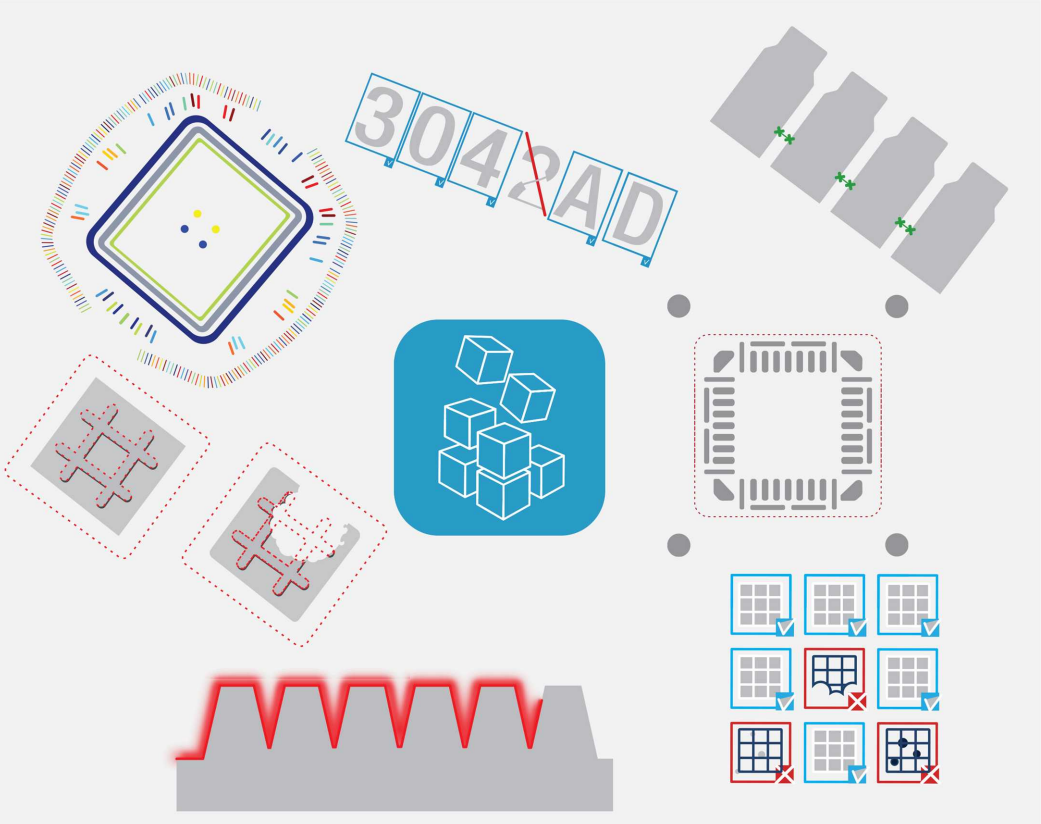
euresys
Empowering Computer Vision

RELEASE NOTES

Open eVision

Release 26.6.1



© EURESYS S.A. 2026 - Doc. D101EN-Release Notes-Open eVision-26.6.1.1239 built on 2026-07-31

This documentation is provided with **Open eVision 26.6.1** (doc build **1239**).
www.euresys.com

This documentation is subject to the General Terms and Conditions stated on the website of **EURESYS S.A.** and available on the webpage https://www.euresys.com/en/Menu-Legal/Terms-conditions. The article 10 (Limitations of Liability and Disclaimers) and article 12 (Intellectual Property Rights) are more specifically applicable.

Contents

1. Release Benefits

What's new in **Open eVision 26.6**

EasyDeepOCR

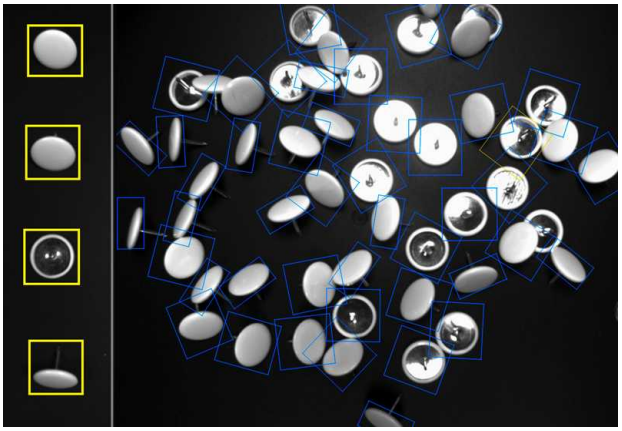
User trained models

- You can retrain the neural network behind the text detection, orientation and recognition.
  - A dedicated project type in **Deep Learning Studio** provides an efficient interface for annotating the texts and running the training.
  - With as few as 5 images, you can achieve a noticeable improvement on difficult use cases.

- Training the models
  - You can train the models in **Deep Learning Studio** with a dedicated project type.
  - Use the new class `EDeepOCRTool` in the namespace `EasyDeepLearning` to train 3 types of model: location, orientation and recognition. This class is based on `EDeepLearningTool` as the other **Deep Learning** tools `EClassifier`, `ESupervisedSegmenter` or `ELocator`.
  - The dataset class (`EClassificationDataset`) is extended to support the OCR annotations.
- Inference
  - After training a model, you can export it then import it in `ETextReader` with one of the methods `ETextReader.LocationCustomModel`, `ETextReader.OrientationCustomModel` or `ETextReader.RecognitionCustomModel` according to the type of model that was trained.
- **Open eVision Studio**
  - The tool `Text reader` supports custom model configurations for location, orientation and recognition.

EasyFind

- Use the new method `EPatternFinder.LearnPatterns` to learn several instances of a pattern to be more robust to changes in illumination, perspective or deformations.
  - The new method `NumLearnedPatterns` is also available.
  - The existing methods `CopyLearntPattern`, `GetFeaturePoints` and `DrawModel` are extended to support multiple patterns.



EasyMatch

- Use the new method `EPatternFinder.LearnPatterns` to learn several instances of a pattern to be more robust to changes in illumination, perspective or deformations.
  - The new method `NumLearnedPatterns` is also available.
  - The existing method `CopyLearntPattern` is extended to support multiple patterns.

Open eVision Studio

With this release **Open eVision 26.6**, **Open eVision Studio** is leaving its "preview" state.

This means that:

- Starting now, **Open eVision Studio** will be fully backward compatible.
- The future versions will be able to open projects build with older versions (down to 26.6).
- **Open eVision Studio 26.6** is not compatible with previously saved projects.

New tools:

- **Easy**
  - `Draw 2D` to draw several objects together on an image.
  - `From array` to extract the i^{th} element of an array.
  - `Grid` .
- **EasyColor**
  - `Bayer conversion` to transform an image captured using the Bayer pattern and stored as a gray-level image, into a true color image.

- **EasyImage**
  - `Equalization` to equalize an image histogram (the gray levels are re-mapped).
  - `Fourier operation` to perform the direct Fourier transform using the fast Fourier transform (FFT).
  - `Gabor filter` to apply a Gabor wavelet convolution to the input image (BW8 and BW16 only).
  - `Histogram computation` to compute the histogram of an image.
  - `Image focus` to compute the average gradient energy (focusing metric) of an image.
  - `Image statistics` to compute pixel statistics on images using an optional region.
  - `Sliding Window` to compute the average and standard deviation of gray-level values in a sliding window.
- **EasyObject**
  - `Checker` to check if regions of an image are within expected tolerances.
  - `Object template matcher` to match an `EObjectSelection` or a vector of `EPoint` to a corresponding template.
  - `Region to object` to convert an `ERegion` to an `EObjectSelection`.
- **EasyFind**
  - `Vector model` to load or save a `vectorModel` and specify the polarity of its components.
- **EasyBarCode**
  - `MailBarcode reader` to read mail bar codes in an image.
- **Easy3D**
  - `Draw 3D` to draw several objects together in an `E3DViewer`.
- **Easy3DMatch**
  - `3D Point Cloud merger` to merge point clouds together after a calibration.
- **Easy3DObject**
  - `3D Object extractor` to extract 3D objects from a ZMap.

Deprecated items

See the new section "Deprecation Table" on page 37 for a complete list and history of the deprecated items.

EasySpotDetector

- `ESpotDetector` is now deprecated and will be removed in a future release.

Legacy Open eVision Studio

- The **Legacy Open eVision Studio** is superseded and no longer released with **Open eVision**.
  - ► Instead, use the **Open eVision Studio**.

2. Release Specifications

OS and processor architectures

Windows OS

- **Open eVision** is a 64-bit library that requires a processor compatible with the SSE4 instruction set.
- **Open eVision** runs on the following Windows operating systems:

OS version	Additional information	
Windows 11®	64-bit	
Windows 10®	64-bit	
Windows 11® IoT Enterprise	64-bit	on x86_64 systems
Windows 10® IoT Enterprise	64-bit	

Linux OS

- **Open eVision** is compatible with x86_64 and aarch64 (ARMv8-A) CPUs.
- **Open eVision** and the **Neo License Manager** are designed to be distribution-independent on x86_64 platforms.
- The minimum requirements are:
  - gcc version 10
  - glibc version 2.27
- This release has been validated with the following distributions and their default gcc compilers and cmake programs:
  - For **deb** based distributions, the following distributions are tested and supported:
    - **Ubuntu 22.04** and **Ubuntu 24.04**
    - **Debian 11** and **Debian 12**
  - For **rpm** based distributions, the following distributions are tested and supported:
    - **RHEL 10**
- This release has been validated on the following embedded systems:
  - **Raspberry** Pi 4, **Raspberry** Pi 5 and **Raspberry** Pi Zero 2 W
  - **NVidia** Jetson Orin series

- For **CMake** samples: the client programs must be linked against the pthread library.
- For **Qt** samples: it is not necessary to make this dependency explicit because a program using Qt automatically depends on the pthread library.

NVIDIA GPU support

- **Open eVision** uses **CUDA** v12.9 and supports only GPU with a compute capability of 7.5 or more: from architecture Turing (RTX2000 series) to Blackwell (RTX 5000 series).

Remote connections and virtual machines

- Remote connections
  - You can install and use **Open eVision** licenses on a remote connection using remote desktop, **TeamViewer** or any other similar software.
- Virtual machines
  - Virtual machines are supported. **Microsoft Hyper-V**, **Oracle VirtualBox** and **libvirt** hypervisors have been successfully tested.
  - Only the **Neo Licensing System** is compatible with virtualization.

Supported programming languages and IDEs

The **Open eVision** libraries and tools support C++, Python and the programming languages compatible with the .NET (C#, VB.NET).

C++ requirements

- A compiler compatible with the C++ 11 standard is required to use **Open eVision**.
- Some **Open eVision** samples use the **Qt** framework to create a user interface and display images with a graphical overlay. We recommend the **Qt** versions 5.12 to 5.15.

.NET requirements

- The .NET framework 4.8 (or later) or the .NET platform 6.0 (or later) is required to use the C# version of **Open eVision**.

Python requirements

- Python 3.11 or later is required to use the Python bindings for **Open eVision**.
- No other third party dependencies are necessary to use the Python bindings.

Compatibility with IDEs

- Select the recommended API according to your IDE and programming language:

	Programming language	
IDE	**C++**	**C#, VB.NET, C++/CLI**
Microsoft Visual Studio 2017®	C++	.NET Assembly
Microsoft Visual Studio 2019®	C++	.NET Assembly
Microsoft Visual Studio 2022®	C++	.NET Assembly
QtCreator 4.15 with Qt 5.12 (*)	C++	

(*) A C++ compiler like **MSVC** must be installed.

- Any **Windows** and **Linux** IDE supporting a compatible compiler (**MSVC 2017** or higher, **GCC 7.5** or higher) should be suitable to build **Open eVision** applications.

3. Release Details

3.1. Features Updates

New features

All

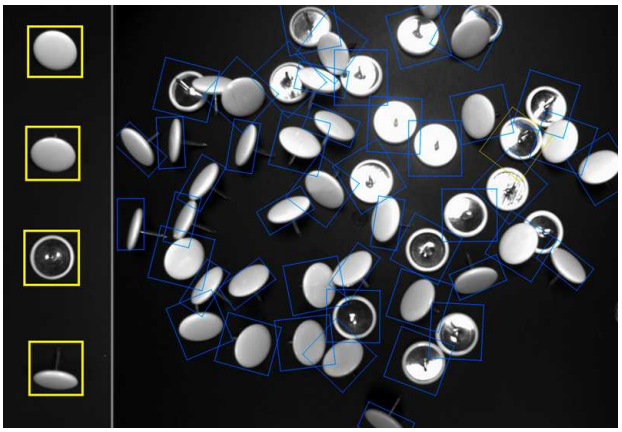
- Use the new command line helper `oev_helper`, added to the binaries folder on all platforms, to provide a direct access to a limited number of **Open eVision** features through the command line for tooling purposes.
  - Currently it only features the sub-command `health-check` to call `Easy.HealthCheck` through the command line.
  - It may be extended in the future to provide more features.
- The version numbering convention of **Open eVision** is changed to be **semver-compatible**.
  - In practice, the only change is that minor versions drop the additional “0” when they are below 10.

Easy

- Use `EGrid`, a new rectangular grid, to divide the input image into smaller regions where you can read codes.
- Use the new method `HandleSize` to set and change the handle size that is used in drawing methods and in hit-test methods.
- Use the new parameter `CustomMinDotArea` of `ECameraCalibration` to tweak the minimum area that a dot in a dot grid should have before being filtered out.
  - This is especially useful for previously failing images. These failing images were large ones with little dots. The internal heuristic was filtering the correct dots, leaving no remaining dots to perform the actual calibration.
- Use the new method `Easy.HealthCheck` to perform an automated check on the **Open eVision** installation.

EasyFind

- Use the new timeouts `EPatternFinder.LearnTimeout` and `EPatternFinder.FindTimeout` to stop a call to `EPatternFinder.Learn` and `EPatternFinder.LearnPatterns` or `EPatternFinder.Find` respectively.
  - This is useful when tuning the settings, for example in **Open eVision Studio**, as some settings can increase the processing time prohibitively.
  - The timeouts are also added in **Open eVision Studio**.
- Use the new option `EPatternFinder.ComputePerFeaturePointScore` to compute a score for each feature point.
  - The score is rendered by using a color map in `EFoundPattern.Draw`. In **Open eVision Studio**, the sample program `Find a pattern` is updated to illustrate this new feature.
- Use the new method `EPatternFinder.LearnPatterns` to learn several instances of a pattern to be more robust to changes in illumination, perspective or deformations.
  - The new method `NumLearnedPatterns` is also available.
  - The existing methods `CopyLearntPattern`, `GetFeaturePoints` and `DrawModel` are extended to support multiple patterns.



EasyMatch

- Use the new timeouts `EMatcher.LearnTimeout` and `EMatcher.MatchTimeout` to stop a call to `EMatcher.Learn` and `EMatcher.LearnPatterns` or `EMatcher.Match` respectively.
  - This is useful when tuning the settings, for example in **Open eVision Studio**, as some settings can increase the processing time prohibitively.
  - The timeouts are also added in **Open eVision Studio**.
- Use the new method `EPatternFinder.LearnPatterns` to learn several instances of a pattern to be more robust to changes in illumination, perspective or deformations.
  - The new method `NumLearnedPatterns` is also available.
  - The existing method `CopyLearntPattern` is extended to support multiple patterns.

EasyObject

- You can now create an EObjectSelection from an ERegion using either a dedicated constructor or the method AddRegionAsObjects.
- An EObjectSelection has now a copy constructor and an assignment operator.
- EObjectTemplateMatcher now supports frames.
- Use the new methods GetMatched, GetExtra and GetMissing of EObjectTemplateMatcher to retrieve the results.
- EObjectTemplateMatcher has new alignment modes: Local and Robust.
- EObjectTemplateMatcher has a new property: MaximumAngle.
- Use the new class EChecker instead of the legacy class EChecker2. It performs the same way except that the alignment is handled externally and the alignment information is given to the class EChecker using EFrames. This allows more flexibility in the alignment during the process.
- Use the new method TrainOnSingleImage of the class EChecker to train on one image by using tolerance thresholds.

EasyBarCode, EasyMatrixCode and EasyQRCode

- The methods Read of EMatrixCodeReader, EBarCodeReader, EQRCodeReader and ECodeReader now take an optional EGrid parameter. The old overloads are deprecated.
- Use the new method DrawDecodedString of EasyBarCode2.EBarCode, EasyMatrixCode2.EMatrixCode, EQRCode and EMailBarCode to draw the decoded string below the position of the code in an image.

EasyBarCode

- Use the new methods EnablePatternCorrection, EnableAdvancedDecoding and EnablePrintingErrorCorrection of the class EBarCodeReader to set the error correction more precisely than what was possible with EnablePermissiveDecoding.
- Use the new methods GetPatternCorrectionUsed, GetAdvancedDecodingUsed and GetPrintingErrorCorrectionUsed of the class EBarCode to query what error correction was used for the given bar code and symbology.

Easy3D

- ERenderStyle has a new constructor.
- Use the new methods E3DViewer.Add3DObjects and E3DViewer.Num3DObjects to render 3D objects from several vectors in the E3DViewer.

EasyDeepOCR

User trained models

- You can retrain the neural network behind the text detection, orientation and recognition.
  - A dedicated project type in **Deep Learning Studio** provides an efficient interface for annotating the texts and running the training.
  - With as few as 5 images, you can achieve a noticeable improvement on difficult use cases.

- Training the models
  - You can train the models in **Deep Learning Studio** with a dedicated project type.
  - Use the new class `EDeepOCRTool` in the namespace `EasyDeepLearning` to train 3 types of model: location, orientation and recognition. This class is based on `EDeepLearningTool` as the other **Deep Learning** tools `EClassifier`, `ESupervisedSegmenter` or `ELocator`.
  - The dataset class (`EClassificationDataset`) is extended to support the OCR annotations.
- Inference
  - After training a model, you can export it then import it in `ETextReader` with one of the methods `ETextReader.LocationCustomModel`, `ETextReader.OrientationCustomModel` or `ETextReader.RecognitionCustomModel` according to the type of model that was trained.
- **Open eVision Studio**
  - The tool `Text reader` supports custom model configurations for location, orientation and recognition.

Open eVision Studio

With this release **Open eVision 26.6**, **Open eVision Studio** is leaving its "preview" state.

This means that:

- Starting now, **Open eVision Studio** will be fully backward compatible.
- The future versions will be able to open projects build with older versions (down to 26.6).
- **Open eVision Studio 26.6** is not compatible with previously saved projects.

New tools:

- **Easy**
  - `Draw 2D` to draw several objects together on an image.
  - `From array` to extract the i^{th} element of an array.
  - `Grid` .

- **EasyColor**
  - `Bayer conversion` to transform an image captured using the Bayer pattern and stored as a gray-level image, into a true color image.
- **EasyImage**
  - `Equalization` to equalize an image histogram (the gray levels are re-mapped).
  - `Fourier operation` to perform the direct Fourier transform using the fast Fourier transform (FFT).
  - `Gabor filter` to apply a Gabor wavelet convolution to the input image (BW8 and BW16 only).
  - `Histogram computation` to compute the histogram of an image.
  - `Image focus` to compute the average gradient energy (focusing metric) of an image.
  - `Image statistics` to compute pixel statistics on images using an optional region.
  - `Sliding Window` to compute the average and standard deviation of gray-level values in a sliding window.
- **EasyObject**
  - `Checker` to check if regions of an image are within expected tolerances.
  - `Object template matcher` to match an `EObjectSelection` or a vector of `EPoint` to a corresponding template.
  - `Region to object` to convert an `ERegion` to an `EObjectSelection`.
- **EasyFind**
  - `Vector model` to load or save a `vectorModel` and specify the polarity of its components.
- **EasyBarCode**
  - `MailBarcode reader` to read mail bar codes in an image.
- **Easy3D**
  - `Draw 3D` to draw several objects together in an `E3DViewer`.
- **Easy3DMatch**
  - `3D Point Cloud merger` to merge point clouds together after a calibration.
- **Easy3DObject**
  - `3D Object extractor` to extract 3D objects from a ZMap.

New sample projects:

- `MailBarcode reading` reads mail bar codes. This sample relies on the **EasyBarCode**.
- `Bayer conversion` uses **EasyColor** to transform an image captured using the Bayer pattern and stored as a gray-level image, into a true color image, applying Bayer conversion.
- `Find a vector pattern` uses **EasyFind** to find a pattern in an image. The pattern was learned from an `EVectorModel`, which was loaded from a `dxf` file.
- `Find two patterns and draw results` uses **EasyFind** to find 2 patterns, the top and bottom faces of the same item, in an image. The 2 types of found pattern are then displayed in the same tool.
- `Learn multiple patterns` uses **EasyFind** to learn 4 patterns corresponding to different perspectives of the same object and find them in an image.

- `Histogram computation` uses the histogram computation to compare the histogram of a gray-level image.
- `Equalization` equalizes several images to make their contents more visible.
- `3D objects extractor` detects defects in a pharmaceutical blister by extracting only the blisters whose volume is higher than a specified threshold.
- `Merging point clouds` uses **Easy3DMatch** to merge point clouds.
- `Checker without alignment` checks defects on printed characters that are already aligned with `TrainOnSingleImage` on a single image and tolerance thresholds.
- `Checker with basic alignment` checks defects on chips after alignment with a pattern finder.
- `Checker with advanced alignment` checks defects on PCBs after alignment of 2 fiducials combined to form a frame with 2 pattern finders.
- `Matrix code grid` divides the input image into a rectangular grid and reads multiple matrix codes cell by cell.
- `Learn a pattern with a region` uses **EasyFind** to find a pattern in an image. The pattern is learned with a region of interest to filter out bad feature points.
- `Align a pattern` uses **EasyFind** to find a pattern in an image. The image is then transformed to make the pattern horizontal and cropped around it.
- `Matching objects with a template` uses `EObjectTemplateMatcher` to check if all the balls are present on a BGA.
- `Bearing inspection` uses **EasyMatch** to perform an alignment of the bearing, in position and orientation. Then, 4 circle gauges are used to check the position and the shape of 4 rolling elements.

These tools are improved:

- **Easy**
  - `To array` accepts `E3DTransformMatrix` and `EPointCloud` as inputs.
  - `To frame` now supports an EPoint as input.
  - `Line` can now be set from two `EPoint` inputs.
- **EasyImage**
  - `Canny edge detector` now supports `ERegion` and `EFrame`.
  - `Threshold` now supports `ERegion` and `EFrame`.
- **EasyMatch**
  - `Matcher` now supports a vector of match images as input.
- **EasyFind**
  - `Pattern finder` now supports a vector of find images as input.
- **EasyBarCode**, **EasyMatrixCode** and **EasyQRCode**
  - `BarCode reader`, `MatrixCode reader` and `QRCode reader` now support grid as input.

The interface is improved:

- The tools now display a processing indicator in the graph in the form of a pulsing light green border. Consequently, the status `Processing...` has been removed.
- There is a link to the online help home page of **Open eVision Studio**.
- You can now duplicate a tool either alone or with its child tools.
- The **Open eVision Studio** executable has a new sub-command `health-check` to perform an automated check of the installation of the program for tooling purposes.
- The message log display window now has a button `Clear`.

Improvements

All

- The following terms are no longer are reserved keywords: `EasyWorld`, `EUnit_um`, `EUnit_mm`, `EUnit_cm`, `EUnit_dm`, `EUnit_m`, `EUnit_dam`, `EUnit_hm`, `EUnit_km`, `EUnit_mil`, `EUnit_inch`, `EUnit_foot`, `EUnit_yard`, `EUnit_mile`.

Easy

- Drawing the feature points in `EFoundPattern.Draw` has been improved when the zoom factor is > 1.
- On vectors, the methods `GetElement` are now `const` (see `VectorTypes.h`).
- Use the new method `EFoundPattern.LearntPatternOffset` to get the offset of the learned pattern corresponding to the `EFoundPattern`.
- The classes `ERectangle` and `EQuadrangle` have a new method `ComputeOverlap`.
- `EFrame.DisplaySize` now uses the absolute values of the scales when computing the display size.
  - Previously, it could return a negative value when both the `ScaleX` and `ScaleY` were negative (which was displayed as 0 in the **Open eVision Studio** tool).

EasyImage

- The following methods now support overloads with `ERegion` and `EFrame` to apply these operators on a limited area:
  - `ECannyEdgeDetector.Apply`
  - `EasyImage.LocalAverage` and `EasyImage.LocalDeviation`
  - `EasyImage.Focusing`
- The following methods now work faster when using an `ERegion`:
  - `EasyImage.Histogram`
  - In the classes `EImageStatisticsBW8` and `EImageStatisticsBW16`: `GravityCenterOfPixelsAboveHighThreshold`, `GetBinaryMoments` and `GetWeightedMoments`
  - Most methods of `EasyImage` when the area of the `ERegion` is smaller than the area of the image.
- `EBWHistogramVector.Draw` now manages the `EPlotDecorationOptions`.
- `EasyImage.SetupEqualize` now manages a `BW16` histogram.

EasyObject

- You can now draw directly an `ECodedElement` and a `EObjectSelection` with their new methods `Draw` instead of using an object `ECodedImage2`.
- The largest run is no longer enabled by default.
- The result list now displays the total number of objects.

EasyGauge

- The usage of `SetMinNumFitSamples` is improved with a better documentation and the addition of more expressive overloads. This parameter is now also available **Open eVision Studio**.

EasyFind and EasyMatch

- Using a learn/find frame with a scale < 0 and an angle ≠ 0 is now allowed (previously, it threw an exception).

EasyMatrixCode

- The read capabilities are improved in some cases.

EasyDeepOCR

- The maximum ratio that a miniature can have for the recognition is increased from 1:20 to 1:40. This means that you can now read wider texts.
- When a miniature had a ratio bigger than 1:20, the reader threw an "`internal error 15`". This exception is renamed to `EError_EDOReaderError_Recognize` with a more helpful message.

Open eVision Studio

- Interface
  - The clarity of the message displayed when hovering the mouse over depth maps or ZMaps has been improved.
  - In the `Tool catalog`, the tool sorting has been improved.
  - In the `Sample catalog`, the sample sorting has been improved.
  - The result tables now highlight the selected row with the standard application style.
  - The depth map settings for the tools `3D DepthMap file`, the `3D Image to DepthMap converter` and `StudioLink` (if available) are unified.
  - The dialog boxes for the project `Load` and `Save` now open in the documents folder by default, and remember the last project folder independently of the tool file pickers.
  - The splash screen now appears a little faster at start-up.

- Acquisition
  - The tool `StudioLink` (if available) is updated and the related tools are extended to handle its additional functionalities.
  - The tool `Image file` now displays the current image index over the total number of loaded images in the image preview.
  - In the tools `3D DepthMap file`, `3D Mesh file`, `3D Point Cloud file`, `3D ZMap file` and `Image file`, the error logs now contain the path of problematic files that could not be loaded.
- **Easy**
  - The tool `To array` now supports ROIs and up to 6 inputs.
  - In the tool `Region editor`: The combined regions, the morphological operations (grow, shrink, contour) and the affine transforms (translate, scale, rotate) are now combined in a tree structure in which you can edit each node. Use the new edition panel and the dedicated toolbar buttons to edit these nodes.
  - *(26.6.1)* In the tool `Region editor`: The property spin boxes now support up to 7 decimal places and a wider value range. The default increments are also changed.
  - *(26.6.1)* In the tool `Frame`: The property spin boxes now support up to 7 decimal places. The default increments are also changed.
- Code readers
  - In all code reader tools, the setting `Use human readable code for GS1` is now taken into account when generating code.
- **EasyBarCode**
  - In the tool `BarCode reader`, the settings `Include checksum` and `Symbologies` are now taken into account when generating code.
- **Easy3D**
  - The tool `3D Principal axis extractor` now displays the axes in both the setup widget and the generated code.
- **EasyDeepOCR**
  - In the tool `Text reader`, you can now configure the settings in the section `Draw` (`ETextDrawingParameters`).

Region icon:

- The 2D display toolbar shows the region icon when an `ERegion` is connected as an input of the tool.
- The following tools are updated to make the `ERegion` drawing optional:
  - `Canny edge detector`
  - `Convolution`
  - `BarCode reader`
  - `Draw 2D`
  - `Gabor filter`
  - `Image statistics`
  - `Image to region`
  - `MatrixCode reader`
  - `Morphology`
  - `QRCode reader`
  - `Region to image`
  - `Transform`
  - `Threshold`
  - `To frame`

3.2. Breaking Changes

Starting with this release **26.6**, **Open eVision** implements the following changes:

Easy

- *(26.6.0 - Breaking change)* `ETopology` and `ETopologyLine` are moved from the namespace `Euresys::Open_eVision::EasyDeepOCR` to the namespace `Euresys::Open_eVision` and only requires a `BasicTypes` license.
  - Therefore, you can use it to, for example, filter codes read with `EBarCodeReader`, `EMatrixCodeReader`...
- *(26.6.0 - Breaking change)* `Easy.GetErrorText` is removed.
- *(26.6.0 - Breaking change)* The way of computing the `ERuns` of the `EPolygonRegion` has been slightly changed so that they are equal to the `ERuns` computed with a `ERectangleRegion`.
  - This may provoke small changes in the results throughout other **Open eVision** libraries that involve such regions.
- *(26.6.0 - Breaking change)* The format of `Easy.Version` is modified from `"\d+.\d+.\d+.\d+"` to semver syntax.

EasyFind

- *(26.6.0 - Breaking change)* `EPatternFinder.PointShape`, as well as the daughters argument of `EPatternFinder.Save` and `EPatternFinder.Load` are removed.
- *(26.6.0 - Breaking change)* The results of **EasyFind** were not completely consistent between Linux x64 and Windows. The behavior on all platforms is slightly altered to fix this.

EasyGauge

- *(26.6.0 - Breaking change)* When using multiple filtering passes, all the gauges have now a different behavior. The setting of the parameter `MinNumFitSamples` was not applied after the first pass if it meant that some samples that were previously valid were now outliers. This has been fixed.

EasyBarCode

- *(26.6.0 - Breaking change)* The new property `EnablePrintingErrorCorrection` of `EBarCodeReader` is now disabled by default. In previous version, it was enabled and you could use `EnablePermissiveDecoding` to change it.

EasyOCV

- *(26.6.0 - Breaking change)* The class `EChecker` was moved to the namespace `Legacy`.

EasyDeepOCR

- *(26.6.0 - Breaking change)* `ETopology` and `ETopologyLine` are moved from the namespace `Euresys::Open_eVision::EasyDeepOCR` to the namespace `Euresys::Open_eVision`. For more details, see the section about **Easy** above.

Deprecated items

Easy

- *(26.6.0 - Breaking change)* The classes `EImageC15`, `EImageC16`, `EImageBW1` and all the related methods, enums and the segmenter are now removed. These image types were declared deprecated and flagged for removal in **Open eVision** 25.10.

EasyObject

- *(26.6.0 - Breaking change)* The `EChecker2` class is deprecated and moved to the Legacy namespace.

3.3. Changes

Starting with this release **26.6**, **Open eVision** implements the following changes:

All

- *(26.6.0)* The types `EStockMeasurementUnit`, `EMeasurementUnit` and `DimensionalValue` are removed.

Easy

- *(26.6.0)* In the C++ API, `EVectorModel::GetScale`, `EVectorModel::GetCenter` and `EVectorModel::Draw` are now `const` methods.
- *(26.6.0 - Breaking change)* `ETopology` and `ETopologyLine` are moved from the namespace `Euresys::Open_eVision::EasyDeepOCR` to the namespace `Euresys::Open_eVision` and only requires a `BasicTypes` license.
  - Therefore, you can use it to, for example, filter codes read with `EBarCodeReader`, `EMatrixCodeReader`...
- *(26.6.0 - Breaking change)* `Easy.GetErrorText` is removed.
- *(26.6.0 - Breaking change)* The way of computing the `ERuns` of the `EPolygonRegion` has been slightly changed so that they are equal to the `ERuns` computed with a `ERectangleRegion`.
  - This may provoke small changes in the results throughout other **Open eVision** libraries that involve such regions.
- *(26.6.0)* `ERectangleRegion.IsPointInRegion` is modified to test if the point belongs to the rasterized region.
- *(26.6.0 - Breaking change)* The format of `Easy.Version` is modified from "`\d+.\d+.\d+.\d+`" to semver syntax.

EasyImage

- *(26.6.0)* The method `EasyImage.Overlay` and all its overloads now have default values for the parameters `panX`, `panY` and `referenceValue`.

EasyFind

- *(26.6.0 - Breaking change)* `EPatternFinder.PointShape`, as well as the daughters argument of `EPatternFinder.Save` and `EPatternFinder.Load` are removed.
- *(26.6.0 - Breaking change)* The results of **EasyFind** were not completely consistent between Linux x64 and Windows. The behavior on all platforms is slightly altered to fix this.

EasyGauge

- *(26.6.0 - Breaking change)* When using multiple filtering passes, all the gauges have now a different behavior. The setting of the parameter `MinNumFitSamples` was not applied after the first pass if it meant that some samples that were previously valid were now outliers. This has been fixed.
- *(26.6.0)* An `ECameraCalibration` whose calibration mode is `ECalibrationMode_Advanced` now throws an exception when trying to get non-pertinent parameters. This is the case for `TiltXAngle`, `TiltYAngle`, `PerspectiveStrength` and `DistortionStrength`. On the other hand, the advanced calibration parameters are now exposed when the calibration mode is `ECalibrationMode_Advanced` (and throws an exception if it is not the case).

EasyBarCode

- *(26.6.0 - Breaking change)* The new property `EnablePrintingErrorCorrection` of `EBarCodeReader` is now disabled by default. In previous version, it was enabled and you could use `EnablePermissiveDecoding` to change it.

EasyOCV

- *(26.6.0 - Breaking change)* The class `EChecker` was moved to the namespace `Legacy`.

Easy3D

- *(26.6.0)* The drawn `E3DObject` are now taken into account when computing the view of a scene with `E3DViewer.ResetView`.

EasyDeepOCR

- *(26.6.0 - Breaking change)* `ETopology` and `ETopologyLine` are moved from the namespace `Euresys::Open_eVision::EasyDeepOCR` to the namespace `Euresys::Open_eVision`. For more details, see the section about **Easy** above.

Linux installers

- *(26.6.0)* `libgl1` is added to the recommended dependencies of the x86_64 and arm64 Debian packages.

Deep Learning tools

- *(26.6.0)* When `EDeepLearningTool.OptimizeBatchSize` is `true`, the optimized batch size is only computed when calling `EDeepLearningTool.InitializeInference` or the method `Train` of a tool.
  - Before, the batch size was directly optimized for inference when loading a trained model. This set `EDeepLearningTool.OptimizeBatchSize` to `true`, set `EDeepLearningTool.EnableGPU` to `true` or changed `EDeepLearningTool.GPUIndexes`.

EasyLocate

- *(26.6.0)* The sample is changed so the tool name is now the same as the project name. And if the tool is not found, the training is now done for 50 epoch .

Deep Learning Studio

- *(26.6.0)* **Deep Learning Studio** is upgraded to use Qt 6.8 instead of Qt 5.15.

Demo applications

- *(26.6.0)* The demo samples `Fuse Inspection` and `Clinical Analysis` are updated to use the new **EasyGauge** API introduced in release 25.10.
  - The drawing and the results of the sample `Fuse Inspection` are slightly modified.

Open eVision Studio

- *(26.6.0)* The **new Open eVision Studio (preview)** is now named **Open eVision Studio**.
- *(26.6.0)* Tool `Dot grid calibration`: The non-pertinent parameters displayed in the results tabs of the calibration tool are now hidden (see the relevant change above in the section **EasyGauge**).
- *(26.6.0)* Tool `Image Stitcher`: Enabling the reference image and the grid mode with images `EImageBW8` as input triggered a crash. This has been fixed.
- *(26.6.0)* The tools code readers, find, match, object selection, image encoder, text reader and `EasyLocate` are now in failing state (the tool is colored in red) when no code / pattern / object / text / element is found and the output port is not connected.
- *(26.6.0)* Tools `MatrixCode reader` and `QRCode reader`: The setting `Show human readable code for GS1` is now enabled by default.

Deprecated items

See the new section "Deprecation Table" on page 37 for a complete list and history of the deprecated items.

Easy

- *(26.6.0 - Breaking change)* The classes `EImageC15`, `EImageC16`, `EImageBW1` and all the related methods, enums and the segmenter are now removed. These image types were declared deprecated and flagged for removal in **Open eVision** 25.10.

EasySpotDetector

- *(26.6.0)* `ESpotDetector` is now deprecated and will be removed in a future release.

Code readers

- *(26.6.0)* The grid reading methods of `EMatrixCodeReader`, `EBarCodeReader`, `EQRCodeReader` and `ECodeReader` are deprecated.
  - ▶ Instead, use the new overloads taking a parameter `EGrid`.
- *(26.6.0)* The property `EnablePermissiveDecoding` of `EBarCodeReader` is deprecated.

EasyObject

- *(26.6.0 - Breaking change)* The `EChecker2` class is deprecated and moved to the Legacy namespace.
- *(26.6.0)* `EObjectTemplateMatcher.EnableAlignment` is deprecated in favor of setting an `enum`.
  - ▶ Instead, use `EObjectTemplateMatcher.Alignment`.
- *(26.6.0)* `EObjectTemplateMatcher.SortPositions` is deprecated.
  - ▶ Instead, use `EObjectTemplateMatcher.Match`.

EasyDeepOCR

- *(26.6.0)* `ETextReader.Localize` is deprecated.
  - ▶ Instead, use `ETextReader.Locate`.

Legacy Open eVision Studio

(26.6.0) The **Legacy Open eVision Studio** is superseded and no longer released with **Open eVision**.

- ▶ Instead, use the **Open eVision Studio**.

3.4. Solved Issues

The following issues have been fixed in **Open eVision 26.6**:

Easy

- *(26.6.0)* `EVectorModel.Scale` and `EVectorModel.Center` did not work properly when called after a modification of the polarity of one of the shapes of the `EVectorModel`. This has been fixed.
- *(26.6.0)* `EShape.HitTest` did not work as intended when not providing a value for the parameter `zoomY`. Instead of using the same value as `zoomX`, its value was equal to 1. This has been fixed.

EasyImage

- *(26.6.0)* When using `EasyImage.SetupEqualize` with large images, overflows sometimes happened. This has been fixed.
- *(26.6.0)* When using `EasyImage.Focusing` with BW16 images, overflows sometimes happened leading to wrong and inconsistent results. This has been fixed.
- *(26.6.0)* When using `EasyImage.MathFrames` with BW16 images, overflows sometimes happened leading to wrong results. This has been fixed.
- *(26.6.0)* When using `EasyImage.AnalyseHistogram` and `EasyImage.AnalyseHistogramBW16` especially with large and/or BW16 images, integer overflows sometimes happened. This has been fixed.
- *(26.6.0)* The BW16 version of `EasyImage.Lut` supposed that all 16 bits were significant. It has been extended from 9- to 16-significant bits.
- *(26.6.0)* In `EImageStitcher`, the calculation method to compute the transformation failed to correctly filter erroneous 90° rotations during the grid stitching. This has been fixed.
- *(26.6.0)* Calling `EasyImage.ConvolUniform` with a default constructed `EFrame` ignored the half kernel width and height parameters. This has been fixed.
- *(26.6.0)* The method `EasyImage.Overlay` and all its overlays threw invalid exceptions or crashed and their documentation was unclear. This has been fixed.
- *(26.6.0)* `EasyImage.Histogram` did not compute a saturated histogram when the input was `BW16` and `mostSignificantBit` was not 15. This has been fixed.
- *(26.6.0)* `EasyImage.AnalyseHistogramBW16` could return a negative value for the features `EHistogramFeature_MostFrequentPixelFrequency` and `EHistogramFeature_LeastFrequentPixelFrequency` when the true value was ≥ 2^31. This has been fixed.

EasyObject

- *(26.6.0)* The assignment and the copy of an `EObjectTemplateMatcher` did not work properly after a using `BuildTemplate`. This has been fixed.
- *(26.6.1)* Converting an object selection `EObjectSelection` to an `ERegion` produced wrong results when the content of the object selection was computed using an `EFrame`. This has been fixed.

EasyFind

- *(26.6.0)* The feature points drawn by `EFoundPattern.Draw` were sometimes one pixel off. This has been fixed.
- *(26.6.0)* When a vector model was learned, the sequence “assignment / copy construction, serialization, deserialization” resulted in a `EPatternFinder` not completely equivalent to the original. This has been fixed.

EasyMatch

- *(26.6.0)* Setting an overlap with `EMatcher.MaxOverlap` could cause the matcher to find less matches than specified by `EMatcher.MaxPositions`. This has been fixed.
  - ► Note that there is no guarantee to get all the requested matches.

Code readers

- *(26.6.0)* Reading with an `ERegion` outside of the image could throw an exception instead of returning an empty vector (including in the grid mode). This has been fixed.

EasyBarCode

- *(26.6.0)* For Pharmacodes, returning the decoded string without a checksum yielded an empty string. This has been fixed.

EasyMatrixCode

- *(26.6.0)* The timeouts did not work properly in the grid mode. This has been fixed.
- *(26.6.0)* When a `learn` was performed on a grid followed by a `read`, an exception concerning a ROI of a different size was thrown even if the image was a valid input. This has been fixed.

Deep Learning Studio

- *(26.6.0)* In the **EasyLocate** projects, the detection threshold was not cleared when removing the last tool of a project. This has been fixed.
- *(26.6.0)* When the training iterations are very short, clicking on the check boxes to show or hide the various metric curves in the tab `Tool` could be ineffective. This has been fixed.
- *(26.6.0)* When the automatic iterations was set to `true` when training a tool, and the tool finishes training, the validation and the results were computed using the training device or engine and not the inference device or engine. This has been fixed.
- *(26.6.0)* When importing images with the same file name, removing the first image and loading the second one could display a stale thumbnail while the image viewer showed the correct image. This has been fixed.
- *(26.6.1)* In the tools `Locator Deep OCR`, `EasySegment Supervised` and `EasySegment Unsupervised`, in the tab `Validation & Results`, the plot widget would automatically reset its bounds when modifying the threshold. This has been fixed.
- *(26.6.1)* **Deep Learning Studio** crashed when displaying a tool `EasySegment Unsupervised` in the tab `Tool` if it was trained only with good images. This has been fixed.

Easy3D

- *(26.6.0)* When `NumCalibrationPasses` was > 1, the `EObjectBasedCalibrationGenerator` could return a cryptic error "`List is Empty`" in some cases. This has been fixed and it now returns a proper error.
- *(26.6.0)* `EConverter.Convert` introduced an incorrect shift when converting an `EZMap32f` to an `EZMap8` or an `EZMap16`. This has been fixed.

EasyDeepOCR

- *(26.6.0)* `ETextReader.TextAngle` did not use the `Easy.AngleUnit` and always used `degrees` as units. This has been fixed.
- *(26.6.1)* When training a Location model with scale data augmentation, the training results were not good. This has been fixed.

Open eVision Studio

- General
  - *(26.6.0)* Saving a ZMap or a depth map from the 2D display widget produced "`.json`" metadata files while loading a ZMap or a depth map expected "`.mtd`" files. The saving now produces "`.mtd`" files as well.
  - *(26.6.0)* The tools performing batch processing on arrays sometimes produced an invalid code. This has been fixed.
  - *(26.6.0)* After clearing a project, the CPU consumption could be high. This has been fixed.
- Interface
  - *(26.6.0)* In the `Tool catalog`, the tool tip color changed depending on the selection state of the underlying tool. This has been fixed.
  - *(26.6.0)* In the `Tool catalog`, a display problem could introduce some 'holes' in the catalog. This has been fixed.
- **Acquisition**
  - *(26.6.0)* In the tool `eGrabber studio`, when the output was a depth map, the display widgets were interpreting the data as images. This has been fixed.
  - *(26.6.0)* In the tool `eGrabber studio`, the parameter `ZResolution` updated the generated code but not the output depth map. This has been fixed.
- **Multiple tools**
  - *(26.6.1)* In several tools (including `Checker`, `Matcher` and `Pattern finder`), an error could occur when saving a project that contains tools whose serialization produced large files (depending on their input and configuration in the project). This has been fixed.
  - *(26.6.1)* In several tools, selecting a result with the mouse or the keyboard in the results table did not reliably update the highlighted item in the display view. This has been fixed at least for the tools `Pattern finder`, `Matcher`, `Spot detector`, `EasyLocate`, `Point gauge`, `Text reader`, `EasySegment supervised` and `3D Object Extractor` and all the tools in **EasyObject**.

- **Easy**
  - *(26.6.0)* The tool `Circle` generated code using the radius instead of the diameter, so the circle was half its expected size. This has been fixed.
  - *(26.6.0)* The tool `Region to image` incorrectly used `SensorToLocal` instead of `LocalToSensor` when a frame was provided. This has been fixed.
  - *(26.6.0)* In the shapes and gauges tools, when `View parent` was selected in the display widget and the display widget was showing an ROI, the shape was drawn with an incorrect pan. This has been fixed.
  - *(26.6.0)* In the tool `Frame` when an isotropic input frame was connected, the checkbox `Rotatable` could not be unchecked. This has been fixed.
  - *(26.6.1)* In the tool `Region editor`, unwanted scrolling was allowed in the node, the setup and the additional views of the tool. This has been fixed.
  - *(26.6.1)* In the tool `Region editor`, the currently selected region was highlighted in the node and the additional views. This was misleading regarding what the output of the tool really looked like. This has been fixed.
  - *(26.6.1)* In the tool `Region editor`, the combination operations (union, intersection and subtraction) that produce an empty region are now allowed. A message is logged instead of blocking the operation and empty operands are kept in the primitive tree.
- **EasyColor**
  - *(26.6.0)* The tool `Color transform` generated code errors. This has been fixed.
- **EasyImage**
  - *(26.6.0)* In the tool `Threshold`, when connecting an `EImageC24`, all the thresholds were not serialized. This has been fixed.
  - *(26.6.1)* In the tools `Resize` and `Circle warp`, the generated code included a wrong `using namespace` (C++), `using` (C#) or `import` (Python). This has been fixed.
- **EasyObject**
  - *(26.6.0)* In the tool `Object selection`, the coded image argument was missing in the generated code in the double-threshold feature filters. This has been fixed.
  - *(26.6.1)* In the tool `Checker`, the button `Display region` button was always visible and, when toggled, it displayed a region covering the entire image even when no inspection region input was connected. This has been fixed.
- **EasyMatch**
  - *(26.6.0)* The tool `Matcher` did not re-learn its pattern when the learn frame changed. This has been fixed.
  - *(26.6.0)* In the tool `Matcher`, when the parameter `Advanced learning` was disabled, the generated code set `AdvancedLearning` to `true` instead of `false`. This has been fixed.
  - *(26.6.0)* When using several tools `Matcher`, the generated code had several variables with the same name. This has been fixed.

- **EasyGauge**
  - *(26.6.0)* In the tool `Point gauge`, when the output `points` was connected and its type was `EPointSet`, the generated code used a variable `pointSet` instead of `points`, and the downstream tools were not wired correctly. This has been fixed.
  - *(26.6.0)* In the tool `Point gauge`, when the output `points` was connected and its type was `Array of EPoint`, an error occurred. This has been fixed.
  - *(26.6.0)* In the shapes and gauges tools, when `View parent` was selected in the display widget and the display widget was showing an ROI, the shape was drawn with an incorrect pan. This has been fixed.
  - *(26.6.0)* In the tool `Convolution`, `ConvolLowpass1` was missing from the code generation operation map. This has been fixed.
  - *(26.6.0)* In the tool `Morphology`, the input `EFrame` was not taken into account in the generated code. This has been fixed.
- **Easy3D**
  - *(26.6.0)* In the tool `3D Features aligner`, a flickering could occur in the 3DViewers. This has been fixed.
  - *(26.6.1)* In the tool `3D Principal axis extractor`, the port `referenceTransform` was displayed as an output. This has been fixed.
  - *(26.6.1)* In the tool `3D Principal axis extractor`, the input port `cloud` accepted an `E3DTransformMatrix`. This has been fixed.
- **Easy3DLaserLine**
  - *(26.6.0)* In the tool `3D Laser line object calibration`, the parameter `Along positive Z-axis` was not properly reset. This has been fixed.
  - *(26.6.0)* In the tool `3D Laser line scale calibration`, the parameter `Scale Z` was not properly saved. This has been fixed.
- **EasySpotDetector**
  - *(26.6.0)* In the tool `Spot detector`, when the output `spots` was connected, the generated code used a tool-prefixed variable name instead of `spots` and the downstream tools were not wired correctly. This has been fixed.
- Deep learning tools
  - *(26.6.0)* In the tools using deep learning, using the “default” engine with a NVIDIA GPU reinitialized the network for each inference leading to delays in the execution of the graph. This has been fixed.

4. Known Issues

All

- *(25.02.1)* Since version 24.02, **Open eVision** cannot read back the serialization of some objects made with previous versions.
  - `ERectangleRegion` or `E3DObject` are known to be affected by this issue.

eGrabber and VimbaX

- *(25.10.0)* To use **eGrabber** or **VimbaX** with **Open eVision** in C++, use the full header (`Open_eVision.h`) as there is currently an incompatibility with the per-library headers.

Open eVision License Manager

- The **Open eVision License Manager** might not start if the **.NET Framework 4.8** is not installed.
- Using **Open eVision License Manager** in English language mode on a Chinese or Japanese Windows version can lead to truncated text being displayed. This is an issue linked to the automatic font selection and there is currently no workaround. Please note however that, by default, the **Open eVision License Manager** runs in the OS language, including Chinese and Japanese.

Neo License Manager

- *(24.06.0)* In some cases, offline to offline reactivation may not work.

Deep Learning tools

- *(2.15.0)* The Deep Learning tool objects (`EClassifier`, `EUnsupervisedSegmenter`, `ESupervisedSegmenter` and `ELocator`) can leak CPU and GPU memory at destruction.
  - ► So, it is not recommended to create and delete a lot of Deep Learning tools in the same program.

Samples

- *(25.02.0)* In the provided Python samples using both **Open eVision** and **Qt**, a rare bug was reported causing failure at loading **Open eVision** shared library due to negative interaction with the Qt library on some computers only.
  - The conditions necessary to reproduce this bug are currently unknown.
  - A workaround and an explanation of this bug is now provided in those Python / Qt samples.

Licensing

On some installations, the licensing systems can take a long time to start (from 10 seconds up to a few minutes). If you have this issue, you can try the following procedures:

- Clean your software license cache.
  - The software license cache can become bloated by usage.
  - It can also happen if you use only dongles, as the system checks the presence of software licenses in all cases.
  - To clean the cache, use the `LicenseManager.exe /DeleteLicenseFiles` command.

  ⚠ This command deletes all the licenses that are not managed by the **Neo License Manager** on the system. Reactivate these licenses after the cleaning.
- Update your system root certificates.
  - If your root certificates are expired, the validation of the licensing system signatures might fail and timeout.
  - This only happens if the computer is on a network, even if the network is not connected to the Internet.
- Enable only the licensing system(s) you use.
  - By default, all the supported licensing systems are enabled.
  - Use the new (available from 2.13) `Preconfiguration::SelectLicensingModels` method to select exactly the licenses you want to enable and avoid issues arising from the usage of the other ones.

Image formats

- If you use some types of 96-bit RGB Tiff image, **Open eVision** may crash.

Memory leaks

- If you use the CRT library to detect memory leaks in your program, it can falsely detect some memory leaks when you use the **Open eVision** library.
  - This is a known limitation of the CRT library memory leak detection scheme.
    See: `https://docs.microsoft.com/en-us/visualstudio/debugger/ finding-memory-leaks-using-the-crt-library`
  - It happens when the memory leak detection scheme is ended before the **Open eVision** DLL is unloaded or the code in the **Open eVision** headers is uninitialized.

C# version

(24.10.2) Multiple methods taking arrays as `ref` parameters (for example: `EClassifier.Classify`) may generate random crashes of the application. This is a regression introduced following changes made in release 24.10.0.

Basic types: retrieving and setting pixel values

Using the `GetPixel()` and `SetPixel()` methods of the various ROI classes can sometimes be slow if you make many calls (regardless of the language used).

- In order to greatly speed up the ROI/image buffer access, embed the buffer access in your own code.
- See the examples below that use the new **Open eVision** API.

For a better readability of these examples, the variable declarations and initializations have been omitted when possible.

Example in C++

```
void* pixAddr;
UINT8 pix;
...
for (int y = 0; y < height; ++y)
{
  pixAddr = bw8Image.GetImagePtr(0,y);
  for (int x = 0; x < width; ++x)
  {
    pix = *(reinterpret_cast<UINT8*>(pixAddr)+x);
  }
}
```

Example in C#

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

Basic types: ROI zooming and panning issue

- When drawing an ROI with a zoom factor, applying panning (retrieved from a scroll bar) causes the ROI display to be shifted. Consequently, the `HitTest()` and `Drag()` functions fail because the handles do not appear at their actual positions.

  Workaround: The panning values should be divided by the zoom factor before calling the `DrawFrame()`, `HitTest()` and `Drag()` functions.

Basic Types: miscellaneous issues

- TIFF files containing RGB values + alpha values are not supported.
- Filenames with multibyte characters are not supported. The error is "Unrecognized file format".
  - Use UTF-8 encoded strings to handle filenames with non-latin characters.
- `Easy::GetBestMatchingImageType()` only works for BW8 and C24 images.

EasyBarCode

- EasyBarCode requires that a quiet zone of at least one full module is present around the whole bar code to be read.

EasyOCR2

- *(2.13.0)* The detection of a topology with ranged characters was always failing with the proportionnal detection method. This functionnality is now disabled and an error is thrown.

EasyImage

- *(24.10.2)* The constructor `EKernel` using an `EKernelType` as parameter does not use the same rectifier as the associated method `Convol`.
  - For instance, `EKernel(EKernelType_GradientX)` returns a kernel with a `EKernelRectifier_KeepPositive` rectifier while the method `ConvolGradientX` uses a `EKernelRectifier_Absolute` rectifier.
- *(24.10.2)* The method `ConvolKernel` does not have the same border value behavior (`constant`) as the pre-defined method like `ConvolGradientX` which is `mirror`.

EasyObject

- The `ECodedImage2` and `EHarrisDetector` results are drawn slowly when there are many results.

EasyMatch

- Matching a vertically symmetric pattern with an angle tolerance around 180° and in the original image can lead to an error of 1 pixel on the detected position.
- By default, EasyMatch interpolation does not work on 15 x 15 and smaller patterns.

  Workaround: For pattern sizes smaller than 16 x 16, adjust the `MinReduced` area to fit the `MinReducedArea < W*H/4` (if interpolation is needed).

EasyGauge

The following known issues only refer to the Legacy API of **EasyGauge** (in the namespace `Euresys::Open_eVision::Legacy`).

- In .NET, the `EPointGauge.GetMeasuredPoint()` overload with no argument is not available. To get the default measured point, use -1 as index.
- By design, an `ELineGauge`, `ERectangleGauge`, `ECircleGauge` or `EWedgeGauge` is reported as invalid if at least one of its sample points is invalid. In addition, these invalid sample points cannot be drawn as they have not been measured successfully.
- The `EWedgeGauge::SetActiveEdges()` method incorrectly gets the `EDragHandle_Edge_r` and `EDragHandle_Edge_RR` bits mixed up when processing its argument.

  Workaround: In order to activate the inner circle, set the `EDragHandle_Edge_RR` flag and use the `EDragHandle_Edge_r` flag to activate the outer circle.
- Using a gauge on an ROI leads to drawing problems.

  Workaround: Use the gauge on the parent image.

- In the custom `EDraggingMode_ToEdges` dragging mode, you cannot resize the nominal wedge gauge position using the on-screen handles, neither in a custom application nor in **Open eVision Studio** or in **Open eVision** Eval.

  Workaround: Enter numerical values for the wedge gauge position.

- *(25.06.0)* When attaching a gauge to a `worldshape` or another gauge, you must keep a reference of this gauge alive (that is keep the gauge in a variable or an array in your program) otherwise it "disappears" from the hierarchy (immediately in C++ , depending on the garbage collector in Python and C#).

EasyMatrixCode

- When grading is enabled, the optimizations are made in order to get accurate grading rather than have the best possible reading. As a result, the number of decoding errors reported with grading can be higher than without grading.
- Inspecting images with a lot of details, even if they are low contrast, can require much more time spent in EasyMatrixCode than the `TimeOut` set previously.
- In .NET, retrieving the coordinates of a MatrixCode using `EMatrixCode.GetCorner()` or `EMatrixCode.Center()` can lead to an unhandled exception when the garbage collection starts up. To avoid this problem, call `Dispose()` on the `EPoint` objects returned by these functions when they are no longer needed.

Easy3D

- *(2.16.0)* When using the class `EMeshToZMapConverter` without extension, some triangles of the mesh close to the ZMap border may be removed.
- *(22.04.1)* The Linux virtual machines do not support EDL in the `E3DViewer`.
- *(24.06.0)* When using the Python wrapper, if EDL is enabled on Linux (x64 and arm) platforms, the `E3DViewer` does not work properly.

Open eVision installer

- There is a conflict between the **Open eVision** installer and any program using the UDP:6001 port. When a software is already using this port, the installation fails and rolls back.

  Workaround: Install **Open eVision** first, and then the other software.

  This port is typically used by National Instrument software such as LabView.

- Before installing any Euresys product, make sure that your OS is up-to-date (using Microsoft Update), otherwise, problems might occur.

5. End of Life and Support

End of life announcements

OS, programming languages and IDEs

- .NET Framework
  - Starting with **Open eVision 25.02** (February 2025 release) the minimum required version for the .NET Framework will be 4.8.
- Windows OS
  - The support for **Windows 7** and **Windows 8** will stop with **Open eVision 25.10** (October 2025 release). **Open eVision 25.06** will be the latest release supporting **Windows 7** and **Windows 8**.

Open eVision soft-based / host licensing system

- The **Open eVision** soft-based / host licensing system will be progressively phased out.
  - It was introduced in 2007 with the first version of **Open eVision** and it is based on an old, and now obsolete technology.
  - It is superseded by the **Neo Licensing System**.

Milestones of the phase out period

- Since 1 January 2023:
  - There are no more sales of **Open eVision** soft-based / host licenses.
  - The related product codes are 4250 to 4289.
  - These products are removed from the price lists.
- Since 1 January 2024:
  - The support for the soft-based / host licenses is removed from the **Open eVision** libraries.
  - The **Open eVision** releases 24.02 and later are not be able to detect and use any soft-based / host licenses activated on the platform.
- Starting 1 January 2029:
  - The soft-based license operation server will be shut down.
  - Activating or recovering a soft-based / host license will not be possible after 2028.

Notes

- All applications using a soft-based / host license will continue to work “forever”, as long as the licenses have been activated and don’t require recovery.
- The **Open eVision Neo Licensing System** replaces the soft-based / host licensing system.
- The **Neo Licensing System** supports dongles and **Neo Software Containers** for licenses.
- The usage and features of the **Neo Software Containers** are the same as the old soft-based / host licenses.
- **eVision** licenses are not affected by these changes.

Support of older environments

32-bit libraries

- The **32-bit Open eVision** libraries (only available for **Windows**) were removed from **Open eVision** distributions in July 2023.
  - You should migrate to 64-bit if you need newer versions of **Open eVision**.
  - ► The last version with 32-bit support is **Open eVision 23.04**.

Soft-based / host licenses

- The support for the soft-based / host licenses is removed from the **Open eVision** libraries.
  - ► The last version to support soft-based / host licenses is **Open eVision 23.12**.

OS and processor architectures

OS		Last version with support
Windows 8®	64-bit	**Open eVision** 25.06
Windows 7® (*)	64-bit	
Windows 11®	32-bit	**Open eVision** 23.04
Windows 10®	32-bit	
Windows 8®	32-bit	
Windows 7®	32-bit	
Windows Vista	—	**Open eVision** 2.3.3.10777
Windows XP	—	**Open eVision** 2.2.2.10255
Windows 2000	—	**Open eVision** 1.0.1.5222

(*) The recommended version is 6.1.7601 (Windows 7 Service Pack 1)

Supported IDE

IDE	Last version with support
Microsoft Visual Studio 2008	**Open eVision** 22.12
Microsoft Visual Studio 2005	**Open eVision** 2.7
Microsoft Visual Studio 2003	
Microsoft Visual Studio 6.0	**Open eVision** 2.5.1.1107

Programming languages

Programming language	Last version with support
Borland C++ Builder 6.0	**Open eVision** 2.5.1.1107
CodeGear Delphi 2009	
CodeGear C++ Builder 2009	
ActiveX API	
Embarcadero RAD Studio XE4 and XE5	**Open eVision** 2.4.1.11114
Borland Delphi 6.0 and 2006	**Open eVision** 1.0.1.5222
Borland C++ Builder 2006	

Legacy Headers

- The **Legacy Headers** are removed from **Open eVision** distributions since the end of 2022.
  - The **Legacy Headers** help the customers to migrate an existing application using **eVision** (last major release in 2006) to **Open eVision**.
  - ▶ The last version with support is **Open eVision 22.12**.

3D Studio

- **3D Studio** is removed from the **Open eVision** package since the end of 2025.
  - ▶ We recommend prototyping your 3D pipeline in the **Open eVision Studio**.
  - ▶ The last version that includes the **3D Studio** application is **Open eVision 25.10.1**.

Legacy Open eVision Studio

- **Legacy Open eVision Studio** is removed from the **Open eVision** package since release 26.6.
  - ▶ We recommend prototyping with **Open eVision Studio**.
  - ▶ The last version that includes the **Legacy Open eVision Studio** application is **Open eVision 26.02.3**.

6. Deprecation Table

Deprecation process

- The deprecation of a feature is always announced in the release notes.
- The deprecated feature stays in the distribution (API, sample, **Open eVision Studio**...) for one year. After a year, it is completely removed from the distribution (exceptions may apply).

Deprecation table

Feature	Depr.	Recommended alternative	Remov.
EasySpotDetector tool (ESpot and ESpotDetector classes)	26.6	None	27.6
EObjectTemplateMatcher::EnableAlignment	26.6	EObjectTemplateMatcher::Alignment	27.6
EObjectTemplateMatcher::SortPositions	26.6	EObjectTemplateMatcher::Match	27.6
EBarCodeReader::EnablePermissiveDecoding	26.6	EnablePatternCorrection, EnableAdvancedDecoding, EnablePrintingErrorCorrection	27.6
EChecker2 (in **EasyObject**)	26.6	New EChecker class (in **EasyObject**)	27.6
EMatrixCodeReader, EBarCodeReader, EQRCodeReader, ECodeReader Read overloads without EGrid parameter	26.6	Overloads taking EGrid	27.6
ETextReader::Localize	26.6	ETextReader::Locate	27.6
SetImagePtr for classes EBaseROI, EFourierImage, EZMap and EDepthMap	26.02	Replaced by method UseExternalBuffer	27.6
EPoint constructor taking EFrame as argument	26.02	Other EPoint constructors	27.6
EFlipAxis enum values	26.02	New values following naming convention (for ex. EFlipAxis_Horizontal)	27.6
E3DPlane::FlipIfNegativeZ	26.02	E3DPlane::OrientNormal (E3DPlaneNormalOrientation_PosZ)	27.6
EPoint::SetCenter, EPoint::GetCenter, EPoint::SetCenterXY	25.10	Direct access to X and Y properties and other EPoint methods	27.6
EPointCloudMerger::Merge (non-pointer-vector overload)	25.06	EPointCloudMerger::Merge taking std::vector<EPointCloud*>	27.6
EPatternFinder::Learn with don't care area mask	25.02	Use ERegion in EPatternFinder::Learn	27.6
EMatcher::DontCareThreshold	25.02	Use ERegion in EMatcher::LearnPattern	27.6
EMatcher::GetNumPositions, EMatcher::GetPosition, EMatcher::GetPositions	25.02	A vector of EFoundPattern is returned by EMatcher::Match	27.6
EMatchPosition struct for EMatcher	25.02	EFoundPattern	27.6
EBarCodeReader::EnableAllSymbologies	25.02	EBarCodeReader::EnableSafeSymbologies	27.6
ESerializer::CreateFileWriter, CreateFileReader, CreateMemoryWriter, CreateMemoryReader, GetWriting	25.02	New EFileSerializer and EMemorySerializer constructors	27.6
EImageEncoder::Serialize	24.10	EImageEncoder::Save or EImageEncoder::Load	27.6
Serialize for EPen, EFont, EBrush classes	24.10	Methods Save or Load	27.6
EBarCodeReader:SetMinModuleSize, GetMinModuleSize, GetUseMinModuleSize, SetUseMinModuleSize	24.10	None	27.6
Easy::Initialize and Easy::Terminate	24.06	None (automatic)	27.6

Feature	Depr.	Recommended alternative	Remov.
EMatcher::GetExtension(int32_t&,int32_t&) and SetExtension(int32_t,int32_t)	24.06	EMatcher::Set/GetExtension with EIntSize type	27.6
EMatcher::GetPixelDimensions(float&,float&) and EMatcher::SetPixelDimensions(float,float)	24.06	EMatcher::Set/GetPixelDimensions with ESize type	27.6
E3DMatcher::GetAnomalyThresholds(float&,float&) and E3DMatcher::SetAnomalyThresholds(float,float)	24.06	E3DMatcher::Set/GetAnomalyThresholds with EAnomalyThresholds type	27.6
E3DComparer::GetAnomalyThresholds(float&,float&) and E3DComparer::SetAnomalyThresholds(float,float)	24.06	E3DComparer::Set/GetAnomalyThresholds with EAnomalyThresholds type	27.6
E3DMatcher::GetAnomalyHysteresis(float&,float&) and E3DMatcher::SetAnomalyHysteresis(float,float)	24.06	E3DMatcher::Set/GetAnomalyHysteresis with EAnomalyThresholds type	27.6
E3DComparer::GetAnomalyHysteresis(float&,float&) and E3DComparer::SetAnomalyHysteresis(float,float)	24.06	E3DComparer::Set/GetAnomalyHysteresis with EAnomalyThresholds type	27.6
E3DMatcher::GetEdgeCroppingParameters(int&,float&) and E3DMatcher::SetEdgeCroppingParameters (int,float)	24.06	E3DMatcher::Set/GetEdgeCroppingParameters with EEdgeCroppingParameters type	27.6
E3DComparer::GetEdgeCroppingParameters (int&,float&) and E3DComparer::SetEdgeCroppingParameters(int,float)	24.06	E3DComparer::Set/GetEdgeCroppingParameters with EEdgeCroppingParameters type	27.6
E3DTransformMatrix::GetAzimuthElevationAngles (float&,float&)	24.06	EAzimuthElevationAngles E3DTransformMatrix::GetAzimuthElevationAngles	27.6
E3DViewer::GetColorRampLocation (float&,float&,float&,float&) and E3DViewer::SetColorRampLocation (float,float,float,float)	24.06	E3DViewer::Set/GetColorRampLocation with EPoint parameters	27.6
E3DViewer::GetFixColorRampBounds(float&,float&) and E3DViewer::SetFixColorRampBounds(float,float)	24.06	E3DViewer::Set/GetFixColorRampBounds with EFloatRange type	27.6
E3DViewer::GetViewTarget(float&,float&,float&) and E3DViewer::SetViewTarget(float,float,float)	24.06	E3DViewer::Set/GetViewTarget with E3DPoint type	27.6
E3DViewer::GetViewAngle(float&,float&,float&) and E3DViewer::SetViewAngle(float,float,float)	24.06	E3DViewer::Set/GetViewAngle with E3DPoint type	27.6
E3DViewer::GetAutoRotate(float&,float&,float&) and E3DViewer::SetAutoRotate(float,float,float)	24.06	E3DViewer::Set/GetAutoRotate with E3DPoint type	27.6
ConvertCoordinatesMapToPixel (float,float,int&,int&) for EZMap and EDepthMap classes	24.06	ConvertCoordinatesMapToPixel (EPoint,EIntPoint&)	27.6
ConvertCoordinatesPixelToMap (int,int,float&,float&) for EZMap and EDepthMap classes	24.06	EPoint ConvertCoordinatesPixelToMap (EIntPoint)	27.6
EZMap::GetSizeInWorld, GetResolution, ImageToZMap, ZMapToImage, GetPixelPositionFromWorldPosition (ref-based overloads)	24.06	Overloads returning/using EPoint, E3DPoint, EIntPoint	27.6
E3DPlane EPlaneFinder::Find(EPointCloud,...)	24.06	float EPlaneFinder::Find (EPointCloud,E3DPlane&,...)	27.6
E3DPlane EPlaneFitter::Fit(EPointCloud,...)	24.06	float EPlaneFitter::Fit (EPointCloud,E3DPlane&)	27.6
ESphereFitter::Fit(...) returning E3DSphere	24.06	Version returning ratio of inliers with sphere as argument	27.6
EPrincipalAxisExtractor::Extract (ref-based overload)	24.06	Overload using E3DPoint	27.6
EStatistics::ComputeStatistics, EStatistics::ComputePixelStatistics (ref-based overloads)	24.06	Overloads returning EStatistics8, EStatistics16 or EStatistics32f	27.6
EPointCloud::AllocateAttributeBuffer, FillAttributeBuffer, AddCustomAttributeBuffer, AllocateCustomAttributeBuffer, SetAttribute (generic overloads)	24.06	Named-type method variants	27.6

Feature	Depr.	Recommended alternative	Remov.
EasyImage::HistogramThreshold and EasyImage::HistogramThresholdBW16 (ref-based overloads)	24.06	Overloads returning EHistogramThreshold	27.6
EasyImage::TwoLevelsMinResidueThreshold (ref-based overload)	24.06	Overload returning ETwoLevelsMinResidueThreshold	27.6
EasyImage::ThreeLevelsMinResidueThreshold (ref-based overload)	24.06	Overload returning EThreeLevelsMinResidueThreshold	27.6
EasyImage::Thin, EasyImage::BiLevelThin, EasyImage::Thick, EasyImage::BiLevelThick (void overloads)	24.06	Overloads returning the number of iterations performed	27.6
EasyImage::MatchFrames (all overloads)	24.06	Overloads returning the optimal shift amplitude	27.6
EKernel::GetKernelData (ref-based overload)	24.06	Overload returning the coefficient value	27.6
EMovingAverage::GetSize (ref-based overload)	24.06	Individual size getters	27.6
EasyImage::PixelAverage, PixelStat, PixelStatBW8, PixelStatBW16, PixelStdDev, PixelCount, Area, AreaDoubleThreshold, PixelVariance, GravityCenter, BinaryMoments, WeightedMoments	24.06	EPixelStatisticsBW8, EPixelStatisticsBW16, EPixelStatisticsC24 classes	27.6
EClassificationDataset::GetImages returning vector<EBaseROI*>	24.06	EClassificationDataset::GetImageCopy	27.6
EasyObject::ContourArea(EPathVector*, int32_t&)	24.06	int32_t EasyObject::ContourArea (EPathVector*)	27.6
EasyObject::ContourGravityCenter, EasyObject::ContourInertia	24.06	EasyObject::ContourStatistics	27.6
EImageEncoder::GetBinaryImageSegmenter and Segmenters::EBinaryImageSegmenter	24.06	None	27.6
ESelectByPosition, ESelectOption, ESortOption enums	24.06	ECodedImage2	27.6
SetMinNumFitSamples(int32_t, int32_t) for ECircleGauge, ELineGauge, ERectangleGauge, EWedgeGauge	24.06	Overloads using EIntegerRange	27.6
AddSkipRange and GetSkipRange (int32_t overloads) for ECircleGauge, ELineGauge, ERectangleGauge, EWedgeGauge	24.06	Overloads using EIntegerRange	27.6
EVectorModel::GetVectorModelExtremas (old overload)	24.06	Overloads using EFloatRange	27.6
EasyQRCodeReader::SetSearchField	24.06	EasyQRCodeReader::Read	27.6
EPlaneFinder::GetSeed and EPlaneFinder::SetSeed	24.02.1	None	27.6
E3DAnomaly::Create3DObject	24.02	E3DAnomaly::GetBoundingBox	27.6
OverallSymbolGrade field in EMatrixCodeIso15415GradingParameters, EMatrixCodeIso29158GradingParameters, EQRCodeIso15415GradingParameters	24.02	ScanGrade	27.6
EGDIPlusDrawAdapter	23.12	EWindowsDrawAdapter	27.6
Easy::OpenImageGraphicContext and Easy::CloseImageGraphicContext	23.12	None	27.6
EImageBW1	23.12	None	27.6
ECodedImage, EFeatureData, EListItem	23.12	ECodedImage2	27.6
EBarCode (**EasyBarCode1**)	23.12	Euresys::Open_eVision::EasyBarCode2::EBarCodeReader	27.6
EMatrixCode (**EasyMatrixCode1**)	23.12	Euresys::Open_eVision::EasyMatrixCode2::EMatrixCode	27.6
EMatrixCodeReader (**EasyMatrixCode1**)	23.12	Euresys::Open_eVision::EasyMatrixCode2::EMatrixCodeReader	27.6
ESearchParamsType	23.12	None	27.6
EasyColor::C24ToBayer(EROIC24*, EROIBW8*, bool evenCol, bool evenRow)	23.12	Overload taking EBayerConfiguration	27.6

Feature	Depr.	Recommended alternative	Remov.
EasyColor::TransformBayer(EROIBW8*, EROIBW8*, EColorLookup*, bool evenCol, bool evenRow)	23.12	Overload taking EBayerConfiguration	27.6
Draw methods with HDC API	23.12	Use the overload taking an EDrawAdapter by using an instance of EWindowsDrawAdapter	27.6
EOCR::ReadTextWide	23.08	EOCR::ReadText	27.6
EOCR::RecognizeWide	23.08	EOCR::Recognize	27.6
EDeepLearningTool::SetEnableGPU and EDeepLearningTool::GetEnableGPU	23.04	None	27.6
EDeepLearningTool::GetNumGPUs	23.04	EDeepLearningTool::NumAvailableDevices	27.6
EDeepLearningTool::SetGPUIndexes and EDeepLearningTool::GetGPUIndexes	23.04	EDeepLearningTool::ActiveDevices	27.6
EQuadrangle::IsInside()	23.04	EQuadrangle::IsPointInside	27.6
EBarCodeReader::GetValidateWithChecksum and EBarCodeReader::SetValidateWithChecksum	23.04	Set/GetValidateMandatoryChecksum, Set/GetValidateOptionalChecksum, Set/GetValidatePharmacode	27.6
EBarCodeReader::GetValidateBarCode and EBarCodeReader::SetValidateBarCode	23.04	Set/GetValidateMandatoryChecksum, Set/GetValidateOptionalChecksum, Set/GetValidatePharmacode	27.6
EClassifier::SetCapacity	22.12	EClassifier::SetModelType	27.6
ELocatorMetrics::GetBestWeightedFScoreAndThreshold	22.08	ELocatorMetrics::GetBestWeightedFScorePoint	27.6
ELocatorMetrics::GetBestWeightedPrecisionAndThreshold	22.08	ELocatorMetrics::GetBestWeightedPrecisionPoint	27.6
ELocatorMetrics::GetBestWeightedRecallAndThreshold	22.08	ELocatorMetrics::GetBestWeightedRecallPoint	27.6
ELocatorMetrics::GetLabelIntersectionOverUnion	22.08	ELocatorMetrics::GetLabelAverageProximity	27.6
ELocatorMetrics::GetIntersectionOverUnion	22.08	ELocatorMetrics::AverageProximity	27.6
EWorldShape::SetDistortion	2.16	EWorldShape::SetDistortionStrength	27.6
EChecker (**EasyOCV**)	2.15	**EChecker2**	27.6
EasyQRCode::Detect and EasyQRCode::Decode	2.13	EasyQRCode::Read	27.6
Tool **EasyOCV** (EOCV class, ECharCreationMode, EDegreesOfFreedom, EDiagnostic, ELearningMode, ELocationMode, EQualityIndicator enums)	2.11	None	27.6