```

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

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

```
// Retrieve the number of occurrences
int numOccurrences= matcher.GetNumPositions();
```

```
// 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;
```

7. EasyFind

7.1. Pattern Learning

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// ROI constructor
EROIBW8 pattern;
```

```
// EPatternFinder constructor
EPatternFinder finder;
```

```
// ...
```

```
// Attach the ROI to the source image
// and set its position
pattern.Attach(&srcImage);
pattern.SetPlacement(214, 52, 200, 200);
```

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

7.2. Setting Search Parameters

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

```
// Image constructor
EImageBW8 pattern;
```

```
// EPatternFinder constructor
EPatternFinder finder;
```

```
// ...
```

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

```
// Set the maximum number of occurrences
finder.SetMaxInstances(5);
```

```
// Set the rotation tolerances
finder.SetAngleTolerance(20.f);
```

```
// Set the minimum score
finder.SetMinScore(0.70f);
```

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

7.3. Pattern Finding and Retrieving Results

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// EPatternFinder constructor
EPatternFinder finder;
```

```
// EFoundPattern constructor
std::vector<EFoundPattern> foundPattern;
```

```
// ...
```

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

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

```
// Retrieve the number of instances
int numInstances= foundPattern.size();
```

```
// Retrieve the score and the
// position of the first instance
float score= foundPattern[0].GetScore();
float centerX= foundPattern[0].GetCenter().GetX();
float centerY= foundPattern[0].GetCenter().GetY();
```

8. EasyGauge

8.1. Point Location

```
//////////////////////////////////////////////////////////////////
// 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;
```

```
// EPointGauge constructor
EPointGauge pointGauge;
```

```
// Adjust the transition parameters
pointGauge.SetTransitionType(ETransitionType_Wb);
pointGauge.SetTransitionChoice(ETransitionChoice_Closest);
```

```
// Set the gauge nominal position
pointGauge.SetCenterXY(256.f, 256.f);
```

```
// Set the gauge length to 100 units and the angle to 45°
pointGauge.SetTolerance(100.f, 45.f);
```

```
// Measure
pointGauge.Measure(&srcImage);
```

```
// Get the measured point coordinates
float measuredX = pointGauge.GetMeasuredPoint().GetX();
float measuredY = pointGauge.GetMeasuredPoint().GetY();
```

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

8.2. Line Fitting

```
////////////////////////////////////////////////////////////////////
// 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;
```

```
// ELineGauge constructor
ELineGauge lineGauge;
```

```
// Adjust the transition parameters
lineGauge.SetTransitionType(ETransitionType_Bw);
lineGauge.SetTransitionChoice(ETransitionChoice_NthFromEnd);
lineGauge.SetTransitionIndex(2);
```

```
// Set the line fitting gauge position,
// length (50 units) and orientation (20°)
EPoint center(256.f, 256.f);
ELine line(center, 50.f, 20.f);
lineGauge.SetLine(line);
```

```
// Measure
lineGauge.Measure(&srcImage);
```

```
// Get the origin and end point coordinates of the fitted line
EPoint originPoint = lineGauge.GetMeasuredLine().GetOrg();
EPoint endPoint = lineGauge.GetMeasuredLine().GetEnd();
```

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

8.3. Circle Fitting

```
//////////////////////////////////////////////////////////////////////
// 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;
```

```
// ECircleGauge constructor
ECircleGauge circleGauge;
```

```
// Adjust the transition parameters
circleGauge.SetTransitionType(ETransitionType_Bw);
circleGauge.SetTransitionChoice(ETransitionChoice_LargestAmplitude);
```

```
// Set the Circle fitting gauge position, diameter (50 units),
// starting angle (10°), and amplitude (270°)
EPoint center(256.f, 256.f);
ECircle circle(center, 50.f, 10.f, 270.f);
circleGauge.SetCircle(circle);
```

```
// Measure
circleGauge.Measure(&srcImage);
```

```
// Get the center point coordinates and the radius of the fitted circle
float centerX = circleGauge.GetMeasuredCircle().GetCenter().GetX();
float centerY = circleGauge.GetMeasuredCircle().GetCenter().GetY();
float radius = circleGauge.GetMeasuredCircle().GetRadius();
```

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

8.4. Rectangle Fitting

```
/////////////////////////////////////////////////////////////////////////
// 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;
```

```
// ERectangleGauge constructor
ERectangleGauge rectangleGauge;
```

```
// Adjust the transition parameters
rectangleGauge.SetTransitionType(ETransitionType_Bw);
rectangleGauge.SetTransitionChoice(ETransitionChoice_LargestAmplitude);
```

```
// Set the rectangle fitting gauge position,
// size (50x30 units) and orientation (15°)
EPoint center(256.f, 256.f);
ERectangle rectangle(center, 50.f, 30.f, 15.f);
rectangleGauge.SetRectangle(rectangle);
```

```
// Measure
rectangleGauge.Measure(&srcImage);
```

```
// Get the size and the rotation angle of the fitted rectangle
float sizeX = rectangleGauge.GetMeasuredRectangle().GetSizeX();
float sizeY = rectangleGauge.GetMeasuredRectangle().GetSizeY();
float angle = rectangleGauge.GetMeasuredRectangle().GetAngle();
```

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

8.5. Wedge Fitting

```
/////////////////////////////////////////////////////////////////////
// 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;
```

```
// EWedgeGauge constructor
EWedgeGauge wedgeGauge;
```

```
// Adjust the transition parameters
wedgeGauge.SetTransitionType(ETransitionType_Bw);
wedgeGauge.SetTransitionChoice(ETransitionChoice_NthFromBegin);
wedgeGauge.SetTransitionIndex(0);
```

```
// Set the wedge fitting gauge position, diameter (50 units),
// breadth (-25 units), starting angle (0°) and amplitude (270°)
EPoint center(256.f, 256.f);
```

```
EWedge wedge(center, 50.f, -25.f, 0.f, 270.f);
wedgeGauge.SetWedge(wedge);
```

```
// Measure
wedgeGauge.Measure(&srcImage);
```

```
// Get the inner and outer radius of the fitted wedge
float innerRadius = wedgeGauge.GetMeasuredWedge().GetInnerRadius();
float outerRadius = wedgeGauge.GetMeasuredWedge().GetOuterRadius();
```

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

8.6. Gauge Grouping

Gauge Hierarchy

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

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// Gauges constructor
ERectangleGauge rectangleGauge;
ECircleGauge circleGauge1, circleGauge2;
```

```
// ...
```

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

```
// Attach the circle gauges to the rectangle gauge
circleGauge1.Attach(&rectangleGauge);
circleGauge2.Attach(&rectangleGauge);
```

```
// Set the first circle gauge name
circleGauge1.SetName("myCircleGauge1");
```

```
// ...
```

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

Complex Measurement

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

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

```
// Retrieve the number of worldShape's daughters
int numDaughters= worldShape.GetNumDaughters();
```

```
// ...
```

```
// 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->GetMeasuredRectangle().GetSizeX();
```

```
// Retrieve the measurement result of a
// daughter gauge called "myCircleGauge1"
ECircleGauge* circleGauge= (ECircleGauge*)worldShape.GetShapeNamed
("myCircleGauge1");
EPoint center= circleGauge->GetMeasuredCircle().GetCenter();
```

8.7. Calibration using EWorldShape

Calibration by Guesswork

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// ...
```

```
// Compute the calibration coefficients
// Field of view: 32x24 mm
worldShape.SetSensor(srcImage.GetWidth(), srcImage.GetHeight(), 32.f, 24.f);
```

```
// Retrieve the spatial resolution
float resolutionX= worldShape.GetXResolution();
float resolutionY= worldShape.GetYResolution();
```

Landmark-Based Calibration

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

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// ...
```

```
// 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
        // ...
```

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

```
// Perform the calibration
worldShape.Calibrate(ECalibrationMode_Skewed);
```

Dot Grid-Based Calibration

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

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// ...
```

```
// 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
        // ...
```

```
        // 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

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

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// EPoint constructor
EPoint sensor;
EPoint world;
```

```
// ...
```

```
// Perform the calibration
worldShape.Calibrate(ECalibrationMode_Scaled | 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

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

```
// Images constructor
EImageBW8 srcImage;
EImageBW8 dstImage;
```

```
// EWorldShape constructor
EWorldShape worldShape;
```

```
// Lookup table constructor
EUnwarpingLut lut;
```

```
// ...
```

```
// Perform the calibration
worldShape.Calibrate(ECalibrationMode_Tilted | ECalibrationMode_Radial);
```

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

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

9. EasyOCR

9.1. Learning Characters

```
//////////////////////////////////////////////////////
// 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;
```

```
// EOCR constructor
EOCR ocr;
```

```
// Text to be learned (all digits)
// Assuming the image contains this text
const std::string text= "0123456789";
```

```
// ...
```

```
// Create a new fon
ocr.NewFont(8, 11);
```

```
// Adjust the segmentation parameters
ocr.SetTextColor(EOCRColor_BlackOnWhite);
ocr.SetMinCharWidth(15);
ocr.SetMinCharWidth(50);
ocr.SetMinCharHeight(15);
ocr.SetMinCharHeight(75);
ocr.SetNoiseArea(15);
```

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

```
// Learn the characters
ocr.LearnPatterns(&srcImage, text, EOCRClass_Digit);
```

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

9.2. Recognizing Characters

```
///////////////////////////////////////////////////////
// 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;
```

```
// EOCR constructor
EOCR ocr;
```

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

```
// ...
```

```
// Recognize the characters
std::string text= ocr.Recognize(&srcImage, 10, EOCRClass_AllClasses);
```

```
// Alternatively
// Define the character filter (2 letters and 3 digits)
std::vector<UINT32> charFilter;
charFilter.push_back(EOCRClass_UpperCase);
charFilter.push_back(EOCRClass_UpperCase);
charFilter.push_back(EOCRClass_Digit);
charFilter.push_back(EOCRClass_Digit);
charFilter.push_back(EOCRClass_Digit);
```

```
// Recognize the characters with class filtering
text= ocr.Recognize(&srcImage, 10, charFilter);
```

10. EasyOCR2

10.1. 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;
image.Load("image.tif");
```

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

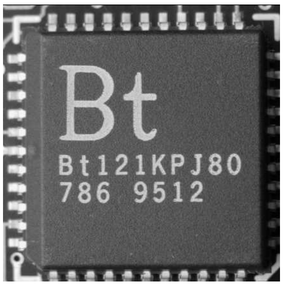
```
// Create an EOCR2 instance
EOCR2 ocr2;
```

```
// Set the expected character sizes
ocr2.SetCharsWidthRange(EIntegerRange(25,25));
ocr2.SetCharsHeight(37);
```

```
// Set the text polarity, in this case WhiteOnBlack
ocr2.SetTextPolarity(EasyOCR2TextPolarity_WhiteOnBlack);
```

```
// Set the topology
ocr2.SetTopology(".{10}\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);
```



The image used in this code snippet

10.2. 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;
image.Load("image.tif");
```

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

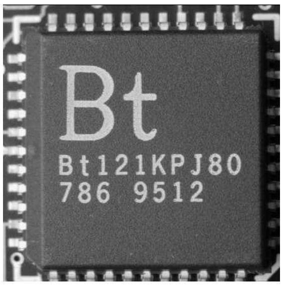
```
// Create an EOCR2 instance
EOCR2 ocr2;
```

```
// Set the required parameters
ocr2.SetCharsWidthRange(EIntegerRange(25,25));
ocr2.SetCharsHeight(37);
ocr2.SetTextPolarity(EasyOCR2TextPolarity_WhiteOnBlack);
ocr2.SetTopology(".{10}\n.{3} .{4}");
```

```
// Learn from the reference image:
//   1) Detect the text in the image
EOCR2Text text = ocr2.Detect(roi);
//   2) Set the true values of the text
text.SetText("Bt121KPJ80\n786 9512");
//   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");
```



The image used in this code snippet

10.3. 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;
image.Load("image.tif");
```

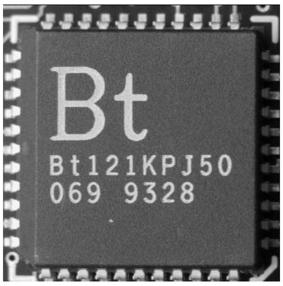
```
// Attach an ROI
EROIBW8 roi;
roi.Attach(&src, 50, 224, 340, 96);
```

```
// Create an EOCR2 instance
EOCR2 ocr2;
```

```
// Set the required parameters
ocr2.SetCharsWidthRange(EIntegerRange(25,25));
ocr2.SetCharsHeight(37);
ocr2.SetTopology("[LN]{10}\nN{3} N{4}");
ocr2.SetTextPolarity(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
std::string result = ocr2.Read(roi);
```



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;
image.Load("image.tif");
```

```
// Attach an ROI
EROIBW8 roi;
roi.Attach(&src, 50, 224, 340, 96);
```

```
// Create an EOCR2 instance
EOCR2 ocr2;
```

```
// Set the required parameters
ocr2.SetCharsWidthRange(EIntegerRange(25,25));
ocr2.SetCharsHeight(37);
ocr2.SetTopology("[LN]{10}\nN{3} N{4}");
ocr2.SetTextPolarity(EasyOCR2TextPolarity_WhiteOnBlack);
```

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

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

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;
image.Load("image.tif");
```

```
// Attach an ROI
EROIBW8 roi;
roi.Attach(&src, 50, 224, 340, 96);
```

```
// Create an EOCR2 instance
EOCR2 ocr2;
```

```
// 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
std::string result = ocr2.Read(roi);
```

11. EasyOCV

11.1. Creating an OCV Model

```
////////////////////////////////////////////////////////
// This code snippet shows how to create an OCV model //
// from a golden template and save it into a file.    //
////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EOCV constructor
EOCV ocv;
```

```
// ECodedImage constructor
ECodedImage blobs;
```

```
// ...
```

```
// Reset the OCV context
ocv.DeleteTemplateTexts();
ocv.DrawTemplateChars();
ocv.DeleteTemplateObjects();
ocv.ClearStatistics();
```

```
// Set the OCV context
ocv.SetTemplateImage(&srcImage);
```

```
// Segment the source image
blobs.SetThreshold(EThresholdMode_MinResidue);
blobs.BuildObjects(&srcImage);
```

```
// Compute blobs area and unselect small objects
blobs.AnalyseObjects(ELegacyFeature_Area);
blobs.SelectObjectsUsingFeature(ELegacyFeature_Area, 0, 50, ESelectOption_
RemoveLesserOrEqual);
```

```
// Add remaining blobs to the OCV context
ocv.CreateTemplateObjects(&blobs);
```

```
// Add all selected free objects
ocv.CreateTemplateChars(ESelectionFlag_True, ECharCreationMode_Separate);
```

```
// Group all selected free characters in a single text
ocv.CreateTemplateTexts();
```

```
// Perform the learning
ocv.Learn(&srcImage);
```

```
// Save the ocv model into a file
```

```
ocv.Save("myModel.ocv");
```

11.2. Inspecting

```
//////////////////////////////////////////////////////
// This code snippet shows how to load an OCV model //
// file and perform an inspection.                  //
//////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EOCV constructor
EOCV ocv;
```

```
// ...
```

```
// Load an EasyOCV model file
ocv.Load("myModel.ocv");
```

```
// Perform the inspection
ocv.Inspect(&srcImage, EThresholdMode_MinResidue);
```

11.3. Setting Inspection Parameters

```
///////////////////////////////////////////////////
// This code snippet shows how to set characters //
// and texts inspection parameters.              //
///////////////////////////////////////////////////
```

```
// EOCV constructor
EOCV ocv;
```

```
// Temporary EOCVText object for parameters modification
EOCVText text;
```

```
// Reset the text parameters
text.ResetParameters();
```

```
// Set the text shift tolerance
text.SetShiftXTolerance(30);
text.SetShiftYTolerance(20);
```

```
// Apply the new parameters to all the texts of the ocv context
ocv.ScatterTextsParameters(text, ESelectionFlag_Any);
```

```
// Retrieve the first text (index 0) parameters
text.ResetParameters();
ocv.GetTextParameters(text, 0);
```

```
// Double the shift tolerance
text.SetShiftXTolerance(text.GetShiftXTolerance() * 2);
text.SetShiftYTolerance(text.GetShiftYTolerance() * 2);
```

```
// Apply the new parameters to the ocv context first text only
ocv.SetTextParameters(text, 0);
```

```
// Temporary OCVChar object for parameters modification
EOCVChar ch;
```

```
// Reset the character parameters
ch.ResetParameters();
```

```
// Set the character shift tolerance
ch.SetShiftXTolerance(15);
ch.SetShiftYTolerance(10);
```

```
// Apply the new parameters to all the characters of the ocv context
ocv.ScatterTextsCharsParameters(ch, ESelectionFlag_Any, ESelectionFlag_True);
```

11.4. Retrieving Diagnostics

```
/////////////////////////////////////////////////////////////
// This code snippet shows how to perform an inspection //
// and retrieve the diagnostics.                         //
/////////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EOCV constructor
EOCV ocv;
```

```
// ...
```

```
// Load an EasyOCV model file
ocv.Load("myModel.ocv");
```

```
// Perform the inspection
ocv.Inspect(&srcImage, EThresholdMode_MinResidue);
```

```
// Retrieve the OCV inspection diagnostics
if(ocv.GetDiagnostics() != EDiagnostic_Undefined)
{
```

```
        // Check if texts have been found
        bool bTextNotFound= ((ocv.GetDiagnostics() & EDiagnostic_TextNotFound) > 0);
```

```
        // Check if there is text mismatch
        bool bTextMismatch= ((ocv.GetDiagnostics() & EDiagnostic_TextMismatch) > 0);
```

```
        // Check if there is text overprinting
        bool bTextOverprinting= ((ocv.GetDiagnostics() & EDiagnostic_TextOverprinting) > 0);
```

```
        // Check if there is text underprinting
        bool bTextUnderprinting= ((ocv.GetDiagnostics() & EDiagnostic_TextUnderprinting) >
0);
```

```
        // Check if characters have been found
        bool bCharNotFound= ((ocv.GetDiagnostics() & EDiagnostic_CharNotFound) > 0);
```

```
        // Check if there is character mismatch
        bool bCharMismatch= ((ocv.GetDiagnostics() & EDiagnostic_CharMismatch) > 0);
```

```
        // Check if there is character overprinting
        bool bCharOverprinting= ((ocv.GetDiagnostics() & EDiagnostic_CharOverprinting) > 0);
```

```
        // Check if there is character underprinting
        bool bCharUnderprinting= ((ocv.GetDiagnostics() & EDiagnostic_CharUnderprinting) >
0);
}
```

11.5. Statistical Learning

```
///////////////////////////////////////////////////////////
// This code snippet shows how to perform a statistical  //
// learning based on several good quality templates.     //
///////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// EOCV constructor
EOCV ocv;
```

```
// ...
```

```
// Clear the statistics
ocv.ClearStatistics();
```

```
// Loop on the number of good quality sample images
for(int i= 0; i < numSampleImages; i++)
{
        // acquire the next sample image into srcImage
        // ...
```

```
        // Perform the inspection
        ocv.Inspect(&srcImage, EThresholdMode_MinResidue);
```

```
        // Update the statistics
        ocv.UpdateStatistics();
}
```

```
// Adjust the tolerance values based on
// the inspected good quality sample images
ocv.AdjustTextsQualityRanges(3.3f, ESelectionFlag_Any);
ocv.AdjustTextsQualityRanges(3.3f, ESelectionFlag_Any, ESelectionFlag_Any);
```

12. EasyBarCode

12.1. Reading a Bar Code

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Bar code reader constructor
EBarCode reader;
```

```
// String for the decoded bar code
std::string result;
```

```
// ...
```

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

12.2. Reading a Bar Code Following a Given Symbology

```
///////////////////////////////////////////////////////////////
// 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;
```

```
// Bar code reader constructor
EBarCode reader;
```

```
// String for the decoded bar code
std::string result;
```

```
// ...
```

```
// Disable all standard symbologies
reader.SetStandardSymbologies(0);
```

```
// Enable the Code32 symbology only
reader.SetAdditionalSymbologies(ESymbologies_Code32);
```

```
// Enable checksum verification
reader.SetVerifyChecksum(true);
```

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

```
// Retrieve the number of symbologies for
// which the decoding process was successful
int numDecoded = reader.GetNumDecodedSymbologies();
```

```
if(numDecoded > 0)
{
        // Decode the bar code according to the Code32 symbology
        result = reader.Decode(ESymbologies_Code32);
}
```

12.3. Reading a Bar Code of Known Location

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Bar code reader constructor
EBarCode reader;
```

```
// String for the decoded bar code
std::string result;
```

```
// ...
```

```
// Disable automatic bar code detection
reader.SetKnownLocation(TRUE);
```

```
// Set the bar code position
reader.SetCenterXY(450.0f, 400.0f)
reader.SetSize(250.0f, 110.0f);
reader.SetReadingSize(1.15f, 0.5f);
```

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

12.4. Reading a Mail Bar Code

```
////////////////////////////////////////////////////////
// This code snippet shows how to read Mail Barcodes  //
// and retrieve the decoded data.                     //
////////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// Mail barcode reader constructor
EMailBarcodeReader reader;
```

```
// Select expected symbologies and orientations (optional)
reader.SetExpectedSymbologies(...);
reader.SetExpectedOrientations(...);
```

```
// ...
```

```
// Read
std::vector<EMailBarcode> codes = reader.Read(srcImage);
```

```
// Retrieve the data included in found mail barcodes
for (unsigned int index= 0; index < codes.size(); index++)
{
    std::string text = codes[index].GetText();
    std::vector<EStringPair> components = codes[index]. GetComponentStrings();
}
```

13. EasyMatrixCode

13.1. Automatic Reading

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Matrix code reader constructor
EMatrixCodeReader reader;
```

```
// Matrix code constructor
EMatrixCode mxCode;
```

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

```
// ...
```

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

```
// Retrieve the decoded string
result = mxCode.GetDecodedString();
```

13.2. Reading with Prior Learning

```
//////////////////////////////////////////////////////////////
// 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;
EImageBW8 srcImage;
```

```
// Matrix code reader constructor
EMatrixCodeReader reader;
```

```
// Matrix code constructor
EMatrixCode mxCode;
```

```
// String for the decoded information
std::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.GetDecodedString();
```

13.3. Advanced Tuning of the Search Parameters

```
////////////////////////////////////////////////////////////////
// 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;
```

```
// Matrix code reader constructor
EMatrixCodeReader reader;
```

```
// Matrix code constructor
EMatrixCode mxCode;
```

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

```
// ...
```

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

```
// Add the 15x15 and 17x17 logical sizes
reader.GetSearchParams().AddLogicalSize(ELogicalSize__15x15);
reader.GetSearchParams().AddLogicalSize(ELogicalSize__17x17);
```

```
// Remove the default families
reader.GetSearchParams().ClearFamily();
```

```
// Add the ECC050 family
reader.GetSearchParams().AddFamily(EFamily_ECC050);
```

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

```
// Retrieve the decoded string
result = mxCode.GetDecodedString();
```

13.4. Retrieving Print Quality Grading

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// Matrix code reader constructor
EMatrixCodeReader reader;
```

```
// Matrix code constructor
EMatrixCode mxCode;
```

```
// ...
```

```
// Enable grading computation
reader.SetComputeGrading(TRUE);
```

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

```
// Retrieve the print quality grading
int axialNonUniformityGrade= mxCode.GetAxialNonUniformityGrade();
int contrastGrade= mxCode.GetContrastGrade();
int printGrowthGrade= mxCode.GetPrintGrowthGrade();
int unusedErrorCorrectionGrade= mxCode.GetUnusedErrorCorrectionGrade();
```

14. EasyQRCode

14.1. Automatic Reading of a QR Code

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// QR code reader constructor
EQRCodeReader reader;
```

```
// ...
```

```
// Set the source image
reader.SetSearchField(srcImage);
```

```
// Read
std::vector<EQRCode> qrCodes = reader.Read();
```

```
// Retrieve the data of the first QR code found if
// one was found and decoding went ok
if ((qrCodes.size() > 0) &&
   (qrCodes[0].GetUnusedErrorCorrection() >= 0))
{
    EQRCodeDecodedStream stream = qrCodes[0].GetDecodedStream();
}
```

14.2. Retrieving Information of a QR Code

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

```
// Image constructor
EImageBW8 srcImage;
```

```
// QR code reader constructor
EQRCodeReader reader;
```

```
// ...
// Set the source image
reader.SetSearchField(srcImage);
```

```
// Read
std::vector<EQRCode> qrCodes = reader.Read();
```

```
// Retrieve version, model and position information
// of the first QR code found, if one was found
if (qrCodes.size() > 0)
{
    int version = qrCodes[0].GetVersion();
    EQRCodeModel model = qrCodes[0].GetModel();
    EQRCodeGeometry geometry = qrCodes[0].GetGeometry();
}
```

14.3. Decoding the First QR Code Detected

```
/////////////////////////////////////////////////////
// This code snippet shows how to decode a QR code //
// from a list of detected ones.                   //
/////////////////////////////////////////////////////
```

```
// Image constructor
EImageBW8 srcImage;
```

```
// QR code reader constructor
EQRCodeReader reader;
```

```
// ...
```

```
// Set the source image
reader.SetSearchField(srcImage);
```

```
// Detect QR Codes
std::vector<EQRCodeGeometry> qrCodeGeometries = reader.Detect();
```

```
// Decode first QR Code
EQRCode qrCode = reader.Decode(qrCodeGeometries[0]);
```

```
// Retrieve the data from the QR Code
EQRCodeDecodedStream stream = qrCode.GetDecodedStream();
```

14.4. Tuning the Search Parameters

```
////////////////////////////////////////////////////
// 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;
```

```
// QR code reader constructor
EQRCodeReader reader;
```

```
// ...
```

```
// Set the source image
reader.SetSearchField(srcImage);
```

```
// Set the search parameters
reader.SetMaximumVersion(7);
reader.SetMinimumIsotropy(0.9f);
```

```
// Set the searched models
std::vector<EQRCodeModel> models;
models.push_back(EQRCodeModel_Model2);
reader.SetSearchedModels(models);
```

```
// Read
std::vector<EQRCode> qrCodes = reader.Read();
```

```
// Retrieve the data of the first QR code found
EQRCodeDecodedStream stream = qrCodes[0].GetDecodedStream();
```