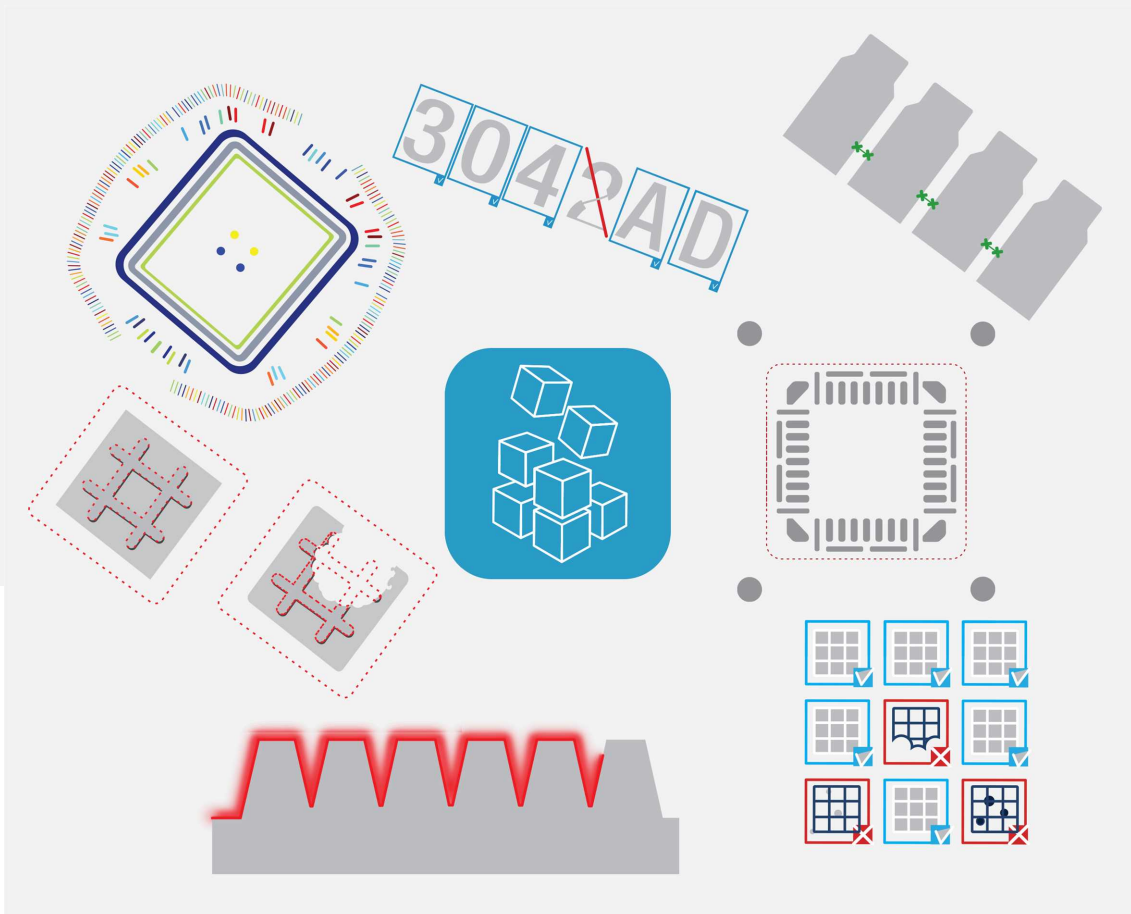EURESYS™
Excellence in vision

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.

Contents

PART I
OPEN EVISION BASICS

1. Solving a Vision Problem

A typical vision-based application follows these steps:

1. Acquire images

External images must be stored in a host PC memory buffer with a known address. They must be linked to the Open eVision image object with parameters containing the address and pitch (32 bytes default) of the buffer, and the width and height of the image.
For details see Image Construction and Memory Allocation.

To optimize resolution and repeatability, the area of interest should occupy most of the field of view, and the targeted inspection equipment should be simulated as accurately as possible, with realistic lighting.

You should create two series of images:

■ **objects without defects** (accepted by the inspection process) in all situations (such as unstable lighting conditions and movement freedom).

■ **objects with unacceptable defects** (to be rejected by the inspection process).

Use Open eVision Libraries to process the images:

2. Pre-process images

Reduce defects or enhance properties (such as contrast between the object of interest and background).
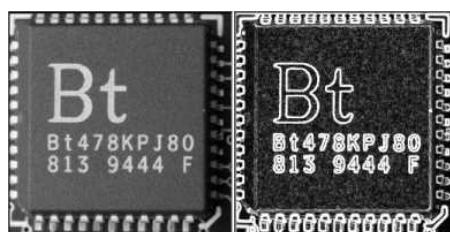
3. Locate objects of interest

Locate the area of interest using techniques such as segmentation, edge detection or pattern matching.

4. Analyse your findings

EasyObject measures unknown shapes
EasyGauge returns accurate measurements of known objects
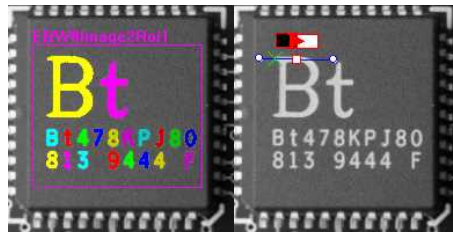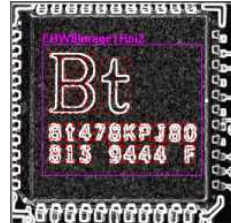EasyImage provides statistical parameters

Measurements are performed locally so quantify the shape of objects geometrically, defects can be related to abnormal gray-level values.

Area	Limit Center Y	Limit Width
252	29.00	31.00
242	60.00	31.00
535	48.00	34.00
654	44.00	54.00
246	101.00	15.00
140	101.50	11.00
180	101.00	15.00
158	101.00	14.00
235	101.50	16.00
214	101.00	15.00
197	101.00	14.00
157	101.50	13.00
226	101.50	15.00

5. Optimize your application

Your solution must reliably separate good images from bad images by comparing ROI features to assess quality, detect defects, and recognize and sort objects.
If it can't, repeat steps 2 and 3 to improve your set of features.

Develop thread safe applications

Open eVision supports simultaneous execution by multiple (unlimited) threads on the same CPU, but data can only be accessed by one thread at a time
So independent tasks can execute simultaneously in your application, but each bit of shared data must be controlled by a separate task.

Rules for Thread-Safe Developments

The following rules avoid data corruption, crashes and misbehaving programs.

Thread-safe basic types classes

Basic types	Recommendations	Restrictions
Basic pixel structures EColor, EPeak, EISH, ELAB, ELCH, ELSH, ELUV, EBW1, EBW8, EBW8Path, EBW16, EBW16Path, EBW32, EC15, EC16, EC24, EC24A, EC24Path, EPath, ERGB, ERGBColor, EVSH, EXYZ, EYIQ, EYSH, EYUV, EDepth8, EDepth16 and EDepth32f		No
Pixel collection classes EColorLookup, EPseudoColorLookup, EPeakVector, EBW8Vector, EBWHistogramVector, EBW8PathVector, EBW16PathVector, EBW16Vector, EBW32Vector, EC24Vector, EPathVector, EColorVector, EColorLookup and EC24PathVector	No restrictions on read-only access.	A single instance may not be modified by several threads. If a thread is modifying an instance, no other thread can access it.
Image classes EImageBW1, EImageBW8, EImageBW16, EImageBW32, EImageC15, EImageC16, EImageC24 and EImageC24A	No restrictions on read-only access.	A single instance may not be modified by several threads. If a thread is modifying an instance, no other thread can access it.
Depth map classes EDepthMap8, EDepthMap16 and EDepthMap32f	No restrictions on read-only access.	A single instance may not be modified by several threads. If a thread is modifying an instance, no other thread can access it.

Basic types	Recommendations	Restrictions
Point cloud classes E3DPointCloud	No restrictions on read-only access.	A single instance may not be modified by several threads. If a thread is modifying an instance, no other thread can access it.
ROI classes EROIBW1, EROIBW8, EROIBW16, EROIBW32, EROIC15, EROIC16, EROIC24 and EROIC24A EDepthMapROI8, EDepthMapROI16 and EDepthMapROI32f	No restrictions on read-only access.	A single instance may not be modified by several threads. If a thread is modifying an instance, no other thread can access it. Different ROI can be added or removed from an image or moved event if their parent image is the same. Consequently, different threads can work on different areas of an image possibly changing in position and size during the process.

Thread-safe library classes

Library	Recommendations	Restrictions
EasyImage and EasyColor	Static methods from this class (provided threading rules applying to their arguments are not broken).	No
EasyObject		No
EasyMatch, EasyFind, EasyQRCode and EasyOCR2 EMatcher, EMatchPosition, EPatternFinder and EFoundPattern		A single instance cannot be accessed from several threads. Search field (read-only) can be shared by different objects.
EasyGauge and Shape subclasses		Can be attached, moved or removed from different threads, even in the same hierarchy. A single instance must not be used by different threads.

Library	Recommendations	Restrictions
Gauging classes (EPointGauge, ELineGauge, ERectangleGauge, ECircleGauge, EWedgeGauge), EWorldShape and EFrameShape)		
Basic geometric classes (EFrame, EPoint, ECircle, ELine, ERectangle and EWedge)		Can be accessed from different threads provided that an instance is not used by two different threads simultaneously.
Gauging classes measuring and processing operations	May be executed in different threads with no blocking even if these gauges perform their measuring operations in the same image. Multiple CPU usage will be optimal.	A single instance cannot be read / modified by two threads .
EWorldShape		A single instance cannot be read / modified from different threads.
EasyOCR and EasyBarCode EOCR, EOCVChar and EBarCode	Different instances may be created and used from different threads.	A single instance cannot be accessed from several threads.
EChecker	Different instances may be used from different threads.	
EasyMatrixCode EMatrixCodeReader	Multiple CPU usage will be optimal.	A single instance cannot be used by several threads. A single MatrixCode cannot be used in multiple threads.

Report errors

When an Open eVision function fails, an exception is thrown which contains an error code and a description. To catch a potential exception, the function call is included in a try-catch block.

Measure execution times

Timing a particular piece of code is achieved simply with start and stop operations:

- Start timing: `Easy::.StartTiming`
- Stop timing: `Easy::.StopTiming`
- Clock resolution: `Easy::.TrueTimingResolution`

help is available:

- *Open eVision Studio*

  Speeds up and automates the creation of a solution. You can test the functions to find appropriate parameter values, and generate code of your operations to copy and paste into prototype applications.

- *Open eVision examples*

  Euresys download area contains:
  - **sample projects**: how to use Open eVision libraries with a particular IDE.
  - **sample application programs**: how to combine Open eVision functions and libraries in a variety of combinations and applications.

2. Discovering Open eVision Libraries

The **Open eVision** libraries are a set of powerful image processing tools, tailored for use in computer vision applications. They cover most state-of-the-art techniques in digital image processing, from classical algorithms to advanced solutions ready-made for specific tasks.

Even though many of the available tools are designed to be self-consistent and easy to use, the advanced user should find everything he needs to build his own workflow by combining the numerous building blocks provided.

Open eVision is made of a set of C++, ActiveX and .NET classes designed to be integrated into your application. The libraries are in no way a closed solution and do require to be integrated in your own application, leaving you all the freedom to deal with all other aspects of the automation not related to image processing.

Foundations

- Basic Types and Operations contains the definition of fundamental objects , types, classes and functions used in all Open eVision components.
- Easy3D contains a set of tools for solving computer vision problem using 3D acquisition and processing.

Preprocessing

- EasyImage contains **gray-level image** processing functions that improve image quality and contrast between background and objects of interest as well as linear and non-linear **filtering**,and **geometric transformations**.
- EasyColor contains **color image** processing functions that efficiently convert images between several color systems.

Blob inspection

- EasyObject obtains information about distinct **objects** (blob analysis), **identifies** them using connected component labeling, then **sorts** them and **selects** with them respect to their geometric features.
- EasyObject 2 contains advanced blob detection and analysis tools.

Pattern matching

- EasyMatch **locates patterns** in an image based on a pixel-by-pixel comparison with a reference pattern or template. It can be used for image registration or component placement inspection.

- EasyFind **locates patterns** in an image based on a geometrical model from a reference pattern or template. Compared to EasyMatch, it is computationally fast,robust against noise, occlusion, blurring and illumination variations.

2D measurement

- EasyGauge **assesses dimensions** of objects to sub-pixel precision, detects edges, locates points and fits geometric models. EasyGauge can measure in physical units (mm, inch, ...) instead of pixels if the field of view is calibrated.

Mark-Inspection Libraries

- EasyOCR performs **optical character recognition**. It can be used for reading serial numbers or printed labels.
- EasyOCR2 contains **advanced** optical character recognition functions.
- Echecker **creates a Golden template and inspects images**.
- EasyBarCode **reads bar codes**.
- EasyMatrixCode and EasyMatrixCode2 **read data matrix codes**.
- EasyQRCode **detects and decodes QR codes**.

Image statistics

- EasyImage statistics contains tools for quantifying image focus, sliding window statistics and histogram analysis.

3. Dealing with Pixel Containers and Files

3.1. Pixel Container Definition

Images

Open eVision image objects contain image data that represents rectangular images.

Each image object has a data buffer, accessible via a pointer, where pixel values are stored contiguously, row by row.
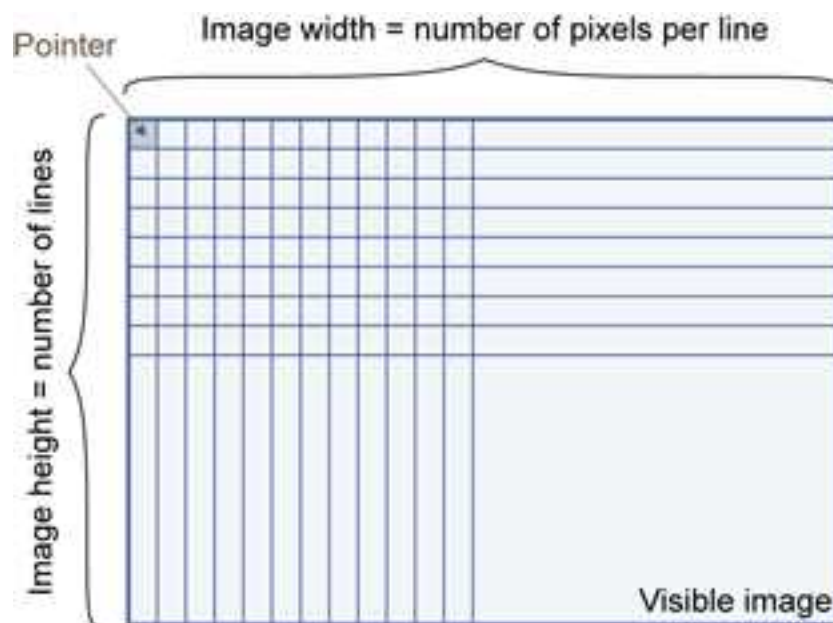

Image main parameters

An Open eVision image object has a rectangular array of pixels characterized by `EBaseROI` parameters .

- `Width` is the number of columns (pixels) per row of the image.
- `Height` is the number of rows of the image. (Maximum width / height is 32,767 (2^{15}-1) in Open eVision 32-bit, and 2,147,483,647 (2^{31}-1) in Open eVision 64-bit.)
- `Size` is the width and height.

The `Plane` parameter contains the number of color components. Gray-level images = 1. Color images = 3.

Classes

Image and ROI classes derive from abstract class `EBaseROI` and inherit all its properties.
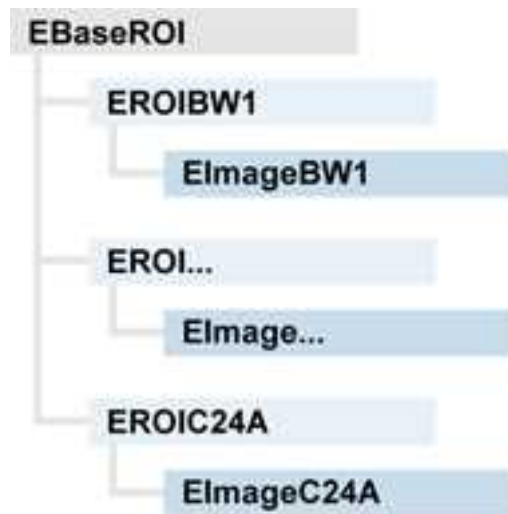
Depth maps

A depth map is way to represent a 3D object using a 2D grayscale image, each pixel in the image representing a 3D point.

The pixel coordinates are the representation of the X and Y coordinates of the point while the grayscale value of the pixel is a representation of the Z coordinate of the point.

Point clouds

A point cloud (https://en.wikipedia.org/wiki/Point_cloud) is an unstructured set of 3D points representing discrete positions on the surface of an object.
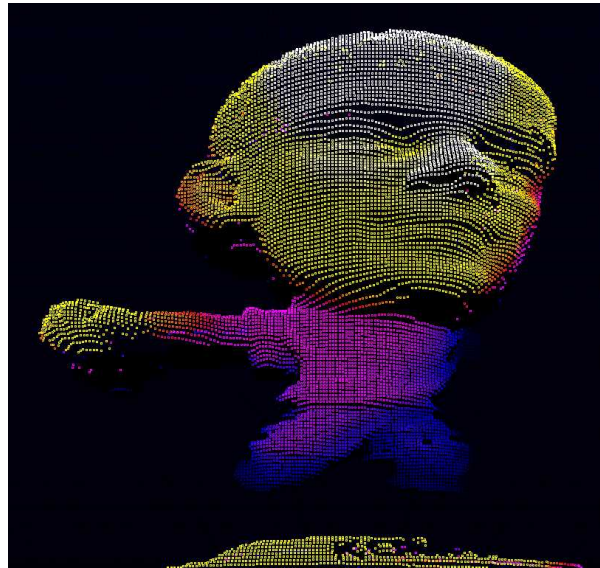
3D point clouds are produced by various 3D scanning techniques, such as Laser Triangulation, Time of Flight or Structured Lighting.

3.2. Pixel Container Types

Images

Several image types are supported according to their pixel types: black and white, gray levels, color, etc.

Easy.GetBestMatchingImageType returns the best matching image type for a given file on disk.

BW1	1-bit black and white images (8 pixels are stored in 1 byte)	`EImageBW1`
BW8	8-bit grayscale images (each pixel is stored in 1 byte)	`EImageBW8`
BW16	16-bit grayscale images (each pixel is stored in 2 bytes)	`EImageBW16`
BW32	32-bit grayscale images (each pixel is stored in 4 bytes)	`EImageBW32`
C15	15-bit color images (each pixel is stored in 2 bytes). Compatible with Microsoft® Windows RGB15 color images and MultiCam RGB15 format.	`EImageC15`

C16	16-bit color images (each pixel is stored in 2 bytes). Compatible with Microsoft® Windows RGB16 color images and MultiCam RGB16 format.	`EImageC16`
C24	C24 images store 24-bit color images (each pixel is stored in 3 bytes). Compatible with Microsoft® Windows RGB24 color images and MultiCam RGB24 format.	`EImageC24`
C24A	C24A images store 32-bit color images (each pixel is stored in 4 bytes). Compatible with Microsoft® Windows RGB32 color images and MultiCam RGB32 format.	`EImageC24A`

Depth Maps

8 and 16-bit depth map values are stored in buffers compatible with the 2D Open eVision images.

EDepth8	8-bit depth map (each pixel is stored in 1 byte as an integer)	`EDepthMap8`
EDepth16	16-bit depth map (each pixel is stored in 2 bytes as a fixed point)	`EDepthMap16`
EDepth32f	32-bit depth map (each pixel is stored in 4 bytes as a float)	`EDepthMap32f`

Point Clouds

Point Cloud	Set of points coordinates (stored as float)	`E3DPointCloud`

3.3. Supported Image File Types

Type	Description
BMP	Uncompressed image data format (Windows Bitmap Format)
JPEG	Lossy data compression standard issued by the Joint Photographic Expert Group registered as ISO/IEC 10918-1. Compression irretrievably looses quality.
JFIF	JPEG File Interchange Format

Type	Description
JPEG-2000	Data compression standard issued by the Joint Photographic Expert Group registered as ISO/IEC 15444-1 and ISO/IEC 15444-2. Open eVision supports only lossy compression format, file format and code stream variants. - **code stream** describes the image samples. - **file format** includes meta-information such as image resolution and color space.
PNG	Lossless data compression method (Portable Network Graphics).
Serialized	Euresys proprietary image file format obtained from the **serialization** of Open eVision image objects.
TIFF	Tag Image File Format is currently controlled by Adobe Systems and uses the LibTIFF third-party library to process images written for 5.0 or 6.0 TIFF specification. File **save** operations are lossless and use CCITT 1D compression for 1-bit binary pixel types and LZW compression for all others. File **load** operations support all TIFF variants listed in the LibTIFF specification.

3.4. Pixel and File Types Compatibility

Depth map to image conversion

For a 8- and 16-bit depth maps, the `AsImage()` method returns a compatible image object (respectively `EImageBW8` and `EImageBW16`) that can be used with Open eVision's 2D processing features.

Pixel and file types compatibility

Pixel access

The recommended method to access pixels is to use `SetImagePtr` and `GetImagePtr` to embed the image buffer access in your own code. See also Image Construction and Memory Allocation and Retrieving Pixel Values.

Use of the following methods should be limited because of the overhead incurred by each function call:

Direct access

`EROIBW8::.GetPixel` and `SetPixel` methods are implemented in all image and ROI classes to read and write a pixel value at given coordinates. To scan all pixels of an image, you could run a double loop on the X and Y coordinates and use `GetPixel` or `SetPixel` each iteration, but this is not recommended.

> ✓ **TIP**
>
> For performance reasons, these accessors should not be used when a significant number of pixel needs to be processed. When that is the case, retrieving the internal buffer pointer using `GetBufferPtr()` and iterating on the pointer is recommended.

Quick Access to BW8 Pixels

In BW8 images, a call to `EBW8PixelAccessor::.GetPixel` or `SetPixel` will be faster than a direct `EROIBW8::.GetPixel` or `SetPixel`.

Supported structures

- `EBW1`, `EBW8`, `EBW32`
- `EC15` (*), `EC16` (*), `EC24` (*)
- `EC24A`
- `EDepth8`, `EDepth16`, `EDepth32f`,

(*) These formats support RGB15 (5-5-5 bit packing), RGB16 (5-6-5 bit packing) and RGB32 (RGB + alpha channel) but they must be converted to/from EC24 using `EasyImage::.Convert` before any processing.

> 🗒 **NOTE**
>
> Transition with versions prior to eVision 6.5 should be seamless: image pixel types were defined using typedef of integral types, pixel values were treated as unsigned numbers and implicit conversion to/from previous types is provided.

Pixel and File Type compatibility during Load or Save operations

Type	BMP	JPEG	JPEG2000	PNG	TIFF	Serialized
BW1	Ok	N/A	N/A	Ok	Ok	Ok
BW8	Ok	Ok	Ok	Ok	Ok	Ok
BW16	N/A	N/A	Ok	Ok	Ok (***)	Ok
BW32	N/A	N/A	N/A	N/A	Ok (***)	Ok
C15	Ok	Ok (**)	Ok (**)	Ok (**)	Ok (**)	Ok
C16	Ok	Ok (**)	Ok (**)	Ok (**)	Ok (**)	Ok
C24	Ok	Ok	Ok	Ok	Ok (**)	Ok
C24A	Ok	N/A	N/A	Ok	N/A	Ok
Depth8	Ok	Ok	Ok	Ok	Ok	Ok
Depth16	N/A	N/A	Ok	Ok	Ok (***)	Ok
Depth32f	N/A	N/A	N/A	N/A	N/A	Ok

N/A: Not supported. An exception occurs if you use the combination.

Ok: Image integrity is preserved with no data loss (apart from JPEG and JPEG2000, lossy compression).

(**) C15 and C16 formats are automatically converted into C24 during the save operation.

(***) BW16 and BW32 are not supported by Baseline TIFF readers.

3.5. Color Types

EISH: Intensity, Saturation, Hue color system.

ELAB: CIE Lightness, a*, b* color system.

ELCH: Lightness, Chroma, Hue color system.

ELSH: Lightness, Saturation, Hue color system.

ELUV: CIE Lightness, u*, v* color system.

ERGB: NTSC/PAL/SMPTE Red, Green, Blue color system.

EVSH: Value, Saturation, Hue color system.

EXYZ: CIE XYZ color system.

EYIQ: CCIR Luma, Inphase, Quadrature color system.

EYSH: CCIR Luma, Saturation, Hue color system.

EYUV: CCIR Luma, U Chroma, V Chroma color system.

PART II

OPEN EVISION STUDIO FOR LIBRARY EVALUATION

1. Selecting your Programming Language

When you start Open eVision Studio for the first time, the following welcome screen is displayed:
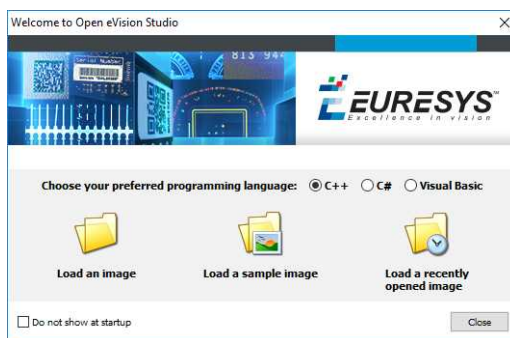
1. Select your programming language.

> ✓ **TIP**
> Your selection is saved and your programming language will be automatically selected next time you start Open eVision Studio.

> 🗒 **NOTE**
> When you change your programming language, any script present in the scripting window is automatically deleted and the window content is reset.

2. Click on one of the Load buttons to already load one or several images for later processing.

3. Check the Do not show at startup box to hide this welcome screen next time you start Open eVision Studio.

> ✓ **TIP**
> To access this welcome screen at any time, and change this setting, go to the Help > Welcome Screen menu.

2. Navigating the Interface
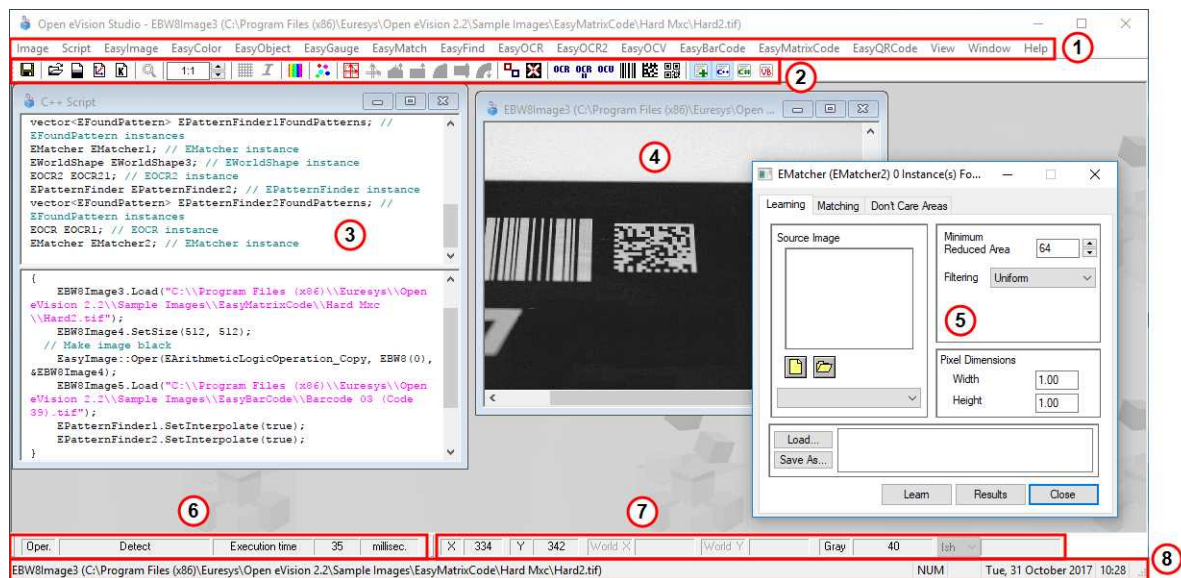
Open eVision Studio graphical user interface (GUI) is organized as follows:

1. The *main menu bar* gives you access to the functions and tools of all libraries.

> ✅ **TIP**
> Open eVision Studio does not require any license and allows you to test all libraries. Of course, if you copy code from Open eVision Studio in your own application but you do not have the required license, you will receive a "missing license" error at run-time.

2. The *main toolbar* gives you a quick access to main Open eVision objects such as images, shapes, gauges, bar codes, matrix codes...

3. The *script window* displays the code, in the programming language you selected, corresponding to the actions you perform in Open eVision Studio. You can save or copy this code in your own application at any time.

4. The *image windows* display the open images that you can process using the libraries and tools.

5. The *tool windows* enable you to easily configure all the available tools. The corresponding settings are automatically added in the script window for easy reuse.

> ✓ **TIP**
> Most tool windows are floating and you can easily move them outside the
> Open eVision Studio main window to make a better use of your screen size.

6. The *execution time bar* displays the precise time taken for the execution of the selected
functions (measured in milliseconds or microseconds) on your computer. This accurate
measurement helps you to evaluate the performance of your application.

7. The *color toolbar* displays current information such as the X and Y coordinates of the cursor
on an image and the corresponding pixel value.

8. The *status bar* displays general information about the application such as the active image file
path...

3. Running Tools on Images

3.1. Step 1: Selecting a Tool

Usually the first step, when using Open eVision Studio, is to select the library and the tool you want to use on your image.

To do so:

1. In the main menu bar, click on the library you want to use.

2. Click on the tool you want to use.

> ✅ **TIP**
> All libraries (except EasyImage, EasyColor and EasyGauge) expose only one tool named New Xxx Tool. Some of these libraries also expose additional functions.

3. In the dialog box, enter a Variable name for the variable that is automatically created and that will contain the result of the processing.
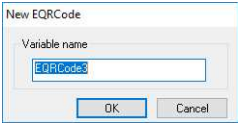
Example of variable creation dialog box for EasyQRCode

4. Click OK.

The selected tool dialog box opens.
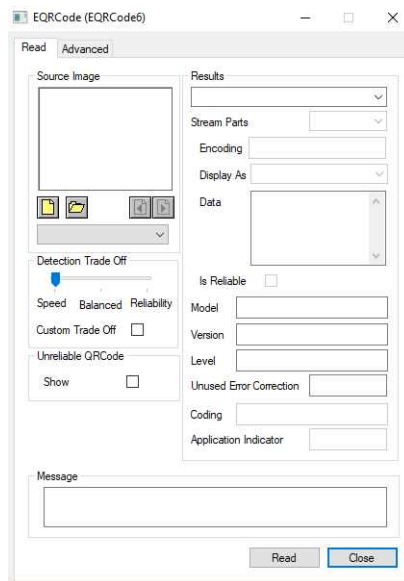
Example of variable creation dialog box for EasyQRCode

The next step is "Step 2: Opening an Image" below.

3.2. Step 2: Opening an Image

Once you have selected your library and your tool, you need to open an image to apply this tool.

In the Source Image area of the selected tool dialog box:

1. Open an image:

☐ Click on the 🗐 Open an Image button and select one or several (using SHIFT and CTRL) images on your computer.

☐ Or select one of the images (or one of the ROIs, if any) already open in the drop-down list.

> **NOTE**
> You can select only images with an appropriate file format (JPG, PNG, TIFF or BMP) and in 8- and/or 24-bit depending on the library.
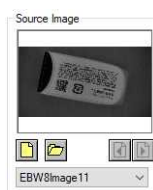
2. If you selected several images, activate one with the 🗐 Load Previous or 🗐 Load Next buttons.

The tool is automatically applied on any loaded image and, at this stage, the result is displayed based on the tool default settings.

The next step is "Step 3: Managing ROIs" below.

3.3. Step 3: Managing ROIs

In some cases, most often to decrease the processing time or to single-out the object you want to read, you do not want to process the whole image but only one or several well defined rectangular parts of this image, or ROIs (Regions Of Interest).

> ✓ **TIP**
>
> In Open eVision, ROIs are attached to an image and exist only as long as the parent image is available.

Creating a ROI

1. Open the image:
   - ☐ If the image is already open, activate the corresponding image window.
   - ☐ If the image is not open yet, go to the main menu: Image > Open... to open one.

2. To create an ROI, go to the main menu: Image > ROI Management....

The ROI Management window is displayed as illustrated below.
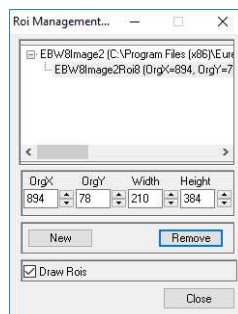
3. Select the image in the tree.

4. Click on the New button.

5. In the dialog box, enter a Variable name for the new ROI.

The ROI is represented as a color rectangle on your image as illustrated below.
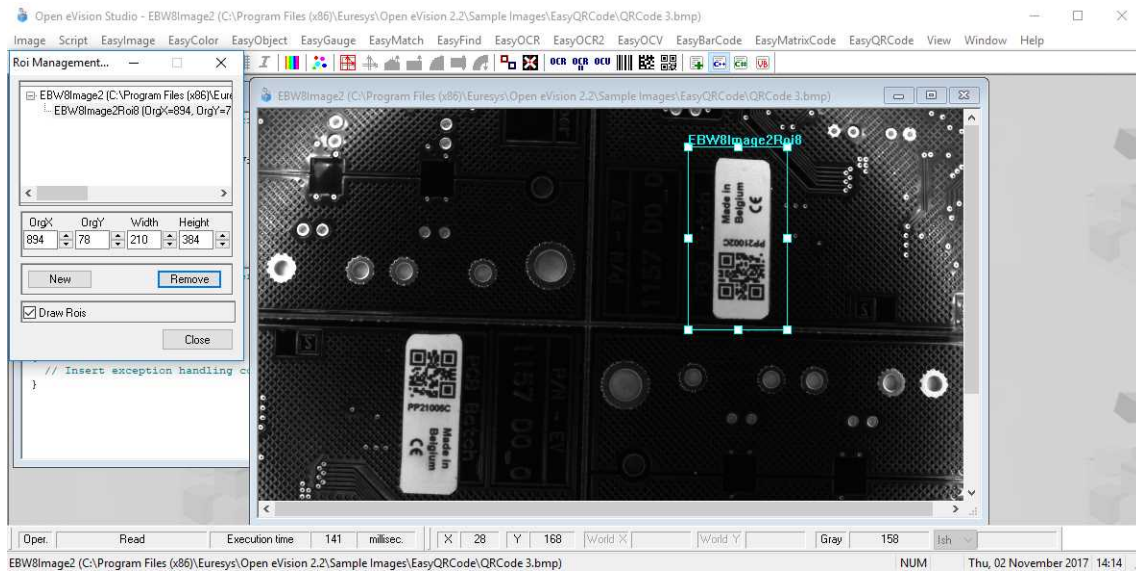
6. Drag the ROI corner and side handles to move it to the required position.

7. Click on the Close button to close the ROI Management window .

The next step is "Step 4: Configuring the Tool" on the next page.

Managing ROIs

You can add, change and remove ROIs.

> ✅ **TIP**
> An image can have several ROIs. Each ROI can be attached directly to the image (meaning that its position is relative to the image) or to another ROI (meaning that its position is relative to this 'parent' ROI).

1. To manage ROIs, go to the main menu: Image > ROI Management....

The ROI Management window is displayed with the ROI relation tree as illustrated below.

If the Draw Rois box is checked, all ROIs are displayed on the image with a different color.
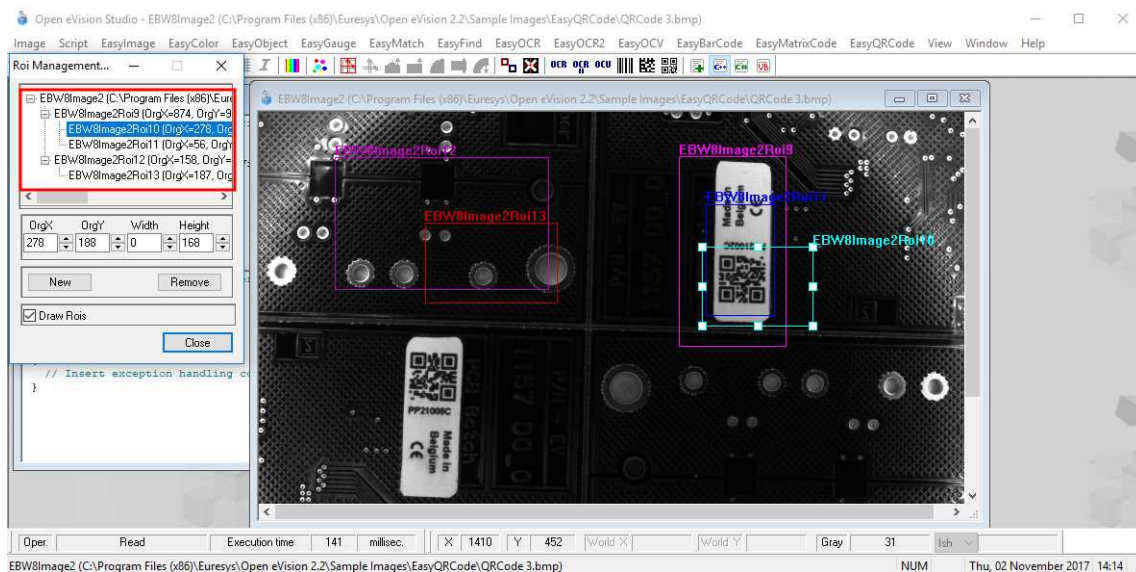
2. Select an ROI in the ROI relation tree.

3. Drag the ROI corner and side handles to change the position and size of the selected ROI (as well as the position of all ROIs attached to it if any).

4. Click on the New button to add a new ROI attached to the selected ROI.

> ✅ **TIP**
> Select the image at the top of the ROI relation tree to attach the ROI directly to the image.

5. Click on the Remove button to delete the selected ROI (and all ROIs attached to it if any).

6. Click on the Close button to close the ROI Management window.

3.4. Step 4: Configuring the Tool

Once your image, including its ROIs if you created some, is ready, you need to configure your tool.

In the tool window:

1. Open the various tabs.

> ✅ **TIP**
> When you create a new tool, all parameters are set with their default value.
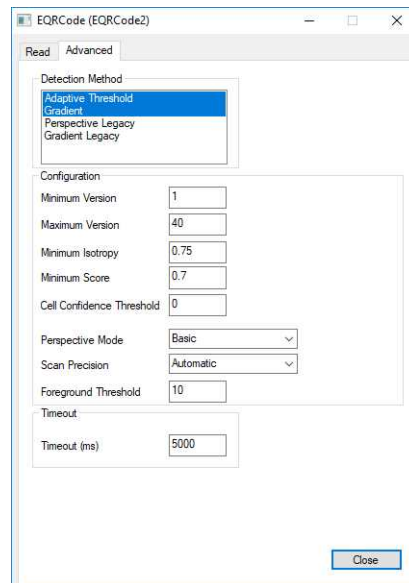
Example of the parameter tab of an EasyQRCode tool

2. In each tab, set the value of the parameters as desired.

Please refer to the "Functional Guide" and to the "Reference Manual" for detailed information about the parameters, their function and their default value.

For specific actions such as learning or using gauges, please refer to the "Functional Guide".

3. Run the tool and analyze the results as described in the next step "Step 5: Running the Tool and Checking Execution Time" below.

3.5. Step 5: Running the Tool and Checking Execution Time

Once your tool parameters are set, run your tool and, if desired, check the execution time on your computer.

In the tool window:

1. Click on the Read, Detect, Results or Execute button (depending on the library function), to run the tool on the selected image.

2. Check the results on the image and in the Results field or area as illustrated below.
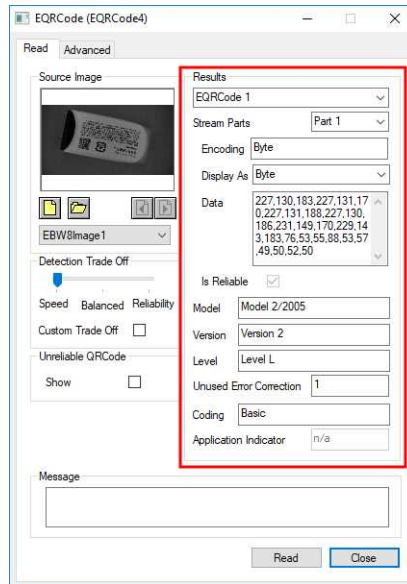
Example of results after reading a QRCode

3. If you do not have the expected results:

  □ Try to change your parameters (start with default values then change one parameter at a time).

  □ If you image is not good enough, try to enhance it as described in .

4. Check the execution time in the execution time bar at the bottom left of the main Open eVision Studio window.
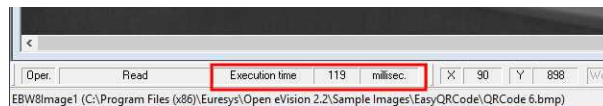
The execution time
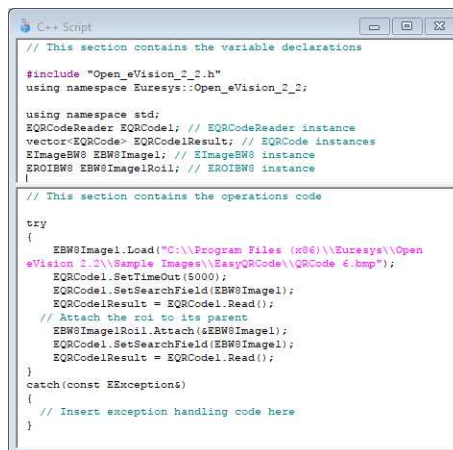
> ✓ **TIP**
> The execution time is the actual time that the processing took as measured on your computer. It depends your computer processor, memory, operating system... and, of course, on the processor load at the time of execution. Thus this execution time slightly varies from execution to execution.

5. To get a more representative execution time, click on the Read, Detect, Results or Execute button several times and calculate the mean execution time.

6. If your application requires that you reduce the execution time, try:

  □ To change the tool parameters,

  □ To add one or several ROIs on your image,

  □ To enhance your image.

The next step is "Step 6: Using the Generated Code" below.

3.6. Step 6: Using the Generated Code

By default, Open eVision Studio translates all the operations you perform in the interface into code in the language you selected as illustrated below.

Once your tool results suit you, you can save or copy this generated code to use it in your own application.

Copy and paste the code in your application

In the script window:

1. Select the code section you want to copy.

2. Right click on this code and click Copy in the menu.

3. Go to you development environment tool and paste the code in place.

Save the code

1. Go to the **Script** menu.

2. Click on Save Script As....

3. Enter a file name and path to save the code as a text file.

Manage the generated code

In the **Script** menu, you can:

☐ Select the programming language (please note that if you change the language, the script window content is automatically deleted).

☐ Activate or deactivate the Script Code Generation. Deactivate this option if you want to perform some operations without saving them as code.

<div style="background:#29ABE2; color:white;">

4. Pre-Processing and Saving Images

</div>

When should you pre-process your images?

Of course, the best situation is to set up your image acquisition system to have good and easy to process images so the Open eVision tools run smoothly and efficiently.

If this is not possible or easy to achieve, you can pre-process your images or your ROIs to enhance and prepare them for the Open eVision tool you want to run.

Using the various available functions, you can adjust the gain and offset of your image, apply a convolution, threshold, scale, rotate and white balance your image, enhance contours... using EasyImage and EasyColor functions.

Pre-processing images

The difference between pre-processing an image and running tools is that the pre-processing generates a new image while the tools mainly extract and retrieve information from the image without changing it.

To pre-process an image or an ROI:

1. In the main menu bar, click on the library you want to use (EasyImage or EasyColor).

2. Click on the function you want to use.

Most function dialog boxes are similar to the one illustrated below with 2 image selection areas and a parameter setting area.
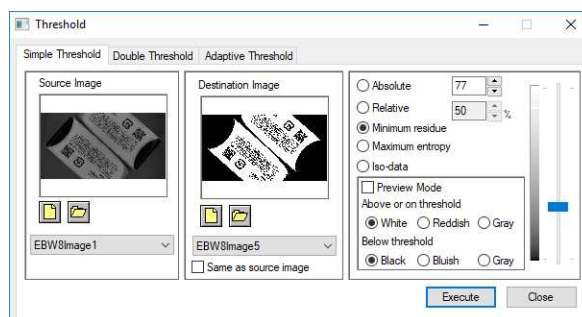
Example of a pre-processing dialog box (Threshold with EasyImage)

3. If there are multiple versions for your selected function, open the corresponding tab.

4. In the Source Image area, open the source image (as described in "Step 2: Opening an Image" on page 25).

5. In the Destination Image area, open or create a new destination image.

6. Set your parameters.

7. Click on the Execute button.

The pre-processed image is available in the destination image as illustrated below.
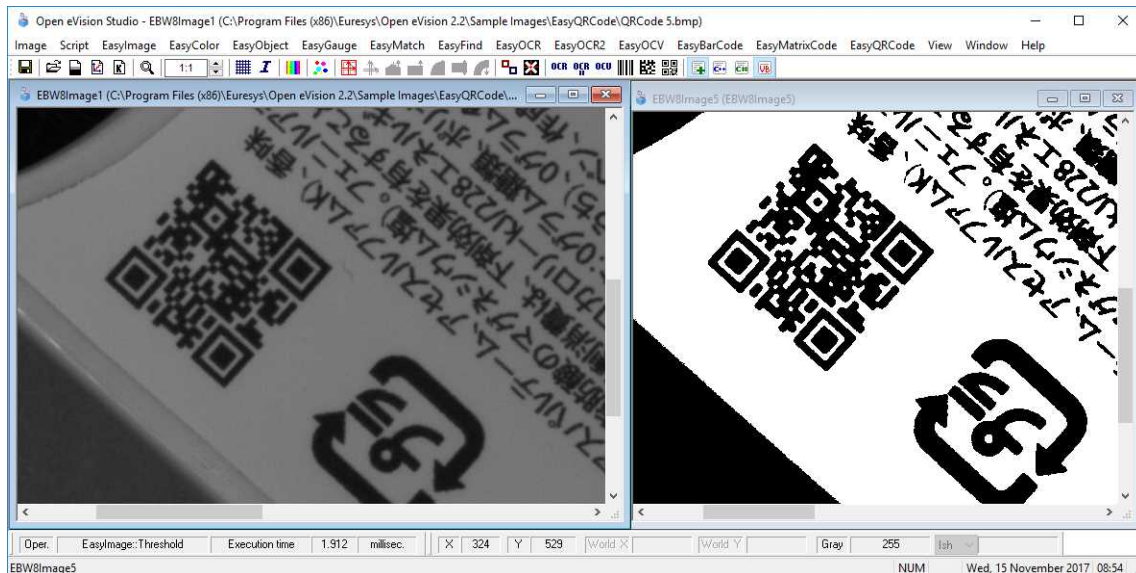
Source and destinations images (Threshold with EasyImage)

8. If you want to use the destination image outside of Open eVision Studio, save it as described below.

Saving an image

1. Click in the image you want to save to activate it.

2. To open the save menu either:

- ☐ Right-click in the image

- ☐ Or open the main menu > Image

3. Click on Save as....

4. Select the file format (JPEG, JPEG2000, PNG, TIFF or Bitmap).

5. Enter a name and select a path.

6. Click on the Save button.

PART III

EASY 3D STUDIO FOR 3D EVALUATION

1. Start the Application and Open a Project

1. Launch Easy3D Studio.

The following screen is displayed:
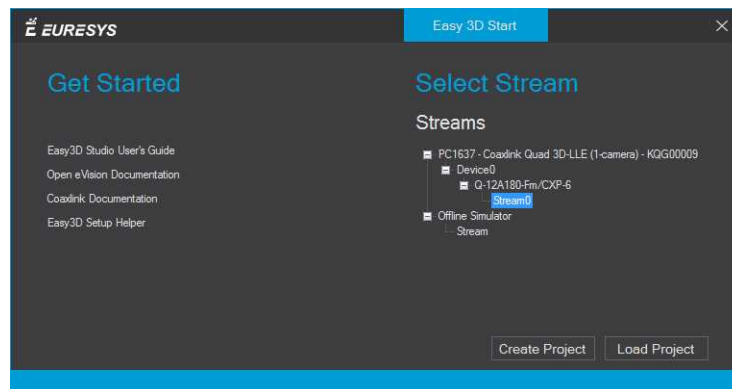
2. Select your stream in the list of the available streams displayed in the Select Stream area.

3. Open your project (`.json` file):

- ☐ Click on Create Project to create a new project.

- ☐ Click on Load Project to open an existing project.

The main screen is displayed as illustrated:
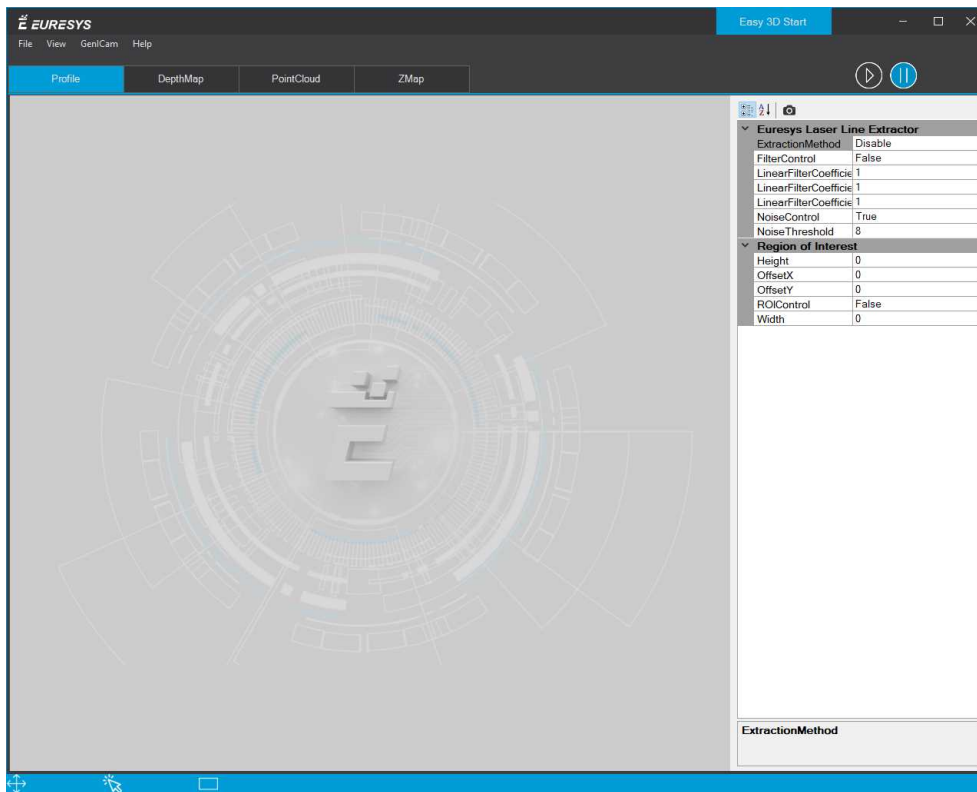
2. Configure the Acquisition ROI

1. Open the Profile tab.

2. Click on the ▷ run button to start free-running acquisition.

3. Place your caliber or your object under the laser line.

> ✓ **TIP**
> To optimize the computation time, make sure the laser line illuminates the lowest as well as the highest areas of the object.
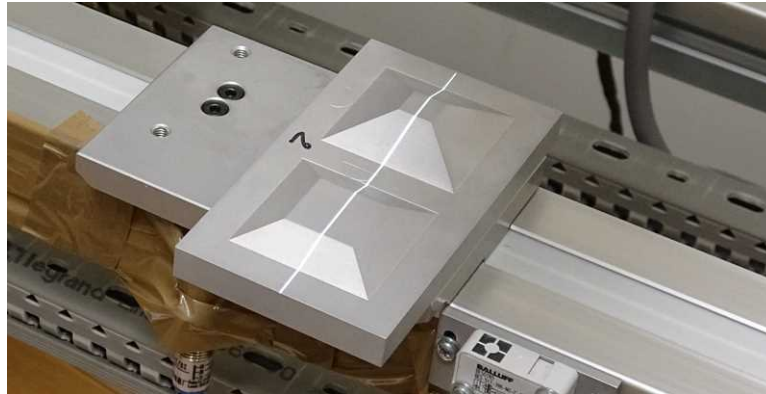
4. In the **Profile** window, move and resize the ROI (the green rectangle) to fit and enclose the laser line image.

> ✅ **TIP**
>
> With a smaller the ROI, the extraction computation is faster and the resolution is better.
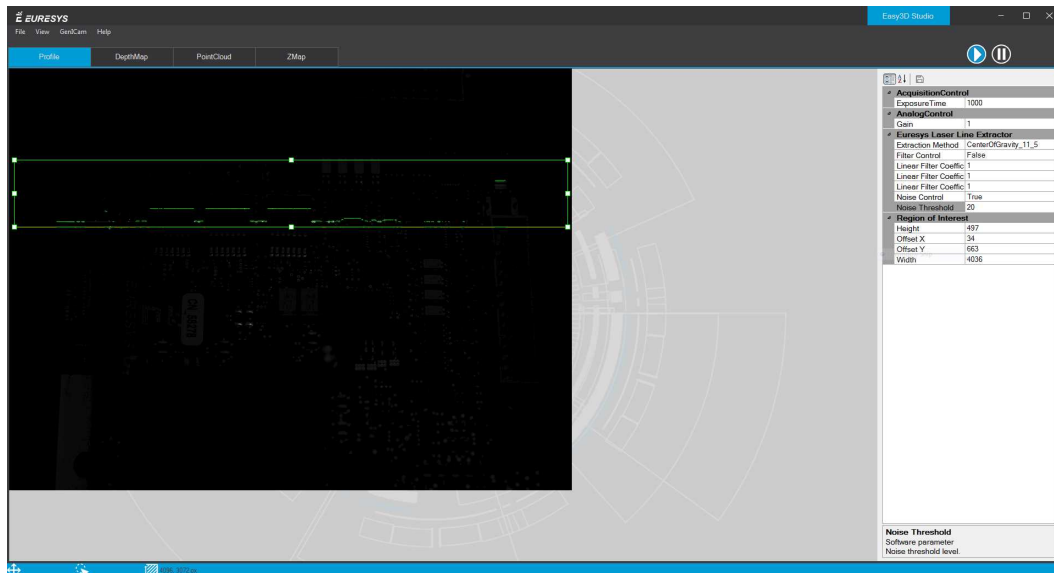
5. In the parameter list, select the `ExtractionMethod` to use to compute the laser line extraction.

> ✅ **TIP**
>
> Do not use `MaxDetection_8` if your ROI height is larger than 256 as this would introduce a saturation effect in the extraction result.

6. Once the setup is complete, click on the ⏸ pause button to stop the acquisition.

3. Calibrate the Setup

1. Open the DepthMap tab.

2. Place your caliber on your system.

3. Click on the ▷ run button to start an acquisition.

4. Move the caliber under the camera and wait for the image acquisition to complete.

5. Once the acquisition is displayed, click on the ‖ pause button to stop the acquisition.
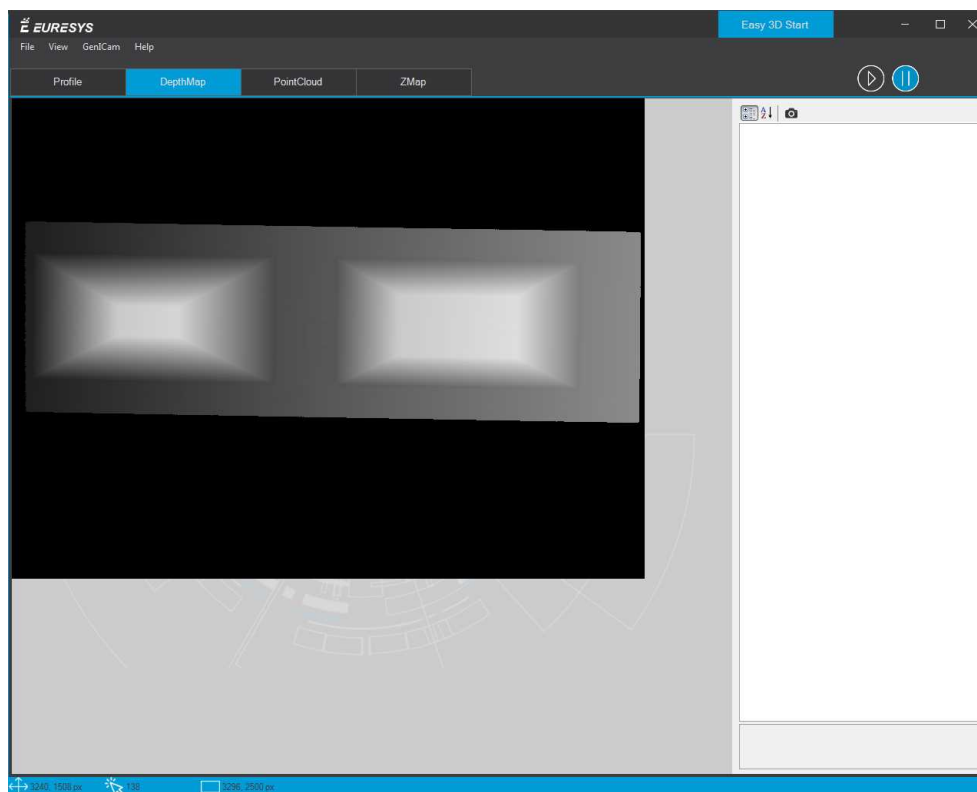
6. Open the PointCloud tab.
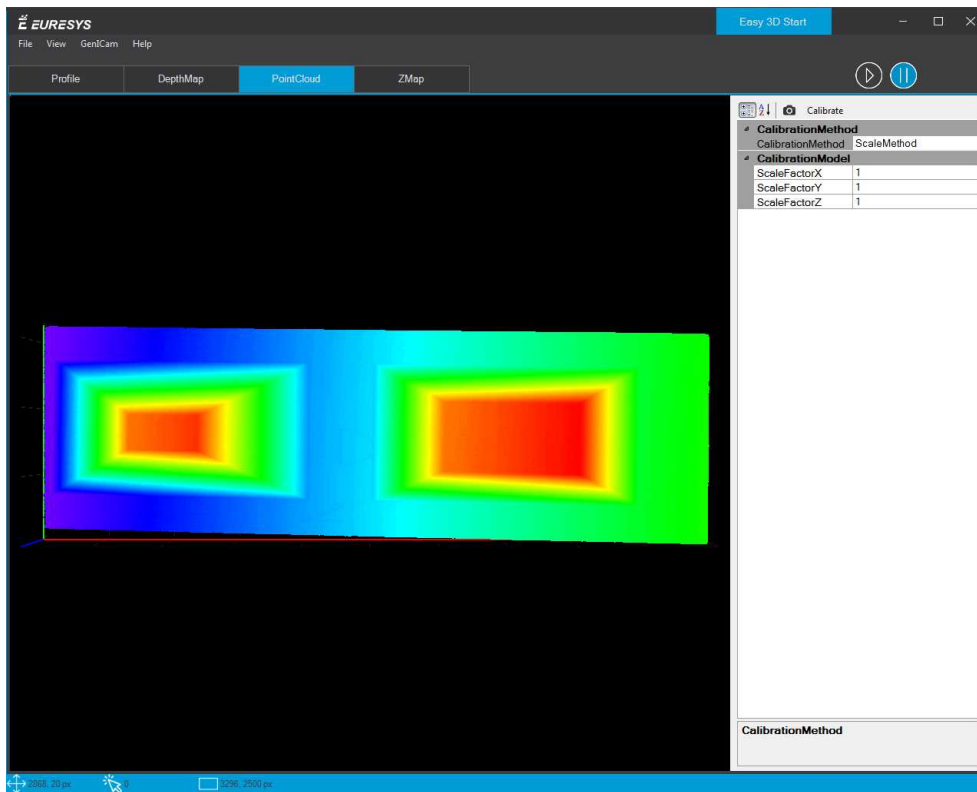
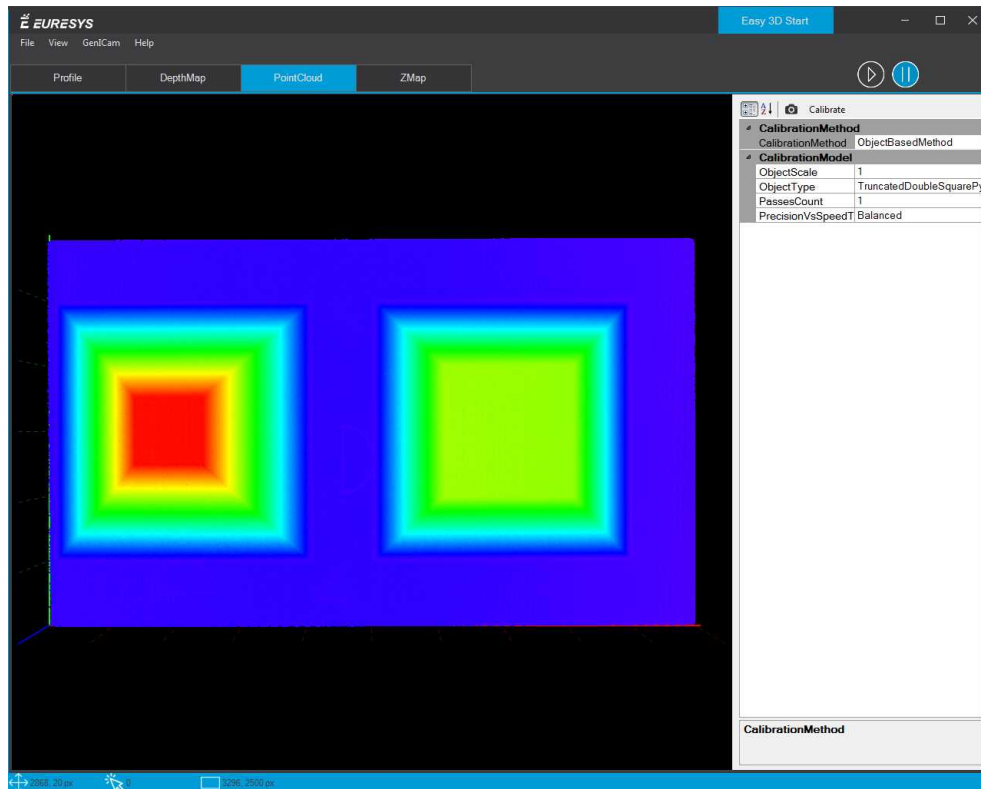**7.** Use the mouse on the image to freely move it around in 3D.
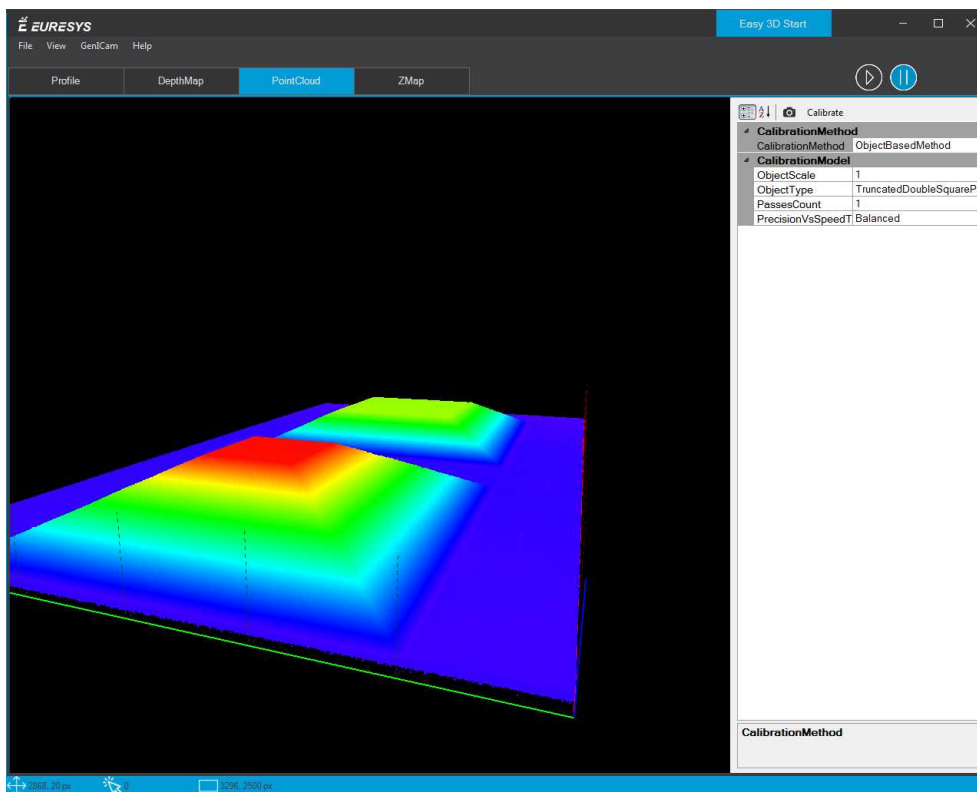
8. In the parameter list:

   □ Set `CalibrationMethod` to `ObjectBasedMethod`.

   □ Select the `Objecttype` corresponding to your caliber.

9. Click on the Calibrate button.

After some processing, the calibrated image point cloud is displayed.

4. Acquire a Depth Map

1. Open the DepthMap tab.

2. Place your object on your system.

3. Click on the ▷ run button to start an acquisition.

4. Move the object under the camera and wait for the image acquisition to complete.

5. Once the acquisition is displayed, click on the ❚❚ pause button to stop the acquisition.
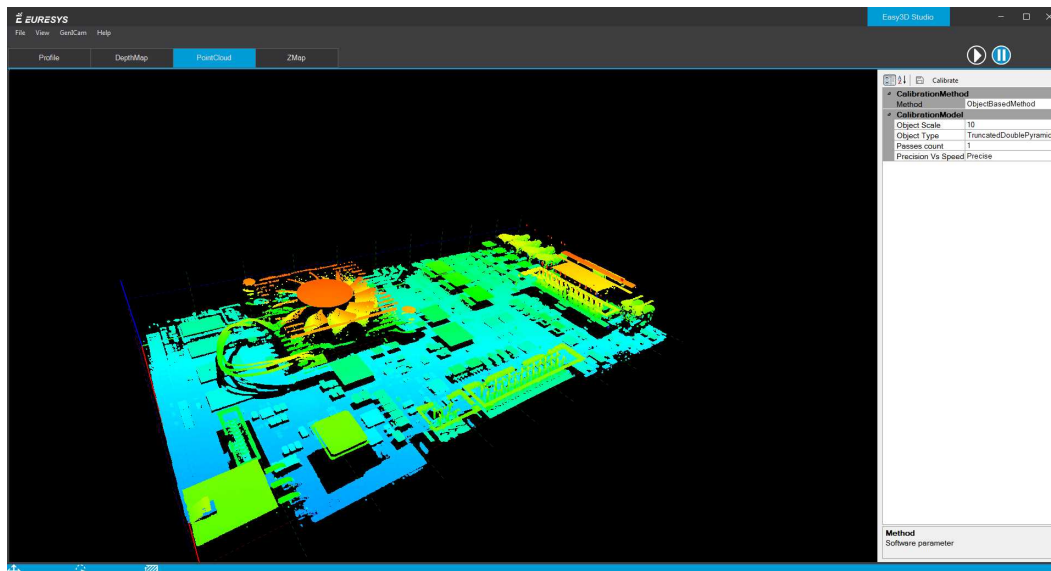
The "raw" image of your object is displayed.
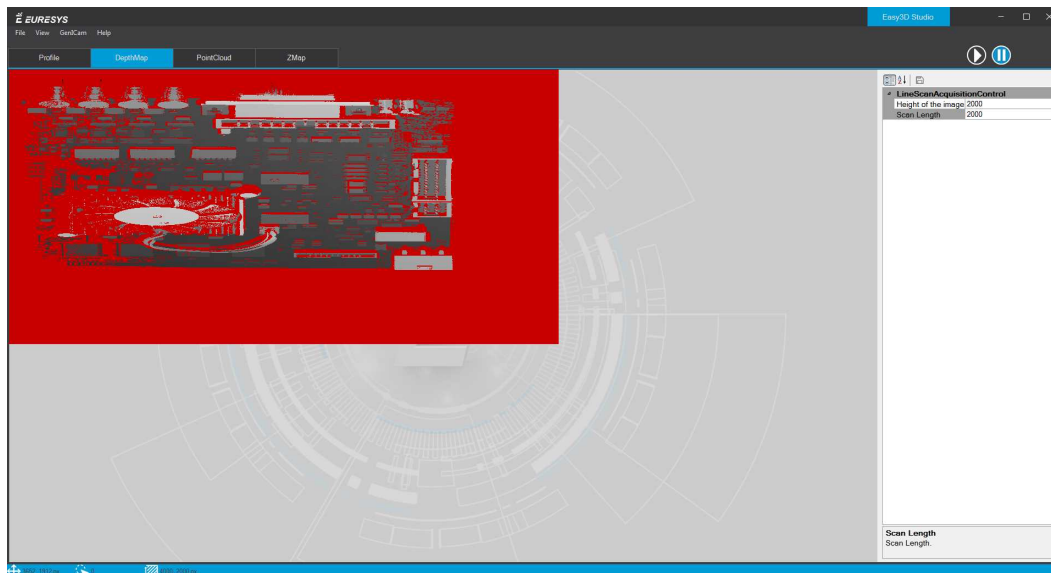
6. Open the PointCloud tab.

The calibration correction is applied to the image before display.
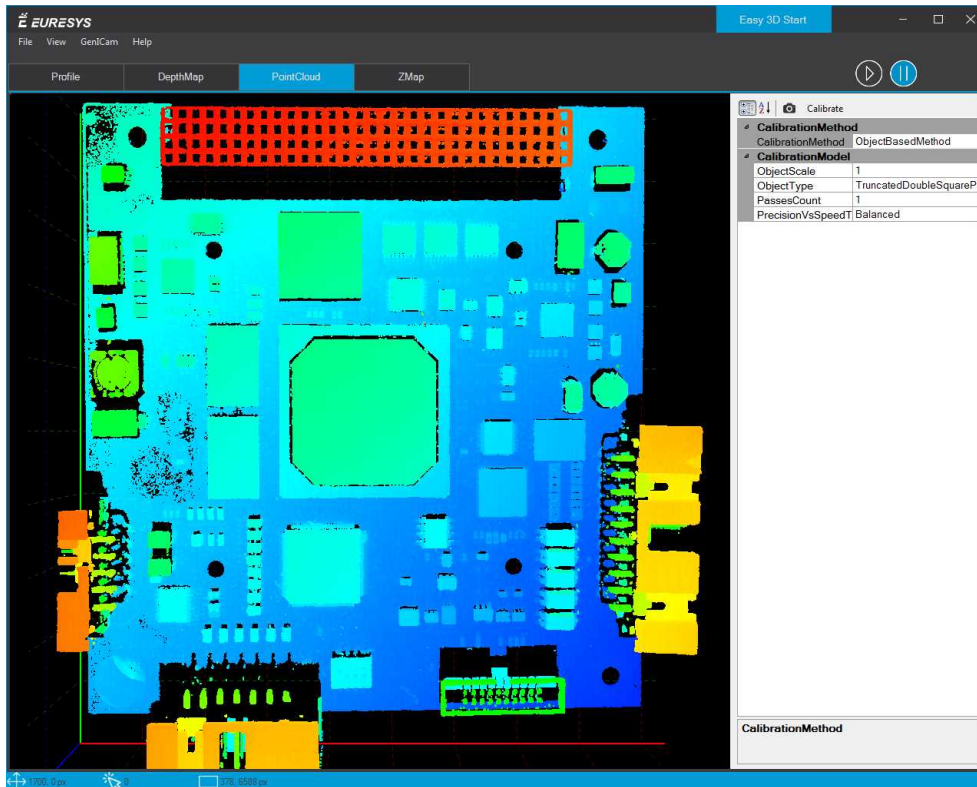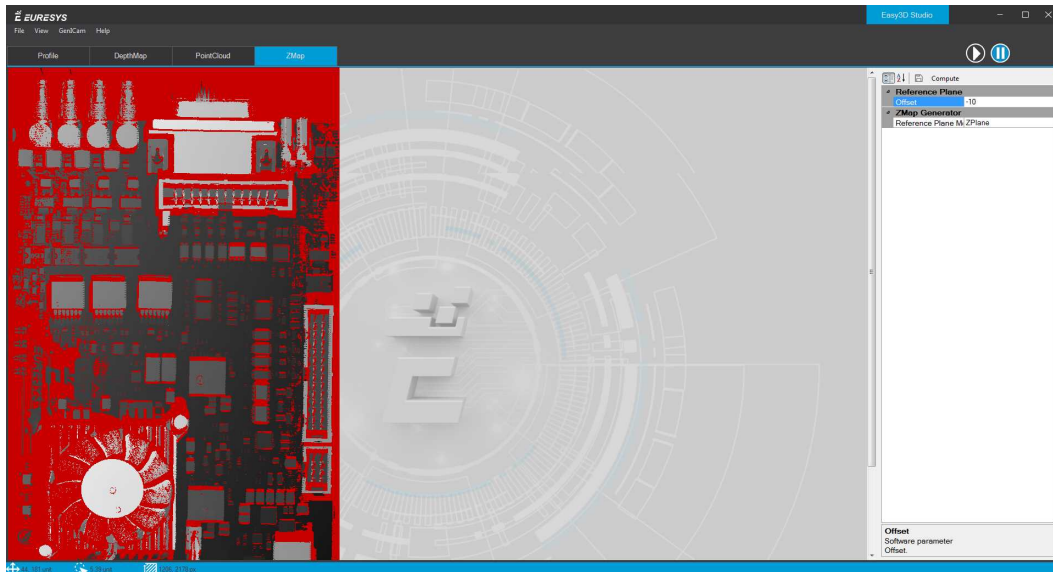

7. Use the mouse on the image to freely move it around in 3D.

8. Open the ZMap tab.

The corresponding ZMap is generated and displayed.

9. Configure the ZMap view:

 □ Select the projection axis of the image: set the `ReferencePlaneMode` as X, Y or Z-plane.

 □ Along the selected axis, set the offset of the displayed image.

10. Click on the Compute button to apply your changes and display the new image.

5. Export a ZMap

1. Open the File menu.

2. Click on Export ZMap.

3. Select the type of file you want to create:

   □ PNG

   □ TIFF

   □ BMP (Bitmap)

4. Enter the file name.

5. Click on the Save button.

You can also save a profile, a depth map or a point cloud the same way.

The available file formats are:

   □ Profile: PNG, TIFF, Bitmap

   □ Depth map: PNG, TIFF, Bitmap

   □ Point cloud: PCD

6. Add or Remove Parameters

The parameters listed in the different tabs are the basic software parameters that you need to control and adjust to use Easy3D Studio.

In the Profile and in the DepthMap tabs, you can add some hardware parameters (related to the frame grabber or to the camera) that you want to quickly and easily display or control:

1. Open the GenICam menu.

2. Click on GenICam.

The GenICam window opens as illustrated.
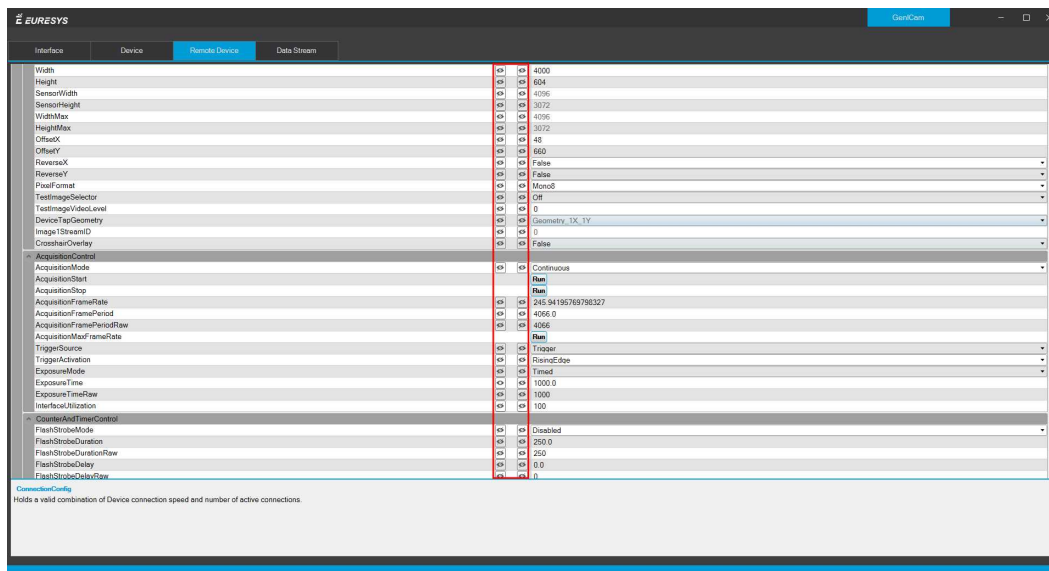
3. To show or hide a parameter, click on the 'eye' icon:

☐ Left 👁 : the parameter is hidden in the Profile tab.

☐ Right 👁: the parameter is hidden in the DepthMap tab.

☐ Left 👁: the parameter is displayed in the Profile tab.

☐ Right 👁: the parameter is displayed in the Profile tab.

> ✅ **TIP**
> The changes are automatically and immediately applied in the Easy3D Studio main window.

4. Close the GenICam window .

5. To save the your changes, open the File menu and click Save Project.