Release Notes Open eVision 1.2.9
October 28, 2016

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

Contents 2

What's New in Open eVision 1.2.9? 4

Correction of various issues with Easy::ImageScaleRotate() and EasyImage::Register() 4
Resolved crash when using non-aligned image buffers with EasyMatch 4
The size of the oemKey needed by Easy::SetOemKey()has been enforced to 8 elements. 4

System Requirements 5

Operating Systems and Processor Architecture 5
Supported Integrated Development Environments and Programming Languages 6
Required system resources 7

Fixes and Improvements 8

Fixes and Improvements since Open eVision 1.2 8
Basic types 8
EasyColor 9
EasyGauge 9
EasyFind 11
EasyImage 12
EasyObject2 13
EasyObject 13
EasyOCR 13
EasyMatch 14
EasyMatrixCode 14
EasyBarCode 16
EasyQRCode 17
Installer 17
Licensing 17
Open eVision Studio/Open eVision Eval 17
Visual Studio 6 18
Delphi 18
System 18
Legacy API 18

Known Issues 19

Reserved keywords 19
ActiveX: Object cleanup 19
.NET: Object cleanup 19
Basic Types: retrieving and setting pixel values 21
Basic Types: ROI zooming and panning issue 22
Basic Types: miscellaneous issues 22
EasyBarCode 22
EasyQRCode 22
EasyObject 22
EasyMatch 22
EasyGauge 23
EasyMatrixCode 23
Open eVision Studio 23

Open eVision Eval Installer 23
Open eVision Installer 23
Open eVision License Manager 24

Migrating from Open eVision 1.0 25

Basic Types and Operations 25
EasyImage 27
EasyColor 29
EasyObject 29
EasyMatch 30
EasyFind 30
EasyGauge 31
EasyOCR 34
EasyOCV 35
EasyBarCode 35
EasyMatrixCode 35

What's New in Open eVision 1.2.9?

Correction of various issues with Easy::ImageScaleRotate() and EasyImage::Register()

There were some issues when using **`Easy::ImageScaleRotate()`** and **`EasyImage::Register()`** with large 64bit images and when using ROIs placed far from the origin. These have been fixed.

Resolved crash when using non-aligned image buffers with EasyMatch

When using image buffers with rows not aligned on 16bytes, a crash could occur in **`EasyMatch::Match()`**. This has been fixed. Please note that could only happen when using image buffers not allocated by Open eVision.

The size of the oemKey needed by Easy::SetOemKey()has been enforced to 8 elements.

An **`EError_CannotWriteOEMKey`** exception is thrown when calling **`Easy::SetOemKey()`** with a key length different to 8. The documentation has been updated accordingly.

System Requirements

Operating Systems and Processor Architecture

Open eVision 1.2.9 is a 32-bit and 64-bit library that requires a processor compatible with the SSE2 instruction set.

Open eVision 1.2.9 can be used on the following operating systems:

OS version	Additional information	
Windows 10® (*)	32-bit	—
Windows 10® (*)	64-bit	—
Windows 8® (*)	32-bit	—
Windows 8® (*)	64-bit	—
Windows 7® (*)	32-bit	—
Windows 7® (*)	64-bit	—
Windows Vista®	32-bit	Service Pack 1
Windows XP®	32-bit	Service Pack 3
Windows Server 2008®	32-bit	—
Windows Server 2008® R2	64-bit	—
Windows Server 2003®	32-bit	Service Pack 1
Windows Embedded Standard 2009®	32-bit	—

Note: The Open eVision 1.2.9 installer does not allow the installation of the product on virtual machines.

Supported Integrated Development Environments and Programming Languages

IDE	Language	Open eVision API module
Microsoft Visual Studio 6.0® SP6	C++	C++
	Basic	ActiveX Library
Microsoft Visual Studio .NET 2003® SP1	C++	C++
Microsoft Visual Studio 2005® SP1	C++	C++
	C#, VB .NET, C++/CLI	.NET Assembly
Microsoft Visual Studio 2008® SP1	C++	C++
	C#, VB .NET, C++/CLI	.NET Assembly
Microsoft Visual Studio 2010®	C++	C++
	C#, VB .NET, C++/CLI	.NET Assembly
Microsoft Visual Studio 2012®	C++	C++
	C#, VB .NET, C++/CLI	.NET Assembly
Microsoft Visual Studio 2013®	C++	C++
	C#, VB .NET, C++/CLI	.NET Assembly
Borland C++ Builder 6.0® update 4	C++	C++
CodeGear C++ Builder 2009®	C++	C++
CodeGear Delphi 2009®	Object Pascal	ActiveX Library
Embarcadero RAD Studio XE4	C++	C++
	Object Pascal	ActiveX Library
Embarcadero RAD Studio XE5	C++	C++
	Object Pascal	ActiveX Library

Note about IDE Detection and Configuration: the installers detect IDEs. Install your IDE first — it will be automatically configured when installing Open eVision. You should restart the installer to configure a *newly* installed IDE.

Required system resources

- Minimal display size: 800 x 600. 1280 x 1024 recommended.
- Minimal color depth: 16 bits. 32 bits recommended.
- Between 20 MB and 300 MB free hard disk space needed for Open eVision Libraries, depending upon selected options.
- Between 60 MB and 400 MB free hard disk space needed for Open eVision Development Tools, depending upon selected options.

For more details on the installation, please refer to the Open eVision documentation (*Installing Open eVision* section).

Fixes and Improvements

Fixes and Improvements since Open eVision 1.2

Basic types

- An **`EError_CannotWriteOEMKey`** exception is thrown when calling **`Easy::SetOemKey()`** with a key length different to 8. The documentation has been updated accordingly.
- When using anisotropic zoom settings, the drawing of ROIs could be incomplete. This has been fixed.
- A new **`EC48`** pixel type class has been added for RGB, 16bit per channel support. The corresponding ROI and image types **`EROIC48`** and **`EImageC48`** have been added as well.
- A new **`EAngleRectifier`** class has been added. This class allows you to modify the value of an angle to ensure its value is in a given (full circle) interval.

```
class EAngleRectifier
{
   static double Rectify(double angle, double middle);
   static double Rectify(double angle, double middle, EAngleUnit currentUnit);
};
```

`angle`: angle to rectify

`middle`: middle of the target interval, defined as **`]middle – pi; middle + pi]`**

`currentUnit`: the unit to use for the angle if you don't want to use the unit currently set in Open eVision.

Note: Due to the necessity to retrieve the current angle unit in the overload where it is not specified, it can be significantly slower. In performance-critical applications, the other should thus be preferred.

- New **`PixelAccessor`** classes have been added to Open eVision to provide access to the pixels of an image using **`GetPixel()/SetPixel()`** style functions. These classes manage **`EBW16`**, **`EBW32`**, **`EC15`**, **`EC16`**, **`EC24`** and **`EC24A`** pixel types like **`EBW8PixelAccessor`** manages the **`EBW8`** pixel type.
- Angle units were not handled correctly in **`Easy::Render3D()`**.This has been fixed. Moreover, **`Easy::Render3D()`** now handle angle units as has been set with **`Easy::SetAngleUnit()`** and not always in Degrees as previously stated in the documentation.
- Under .NET, OEMKey was not checked correctly. This has been fixed.
- A new **`EBW8PixelAccessor`** class has been added to Open eVision to provide fast access to the pixels of an image using **`GetPixel()/SetPixel()`** style functions.
- There was a potential memory leak when creating objects. This has been fixed.
- Some string fields of the **`EImageXXX`** objects were not serialized when using TIFF as the serialization target. This has been fixed.
- A crash could happen when calling the copy constructor of **`EImageXXX`** objects. This has been fixed.

EasyColor

- New functions and a new **EHDRColorFuser** class have been added to provide color image HDR Fusion support:

```
void EasyImage::HDRFusion(const EROIC24* darkSrc, const EROIC24* lightSrc,
                          int shutterSpeedFactor, EROIC48* dst);
void EasyImage::HDRFusion(const EROIC48* darkSrc, const EROIC48* lightSrc,
                          int shutterSpeedFactor, EROIC48* dst);
```

darkSrc	Dark input image (high shutter speed).
lightSrc	Light input image (low shutter speed).
shutterSpeedFactor	Shutter speed factor between light and dark image
dst	Destination image.

```
class EHDRColorFuser
{
      //Constructors. The image used must be the lightest
      //(slowest shutter speed).
      EHDRColorFuser(const EROIC24* lightSrc);
      EHDRColorFuser(const EROIC48* lightSrc);

      //Fuses a dark image with the light image passed during object
      //construction.
      void Fuse(const EROIC24* darkSrc, int shutterSpeedFactor);
      void Fuse(const EROIC48* darkSrc, int shutterSpeedFactor);

      //Retrieves the resulting image. Returns false if part of the potential
      //image dynamic is lost due to the chosen output image type.
      bool GetFusedImage(EROIC24* dst);
      bool GetFusedImage(EROIC48* dst);
};
```

Note : For optimal results, the images must be fused in increasing order of shutter speed.

EasyGauge

- New helper functions have been added to EasyGauge:

```
static FLOAT32 ELine::GetAngleBetweenLines(const ELine& line1, const ELine&
line2);
```

Computes the angle between two lines. The angle returned by this function is signed in the trigonometric sense, meaning that angle(1,2) = -angle(2,1).

line1: First line

line2: Second line

`static FLOAT32 ELine::GetDistanceBetweenPointAndLine(const EPoint& pt, const ELine& line, bool limited = false);`

Computes the distance between a point and a line.

`pt`: The point.

`line`: The line.

`limited`: Indicates if the line parameter should be considered as an infinite line or as a segment.

`static EPoint ELine::GetProjectionOfPointOnLine(const EPoint& pt, const ELine& line);`

Computes the projection of a point on a line.

`pt`: The point.

`line`: The line.

`static INT32 ELine::GetIntersectionOfLines(const ELine& line1, const ELine& line2, EPoint& intersection, bool limited = false);`

Computes the intersection between two lines. The function returns the number of intersections found. It will return -1 if the two lines are overlapping.

`line1`: First line.

`line2`: Second line.

`intersection`: Found intersection.

`limited`: Indicates if the line parameters should be considered as infinite lines or as a segments.

`static FLOAT32 ECircle::GetDistanceBetweenPointAndCircle(const EPoint& pt, const ECircle& circle, bool limited = false);`

Computes the distance between a point and a circle

`pt`: The point.

`circle`: The circle.

`limited`: Indicates if the circle parameter should be considered as a full circle or as an arc.

`static EPoint ECircle::GetProjectionOfPointOnCircle(const EPoint& pt, const ECircle& circle);`

Computes the projection of a point on a circle.

`pt`: The point.

`circle`: The circle.

`static FLOAT32 ECircle::GetDistanceBetweenLineAndCircle(const ELine& line, const ECircle& circle, bool limited = false);`

Computes the distance between a line and a circle.

`line`: The line.

`circle`: The circle.

`limited:` Indicates if the line and circle parameters should be considered as infinite lines and full circles or as a segments and arcs..

`static INT32 ECircle::GetIntersectionOfLineAndCircle(const ELine& line, const ECircle& circle, EPoint& intersection1, EPoint& intersection2, bool limited = false);`

Computes the intersections between a line and circle. Returns the number of intersections and stores the found intersections in the provided point parameters.

`line`: The line.

`circle`: The circle.

`intersection1`: First found intersection.

`intersection2`: Second found intersection.

`limited:` Indicates if the line and circle parameters should be considered as infinite lines and full circles or as a segments and arcs.

`static INT32 ECircle::GetIntersectionOfCircles(const ECircle& circle1, const ECircle& circle2, EPoint& intersection1, EPoint& intersection2, bool limited = false);`

Computes the intersections between two circles. Returns the number of intersections and stores the found intersections in the provided point parameters. Returns -1 if the two circles are overlapping.

`circle1`: First circle.

`circle2`: Second circle.

`intersection1`: First found intersection.

`intersection2`: Second found intersection.

`limited:` Indicates if the circle parameters should be considered as full circles or as arcs.

- In some cases, calling **`Measure()`** on a gauge placed in part outside of the image boundaries caused a crash. This has been fixed.

EasyFind

- Using **`EPatterFinder.Find()`** on very large images could lead to a crash. This has been fixed.
- The Contrasting Regions pattern type has been fixed and is now operational since Open eVision 1.2.1.
- EasyFind now allows overwriting an existing model file when saving a model with the **`EPatternFinder.Save()`** method.

EasyImage

- There was some issues when using **`Easy::ImageScaleRotate()`** and **`EasyImage::Register()`** with large images in 64bits and when using ROIs placed far from the origin. These have been fixed.
- New functions and a new **`EHDRFuser`** class have been added to provide grayscale image HDR Fusion support:

```
void EasyImage::HDRFusion(const EROIBW8* darkSrc, const EROIBW8* lightSrc,
                          int shutterSpeedFactor, EROIBW16* dst);
void EasyImage::HDRFusion(const EROIBW16* darkSrc, const EROIBW16* lightSrc,
                          int shutterSpeedFactor, EROIBW16* dst);
void EasyImage::HDRFusion(const EROIBW16* darkSrc, const EROIBW16* lightSrc,
                          int shutterSpeedFactor, EROIBW32* dst);
```

`darkSrc`	Dark input image (high shutter speed).
`lightSrc`	Light input image (low shutter speed).
`shutterSpeedFactor`	Shutter speed factor between light and dark image
`dst`	Destination image.

```
class EHDRFuser
{
      //Constructors. The image used must be the lightest
      //(lowest shutter speed).
      EHDRFuser(const EROIBW8* lightSrc);
      EHDRFuser(const EROIBW16* lightSrc);

      //Fuses a dark image with the light image passed during object
      //construction.
      void Fuse(const EROIBW8* darkSrc, int shutterSpeedFactor);
      void Fuse(const EROIBW16* darkSrc, int shutterSpeedFactor);

      //Retrieves the resulting image. Returns false if part of the potential
      //image dynamic is lost due to the chosen output image type.
      bool GetFusedImage(EROIBW8* dst);
      bool GetFusedImage(EROIBW16* dst);
      bool GetFusedImage(EROIBW32* dst);
};
```

Note : For optimal results, the images must be fused in increasing order of shutter speed.

- A new function **`EasyImage::AdaptiveThreshold`**, is available in Open eVision. This function performs a locally adaptive threshold on the source image.

```
void EasyImage::AdaptiveThreshold(const EROIBW8* src,
                                  EROIBW8* dst,
                                  EAdaptiveThresholdMethod method,
                                  int halfKernelSize,
                                  int constant = 0);
```

`src`	The source image/ROI.
`dst`	The destination image/ROI. Cannot be the same as **`src`**.
`method`	The thresholding mode.
`halfKernelSize`	Half size of the kernel rounded down.
`Constant`	Constant offset applied to the threshold value. By default 0.

```
enum AdaptiveThresholdMethod;
```

`EAdaptiveThresholdMethod_Mean`.	Use the mean as threshold.
`EAdaptiveThresholdMethod_Median`	Use the median as threshold.
`EAdaptiveThresholdMethod_Middle`	Use the middle of the values interval as threshold.

- Using **`EMovingAverage`** with a number of images to integrate that is a power of 2 could lead to incorrect results. This has been fixed.

EasyObject2

- Rarely, the Ellipse Height feature value was reported as indeterminate instead of zero. This has been fixed.
- Speed improvements have been made to the object selection functions.
- In some circumstances, a **`System.ExecutionEngineException`** could be thrown when using the .NET assembly. This has been fixed.
- There was a memory leak when retrieving **`EListItem`** class instances from an **`ECodedImage`** in C++. This has been solved.

EasyObject

- Speed improvements have been made to the **`ECodedImage.AnalyseObjects()`** function.

EasyOCR

- In some circumstances, the characters in the reported string could ben in the wrong order. This has been fixed.
- There was a potential memory leak when calling the **`EOCR.Recognize`** or **`EOCR.BuildObjects`** methods. This has been solved.

EasyMatch

- When using image buffers with rows not aligned on 16bytes, a crash could occur in **`EasyMatch::Match()`**. This has been fixed. Please note that could only happen when using image buffers not allocated by Open eVision.

EasyMatrixCode

- Further improvements have been made to the ISO 15415 Grid Non uniformity grading procedures of EasyMatrixCode.
- Numerical values for Axial NonUniformity, Grid non Uniformity, Symbol Contrast and Unused Error Correction have been added to the ISO 15415 results. The updated structure is thus as follows:

```
struct EMatrixCodeIso15415GradingParameters
{
public:
  INT32 DecodingGrade;
  INT32 AxialNonUniformityGrade;
  INT32 GridNonUniformityGrade;
  INT32 UnusedErrorCorrectionGrade;
  INT32 SymbolContrastGrade;
  INT32 ModulationGrade;
  INT32 ReflectanceMarginGrade;
  INT32 FixedPatternDamageGrade;
  INT32 OverallSymbolGrade;
  FLOAT32 HorizontalPrintGrowth;
  FLOAT32 VerticalPrintGrowth;
  FLOAT32 GridNonUniformity;
  FLOAT32 AxialNonUniformity;
  FLOAT32 SymbolContrast;
  FLOAT32 UnusedErrorCorrection;
};
```

- Improvements have been made in the grading procedures of EasyMatrixCode to improve reliability on difficult images.
- EasyMatrixCode has been updated to support Data Matrix Codes that are using the GS1 standard. For these, EasyMatrixCode will now return the decoded string with the relevant control characters added. Moreover, a new method, **`EMatrixCode::IsGS1()`** has been added to the **`EMatrixCode`** object. This method will return **`TRUE`** if the Matrix Code is using the GS1 standard and **`FALSE`** otherwise.
- Numerous improvements have been made in the localization, digitalization and grading procedures of EasyMatrixCode.
- The ISO/IEC 15415 Grid Non Uniformity grading procedure has been updated to give more accurate results.
- **Important**: For a successful grading process, the quiet zone (1 module) around the MatrixCode must be contained inside the image. If it is not the case, grading should not be enabled or an **`EError_CouldNotGrade`** error will be reported.

- ISO/IEC 15415 grading has been updated to include the informational values **`HorizontalPrintGrowth`** and **`VerticalPrintGrowth`**. The updated structure is thus as follows:

```
struct EMatrixCodeIso15415GradingParameters
{
public:
  INT32 DecodingGrade;
  INT32 AxialNonUniformityGrade;
  INT32 GridNonUniformityGrade;
  INT32 UnusedErrorCorrectionGrade;
  INT32 SymbolContrastGrade;
  INT32 ModulationGrade;
  INT32 ReflectanceMarginGrade;
  INT32 FixedPatternDamageGrade;
  INT32 OverallSymbolGrade;
  FLOAT32 HorizontalPrintGrowth;
  FLOAT32 VerticalPrintGrowth;
};
```

- The SEMI T10 grading procedures have been updated to better handle low-contrast Matrix Codes. Please note, however, that to get accurate grading values, symbol contrast should be high.
- Grading computation could give incorrect values when the Matrix Code was flipped. This has been fixed.
- In some cases, the Fixed Pattern Damage Grade for ISO/IEC 15415 and ISO/IEC TR 29158 could be miscalculated. This had been fixed.
- ISO/IEC 15415, ISO/IEC TR 29158 and SEMI T10-0701 support has been added to EasyMatrixCode. To compute the SEMI T10 metrics, **`EMatrixCodeReader::SetComputeGrading()`** must be set to **`TRUE`**.

  The function needed to retrieve the grading parameters are the following:

```
EMatrixCodeSemiT10GradingParameters EMatrixCode::GetSemiT10GradingParameters();

EMatrixCodeIso15415GradingParameters EMatrixCode::GetIso15415GradingParameters();

EMatrixCodeIso29158GradingParameters EMatrixCode::GetIso29158GradingParameters();
```

  And the structures returned, containing the grading parameters, are the following:

```
struct EMatrixCodeSemiT10GradingParameters
{
public:
  FLOAT32 SymbolContrast;
  FLOAT32 SymbolContrastSNR;
  FLOAT32 HorizontalMarkGrowth;
  FLOAT32 VerticalMarkGrowth;
  FLOAT32 DataMatrixCellWidth;
  FLOAT32 DataMatrixCellHeight;
  FLOAT32 HorizontalMarkMisplacement;
  FLOAT32 VerticalMarkMisplacement;
  FLOAT32 UnusedErrorCorrection;
  FLOAT32 CellDefects;
  FLOAT32 FinderPatternDefects;
};
```

```
struct EMatrixCodeIso15415GradingParameters
{
public:
  INT32 DecodingGrade;
  INT32 AxialNonUniformityGrade;
  INT32 GridNonUniformityGrade;
  INT32 UnusedErrorCorrectionGrade;
  INT32 SymbolContrastGrade;
  INT32 ModulationGrade;
  INT32 ReflectanceMarginGrade;
  INT32 FixedPatternDamageGrade;
  INT32 OverallSymbolGrade;
};


struct EMatrixCodeIso29158GradingParameters
{
public:
  INT32 CellContrastGrade;
  INT32 CellModulationGrade;
  INT32 MinimumReflectanceGrade;
  INT32 FixedPatternDamageGrade;
  INT32 OverallSymbolGrade;
  FLOAT32 MeanLight;
  BOOL IsMeanLightInRequiredBounds;
};
```

- When restricting EasyMatrixCode to a given Contrast Mode, a MatrixCode of the opposite Contrast Mode could erroneously be returned. This has been fixed.
- The **`Read()`** function caused an unhandled exception when called after a call to **`ClearLogicalSize().`** This has been fixed.
- In .NET, there was a potential delayed crash when retrieving the coordinates of the MatrixCode. This has been fixed.
- In C++, there was a crash when exiting a process where eVision objects were constructed but not used. This has been fixed.
- There was a memory leak in **`EMatrixCodeReader.Read()`** when the timeout was reached. This has been fixed.
- The **`EMatrixCodeReader.TimeOut`** property is now correctly serialized.

EasyBarCode

- Improvements have been made to EasyBarCode to better handle barcodes whose modules vary along the length.
- In 64bit, when using the known location mode, a failed barcode decoding could lead to a crash in **`EBarCode::Read()`**. This has been fixed.

EasyQRCode

- A timeout functionality has been added to EasyQRCode. To set a timeout, use the following property:

  ```
  void EQRCodeReader::SetTimeOut(UINT64 value);
  ```

  This property sets a timeout in microseconds for **`Detect()`**, **`Decode()`**, and **`Read()`**.
- EasyQRCode under VC++ 6.0 could lead to a linker error. This has been fixed.
- A new QR Code detection mode has been implemented in **`EQRCodeReader`** to handle stronger perspective deformations. To enable this new mode, use the **`EQRCodeReader::SetPerspectiveMode()`** method with the value **`EQRCodePerspectiveMode_Improved`**.

  Note: For better detection results, don't forget to also set the **`EQRCodeReader::SetMinimumIsotropy()`** property to an appropriate value.
- A new **`EQRCodeReader::SetCellPolarityConfidenceThreshold()`** threshold has been added to EasyQRCode. This property, when set to a value higher than 0.0f, will enable post-processing on low-confidence cells to improve the quality of the digitization. The closer the value will be to 1.0f, the more cells will become candidates for that post-processing.
- There was a potential crash while reading QRCodes. This has been fixed.

Installer

- Sometimes, when uninstalling Open eVision, the uninstaller remained stuck on the "Unregistering ActiveX" phase. This has been fixed.
- The installers were reporting an error message when being executed on physical machines where VMWare products were installed. This has been solved.

Licensing

- The new EasyQRCode license was not granted when using emergency licenses. This has been fixed.
- A new **`Easy::CheckLicense()`** function has been added to Open eVision to allow checking if a specific license is granted on the current system.
- There was a potential slow down when using dongles on machines where portable licenses had previously been activated and returned.

Open eVision Studio/Open eVision Eval

- The description of Min and Max in the pixel statistics dialog was incorrect. It has been fixed.
- Reloading an image in Open eVision after saving it could lead to an "Unable to create document! Please choose another name" error. This has been fixed.
- There was a crash when loading a color image without the Easy Color license. This has been fixed.

- The EasyColor → Get/Set Color Component feature now works correctly even if the provided color image is not in the RGB color format. It now works if the image is RGB, if the lookup table is RGB or if the image and the lookup table have the same color format (possibly different from RGB).
- Selecting the 'ISH' entry from the 'Color system' combo of the 'Color Conversion' dialog box had no effect; the color system remained the one previously selected. This has been fixed.
- Saved Visual Basic scripts now use the correct file extension.

Visual Studio 6

- Building with Microsoft Visual Studio 6.0 was inoperative. This has been fixed.

Delphi

- Some methods featuring OLEVariant arguments were unusable. Some functions has been added to avoid the use of OLEVariant.

System

- **`Easy::SetOemKey()`** and **`Easy::CheckOemKey()`** have a strict requirement on the length of the key being 8 characters.
- There was an issue when compiling with Open eVision 64-bit with C++ Builder XE4/XE5. This has been fixed.
- A wide optimization campaign has been made on Open eVision since 1.2.1, resulting in faster operation across the board compared to Open eVision 1.2.
- Administrator rights were needed to run an application using Open eVision. This has been solved.

Legacy API

- The Legacy API was not working with Visual Studio 2013. This has been fixed.
- In some cases, using the Legacy API C++ header in an ActiveX could prevent it to successfully register. This has been fixed.
- The alternate set of C++ headers that allows developing or porting code against the older API that was supplied with eVision 6.7.1 (and lower) and Open eVision 1.0 is now available for 64bit compilers.
    - See the "*Open eVision 1.2 – Legacy API Support*" document for detailed information.

Known Issues

Reserved keywords

The following keywords are reserved by Open eVision: **`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`**, **`EasyWorld`**.

Variables, functions, methods, macros and such should not be named using those to avoid conflict.

ActiveX: Object cleanup

As a rule, it is highly recommended to call **`.Dispose`** on Open eVision ActiveX objects when they are not useful anymore. Not doing so might result in unnecessarily high memory usage and crashes.

Example in VB

```
src = New EImageBW8
finder = New EPatternFinder

src.Load ImageFilePath
foundPatterns = finder.Find(src)
…
For Each foundPattern In foundPatterns
 foundPattern.Dispose
Next
finder.Dispose
src.Dispose
```

Moreover, if you used a nested object, like the segmenter properties in EasyObject's encoder objects, it is important to remember to call **`Dispose`** on that object before calling **`Dispose`** on the parent object.

Example in VB

```
imageEncoder.GrayscaleSingleThresholdSegmenter.BlackLayerEncoded = True
…
imageEncoder.GrayscaleSingleThresholdSegmenter.Dispose
imageEncoder.Dispose
```

.NET: Object cleanup

As a rule, it is highly recommended to call **`.Dispose()`** on Open eVision .NET objects when they are not useful anymore. Not doing so might result in unnecessarily high memory usage and crashes.

Example in C#

```
using(EImageBW8 src = new EImageBW8())
using(EPatternFinder finder = new EPatternFinder())
{
  src.Load(ImageFilePath);
  EFoundPattern[] foundPatterns = finder.Find(src);
  …
  foreach(EFoundPattern foundPattern in foundPatterns)
  {
    foundPattern.Dispose();
  }
}
```

Moreover, if you used a nested object, like the segmenter properties in EasyObject's encoder objects, it is important to remember to call **`Dispose()`** on that object before calling **`Dispose()`** on the parent object.

Example in C#

```
imageEncoder.GrayscaleSingleThresholdSegmenter.BlackLayerEncoded = true;
…
imageEncoder.GrayscaleSingleThresholdSegmenter.Dispose();
imageEncoder.Dispose();
```

Basic Types: retrieving and setting pixel values

- Using the **GetPixel()** and **SetPixel()** methods of the various ROI classes can sometimes be slow if many calls are made (regardless of the language used).

  In order to greatly speed up ROI/image buffer access, you can embed the buffer access in your own code.

  You can find some examples below, using the new Open eVision API. For the sake of readability, variable declarations and initializations have been omitted when possible.

Example in C++

```
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)
```

Example in Visual Basic 6.0

```
Private Declare Sub GetMem1 Lib "msvbvm60" (ByVal Addr As Long, RetVal As Byte)
…
Dim curAddr As Long
Dim pix As Byte
…
For y = 0 To h - 1
      curAddr = bw8Image.GetImagePtrXY(0, y)
      For x = 0 To w - 1
            GetMem1 curAddr, pix
            curAddr = curAddr + 1
      Next 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".
- **`Easy::GetBestMatchingImageType()`** only works for BW8 and C24 images.

EasyBarCode

- Due to a bug in the debugger of Visual C++ 2012, the reading time of barcodes may increase after a failed read. This happens only in debug mode and under Visual C++ 2012.
- EasyBarCode requires that a quiet zone of at least one full module is present around the whole barcode to be read.
- EasyBarCode is currently unable to read barcode with curved or bended bars. For reliable reading, the bars must be as straight as possible.
- EasyBarCode is currently not multithread-safe.

EasyQRCode

- MicroQR code support is unavailable in EasyQRCode.

EasyObject

- **`ECodedImage2`** and **`EHarrisDetector`** results draw slowly when there are many results.

EasyMatch

- The maximum size for a pattern in EasyMatch is 1791x1791. This is by design.
- 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.
- EasyMatch interpolation does not work by default on 15x15 and smaller patterns. As a workaround, for pattern sizes smaller than 16x16, the **`MinReduced`** area needs to be adjusted to fit **`MinreducedArea < W*H/4`** (if interpolation is needed).

EasyGauge

- Under .NET, the **`EPointGauge.GetMeasuredPoint()`** overload with no argument is unavailable. To get the default measured point, use -1 as index.
- By design, an **`ELineGauge, ERectangleGauge, ECircleGauge`** or **`EWedgeGauge`** is reported invalid is at least one of its sample points is invalid. Moreover, these invalid sample points cannot be drawn since they haven't been measured successfully.
- The **`EWedgeGauge::SetActiveEdges()`** method incorrectly gets the **`EDragHandle_Edge_r`** and **`EDragHandle_Edge_RR`** bits mixed up when processing its argument. As a workaround, in order to activate the inner circle, the **`EDragHandle_Edge_RR`** flag needs to be set and, conversely, the **`EDragHandle_Edge_r`** value will toggle the outer circle.
- Using a gauge on an ROI leads to drawing problems. As a workaround, use the gauge on the parent image instead.
- In the custom **`EDraggingMode_ToEdges`** dragging mode, it is not possible to resize the nominal wedge gauge position using the on-screen handles, be it in a custom application or in Open eVision Studio or Open eVision Eval. As a workaround, enter numerical values for the wedge gauge position.

EasyMatrixCode

- When grading is enabled, optimizations are made in order to get accurate grading rather than have the best read possible. As such, the number of decoding errors reported when grading can be higher than when not grading.
- When inspecting images with a lot of details, even if those are low contrast, the time effectively spent into EasyMatrixCode can be significantly superior to 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.

Open eVision Studio

- When used from the libraries, the **`EWorldShape`** object only requires that any Open eVision license be present. Unlike libraries, Open eVision Studio currently requires the EasyGauge license to display the EWorldShape/Gauges dialog box.

Open eVision Eval Installer

- When installing Open eVision Eval, if the chosen installation folder contains invalid characters, there is no error message but the invalid characters are removed from the folder. In the very particular case where the folder name contains ONLY invalid characters, the folder name is simply removed and the product gets installed in the parent folder.

Open eVision Installer

- Prior to installing any Euresys product, the OS must be up-to-date (using Microsoft Update). Otherwise, problems can occur.

- When there is not enough free disk space to complete the installation, there is no explicit error message. Clicking the install button is not possible (it is grayed out). Please note, though, that the required space and available space are both displayed.
- The C++ include directory settings are not configured in Visual Studio .NET 2003. As a workaround, manually add the include directory to the general IDE options.
- The C++ include directory settings are not configured in any Visual Studio version for 64-bit development. As a workaround, manually add the include directory to the general IDE options.
- The Open eVision-specific Visual Studio .NET 2003 configuration items (include and library directories) are not removed during uninstallation. They have to be deleted manually.

Open eVision License Manager

- When activating an emergency license, the following error may occur:

  "Error Message: Loading of the ASR failed!"

  This error occurs when all 3 emergency licenses have already been used and the computer has been formatted before trying again.

- Using Open eVision License Manager in English language mode under a Chinese or Japanese Windows version can lead to truncated text being displayed. This is an issue with the automatic font selection. There is currently no workaround. Please note however that, by default, the License Manager will run in the OS language, including Chinese and Japanese.

Migrating from Open eVision 1.0

The interface of Open eVision has been modernized and has substantially changed between Open eVision 1.0 and Open Vision 1.2 The following chapter lays out the differences between the two APIs.

Unless stated otherwise, the following remarks apply to the C++, .NET and ActiveX interfaces.

Basic Types and Operations

- [C++, .NET] All classes and structures now have a leading **`E`**, and are members of the **`Euresys::Open_eVision_1_2`** namespace.
- [C++] The enumeration type names in capital letters separated by underscores are now using "CamelCase", and begin with the letter E.

  Example: **`enum WEEK_DAY { DAY_MONDAY = 0, DAY_TUESDAY = 1, ... }`** becomes **`enum EWeekDay { EWeekDay_DayMonday = 0, EWeekDay_DayTuesday = 1, ... }`**.
- [C++] Global functions have been moved, and are now static methods of classes.
- [C++] The **`EOpenImageDC`** function is replaced by **`Easy::OpenImageGraphicContext`**.
- [C++] The **`ECloseImageDC`** function is replaced by **`Easy::CloseImageGraphicContext`**.
- [C++] **`EResize`** was a global function; it is moved in the class **`Easy`** and renamed as **`Resize`**.
- [C++] The **`EPeaksVector`** class is renamed as **`EPeakVector`**.
- [C++] Now, methods throw exception instead of setting the global error codes.
- [C++] Strings previously stored in **`char*`** or **`const char*`** are now stored in **`std::string`** or **`std::wstring`**.
- [C++] The struct variables like **`m_f32R`**, **`n32Size`** ... have lost their prefixes, and become **`R`**, **`Size`**…
- The **`EImageXXX`** constructor, that allowed to specify the row alignment in bytes, has been removed. A workaround is to allocate the buffer and use **`SetImagePtr`**.
- The .NET method **`SetImagePointer`** now has the same name as the C++ version (**`SetImagePtr`**).
- The ROI constructor taking an image pointer as argument has been removed, because it was highly confusing with the copy constructor. Instead, call the **`EROIXXX.Attach`** method after the ROI construction.
- The **`EROIXXX.Detach`** method has been removed. ROIs can only be in a detached state right after construction.
- [C++] **`Exception::Error`** is replaced by **`EException::GetError`**.
- All the methods that filled a buffer with text and required the user to specify the buffer length (for instance, **`EBarCode.Read`**) now return a string instead (**`std::string`** in C++, **`System.String`** in .NET).
- **`SetRecursiveCopyBehavior`** and **`GetRecursiveCopyBehavior`** have been removed. Hierarchy copying through a constructor copy is ALWAYS recursive. To avoid this recursion, use the **`CopyTo`** method instead.
- **`Easy.Initialize`** and **`Easy.Terminate`** are now useless and have been removed.
- All the **`EROIXXX`** classes now derive from an abstract class named **`EBaseROI`** and they inherit from all their properties and methods. Each **`EImageXXX`** class derives from the corresponding **`EROIXXX`** class.
- In the previous Open eVision versions, all the ROI classes had a constructor that took a pointer to a parent ROI as the first parameter and, optionally, position and size parameters. This constructor has been removed. On the other hand, the **`EBaseROI.Attach`** method has been augmented with parameters allowing to set the parent, position and resize in one shot.

  The following has been removed:

```
EROIXXX::EROIXXX(EROIXXX* parent, int x = 0, int y = 0, int w = 0, int h = 0);
```

The following has been added:

```
void EROIXXX::Attach(EROIXXX* parent, int x = 0, int y = 0, int w = 0, int h = 0);
```

Another advantage of this change is the availability of this method in ActiveX, while constructors featuring arguments are not supported in ActiveX.

- Previously, when an ROI was placed out of its parent image, it was silently resized or repositioned; in some cases, when automatically resized, the ROI could grow. Now, there's no silent resizing or repositioning anymore. Whenever a call on a ROI partially outside the image is made, an exception is thrown. To crop an ROI which is partially out of its image, the new method **`CropToImage`** must be called explicitly.
- [C++] The **`Save`** and **`Load`** methods of the **`EROIXXX`** objects can now be used to load/save image files for both standard and internal Euresys serialization formats.
- Load /Saving images into files
  - The **`EBaseROI.Load/EBaseROI.Save`** method of Open eVision 1.2 loads/saves the image data of an image object from/into a file. It is applicable to all Image types.
  - **`EBaseROI.SaveJpeg`** and **`EBaseROI.SaveJpeg2K`** have been added. They provide the capability to specify the compression quality when saving images into a compressed file format.
  - **`Easy.GetBestMatchingImageType`** returns the best matching image type for a given file on disk.
- [C++] **`EROIXXX::GetPixelDimensions`**, **`SetPixelDimensions`**, **`GetResolution`**, and **`SetResolution`** have been removed.
- [C++] **`EROIXXX::GetVoid`** is renamed as **`IsVoid`**. This method is used to test if the underlying buffer of an image is NULL.
- [C++] **`EXXXVector::GetDataPtr`** was returning a **`XXX*`**; now **`EXXXVector::GetRawDataPtr`** is returning a **`void*`**. The **`GetDataPtr`** method has been removed.
- The following method:
  ```
  void EC24Vector::Draw(HDC graphicContext, FLOAT32 width, FLOAT32 height, FLOAT32 originX=0.0f, FLOAT32 originY=0.0f, const ERGBColor& color0 = ERGBColor(-1 , -1 , -1), const ERGBColor& color1 = ERGBColor(-1 , -1 , -1), const ERGBColor& color2 = ERGBColor(-1 , -1 , -1));
  ```
  is now split in four sub-methods:
  - **`void EC24Vector::Draw(HDC graphicContext, FLOAT32 width, FLOAT32 height);`**
  - **`void EC24Vector::Draw(HDC graphicContext, FLOAT32 width, FLOAT32 height, FLOAT32 originX, FLOAT32 originY);`**
    This sub-method is named **`DrawPanned`** into the ActiveX API.
  - **`void EC24Vector::Draw(HDC graphicContext, FLOAT32 width, FLOAT32 height, FLOAT32 originX, FLOAT32 originY, const ERGBColor& color0, const ERGBColor& color1, const ERGBColor& color2);`**
    This sub-method is named **`DrawPannedWithColors`** into the ActiveX API.
  - **`void EC24Vector::Draw(HDC graphicContext, FLOAT32 width, FLOAT32 height, const ERGBColor& color0, const ERGBColor& color1, const ERGBColor& color2);`**
    This sub-method is named **`DrawWithColors`** into the ActiveX API.
- The following method:
  ```
  void EC24Vector::Draw(EDrawAdapter* graphicContext, FLOAT32 width, FLOAT32 height, FLOAT32 originX=0.0f, FLOAT32 originY=0.0f, const ERGBColor& color0 = ERGBColor(-1 , -1 , -1), const ERGBColor& color1 = ERGBColor(-1 , -1 , -1), const ERGBColor& color2 = ERGBColor(-1 , -1 , -1));
  ```
  is now split in four sub-methods:

- **`void EC24Vector::Draw(EDrawAdapter* graphicContext, FLOAT32 width, FLOAT32 height);`**
  This sub-method is named **`DrawWithAdapter`** into the ActiveX API.
- **`void EC24Vector::Draw(EDrawAdapter* graphicContext, FLOAT32 width, FLOAT32 height, FLOAT32 originX, FLOAT32 originY);`**
  This sub-method is named **`DrawWithAdapterPanned`** into the ActiveX API.
- **`void EC24Vector::Draw(EDrawAdapter* graphicContext, FLOAT32 width, FLOAT32 height, FLOAT32 originX, FLOAT32 originY, const ERGBColor& color0, const ERGBColor& color1, const ERGBColor& color2);`**
  This sub-method is named **`DrawWithAdapterPannedWithColors`** into the ActiveX API.
- **`void EC24Vector::Draw(EDrawAdapter* graphicContext, FLOAT32 width, FLOAT32 height, const ERGBColor& color0, const ERGBColor& color1, const ERGBColor& color2);`**
  This sub-method is named **`DrawWithAdapterWithColors`** into the ActiveX API.

- In the following method:
  `void EDrawAdapter::FilledRectangle(const int orgX, const int orgY, const int width, const int height, const ERGBColor& traceColor = ERGBColor::NoColor, const ERGBColor& fillColor = ERGBColor::NoColor);`
  the last two arguments **`traceColor`** and **`fillColor`** may not be used alone; if one is used, the other must also be used.

  The same rule applies for all derivatives of **`EDrawAdapter`** (**`GDIDrawAdapter`**, ...).

EasyImage

- The global functions called **`ImgXXX`** are now static methods of the **`EasyImage`** class and must be called with **`EasyImage::XXX`**.
- [C++] **`FLOAT32* EKernel::GetDataPtr`** is replaced by **`void* EKernel::GetRawDataPtr`**.
- The **`EasyImage::WeightedMoments`** and **`EasyImage::BinaryMoments`** methods now consider that the center of the pixels is shift by 0.5 pixels. This is a better convention when dealing with sub-pixel coordinates.
- The unwarping mechanism in **`EWorldShape`** allows using a LUT to speedup the unwarping. This LUT used to be an **`EImageSubPixel64`** object. Now it uses an **`EUnwarpingLut`** object. Otherwise, the API usage regarding unwarping has not changed.
- The **`EasyImage::Convert`** method now specifies two overloads for each pixel combination. One overload assumes that the mapping from source pixel to destination uses the maximum destination headroom, while another one accepts a typed pixel structure instead of an integer, as in the previous Open eVision version.
  For instance, the following method:
  `Convert(EROIBW1* sourceImage, EROIBW8* destinationImage, UINT8 highValue = UINT8_MAX);`
  now becomes:
  `Convert(EROIBW1* sourceImage, EROIBW8* destinationImage);`
  `Convert(EROIBW1* sourceImage, EROIBW8* destinationImage, EBW8 highValue);`
- The methods accepting a callback, namely **`Count`**, **`ImgTransform`** and **`ClrTransform`**, are removed.

  Consequently:

- The following code that was using the **`Count`** method:

```
BOOL Odd(EBW8& Pixel) {
  return (Pixel & 1) > 0;
}
// Count pixels with an odd gray-level value
UINT32 Count= ImgCount(&Image, Odd);
```

should be replaced by the following code:

```
int width = image.GetWidth();
int height = image.GetHeight();
UINT8* line = image.GetImagePtr();
int count = 0;
for(int y = 0; y < height; y++)
{
  for(int x = 0; x < width; x++)
    if( line[x] & 1 > 0 )
      count++;
  line += image.GetRowPitch();
}
```

Notice that the new code is much more efficient, and the execution time is CONSIDERABLY reduced.

- LUT-based processing can be used to replace the **`ImgTransform`** and **`ClrTransform`** functions.

- The following function:
`void EasyImage::GainOffset(EROIC24* sourceImage, EROIC24* destinationImage, EColor Gain = EColor(1.f, 1.f, 1.f), EColor Offset = EColor(0.f, 0.f, 0.f));`
is no more available. It has been replaced by the following methods:
  - **`void EasyImage::GainOffset(EROIC24* sourceImage, EROIC24* destinationImage, EColor Gain, EColor Offset);`**
  - **`void EasyImage::Gain(EROIC24* sourceImage, EROIC24* destinationImage, EColor Gain);`**
  - **`void EasyImage::Offset(EROIC24* sourceImage, EROIC24* destinationImage, EColor Offset);`**

- The arguments of the following method:
`EasyImage::Threshold(EROIBW16* sourceImage, EROIBW16* destinationImage, UINT32 threshold = EThresholdMode_MinResidue, EBW16 lowValue = 0, EBW16 highValue = 65535, FLOAT32 relativeThreshold = 0.5f);`
are modified. Following overloads are now available:
  - **`void EasyImage::Threshold(EROIBW16* sourceImage, EROIBW16* destinationImage);`**
    performs a thresholding using the minimum residue method.
  - **`void EasyImage::Threshold(EROIBW16* sourceImage, EROIBW16* destinationImage, UINT32 threshold);`**
    performs a thresholding using the supplied threshold value.
  - **`void EasyImage::Threshold(EROIBW16* sourceImage, EROIBW16* destinationImage, UINT32 threshold, EBW16 lowValue, EBW16 highValue);`**
    performs a thresholding using the supplied threshold value, and using supplied **`lowValue`** and **`highValue`** in the resulting image.

- **`void EasyImage::Threshold(EROIBW16* sourceImage, EROIBW16* destinationImage, FLOAT32 relativeThreshold);`**
  performs a thresholding using the supplied relative threshold value.
- **`void EasyImage::Threshold(EROIBW16* sourceImage, EROIBW16* destinationImage, FLOAT32 relativeThreshold, EBW16 lowValue, EBW16 highValue);`**
  performs a thresholding using the supplied relative threshold value, and using supplied **`lowValue`** and **`highValue`** in the resulting image.

- In the following method:
  `void EasyImage::Threshold(EROIC24* sourceImage, EROIBW8* destinationImage, EC24 minimum, EC24 maximum, EColorLookup* colorLookupTable, EBW8 rejectValue, EBW8 acceptValue);`
  the last two arguments **`rejectValue`** and **`acceptValue`** may not be used alone; if one is used, the other must also be used.
- In the following method:
  `void EasyImage::DoubleThreshold(EROIBW16* sourceImage, EROIBW16* destinationImage, UINT32 lowThreshold, UINT32 highThreshold, EBW16 lowValue, EBW16 middleValue, EBW16 highValue);`
  the last three arguments **`lowValue`**, **`middleValue`** and **`highValue`** may not be used alone; if one is used, the two others must also be used.
- [ActiveX] The **`ECannyEdgeDetector.Scale`** property has been renamed as **`ECannyEdgeDetector.Scale_`** because of a clash with an unknown Visual Basic keyword.

EasyColor

- The global functions called **`ClrXXX`** are now static methods of the **`EasyColor`** class and must be called with **`EasyColor::XXX`**.
- **`EColor`** was a union; it is now a struct and has only 3 members C0, C1, C2.

EasyObject

- There is no more restriction on which segmenters can be used in conjunction with the continuous mode.
- Clean separation between the concepts of objects, holes, coded images and encoders.
- The concept of "class" has been renamed as that of "layer", in order to remove ambiguities between the "programming language classes" and the "coded image classes".
- Adding/removing objects or holes to an **`EObjectSelection`** object invalidates the order in which the objects/holes are returned: A new call to **`Sort`** is necessary.
- EasyObject now uses a pixel coordinate system where the origin is conventionally at the top left corner of the top left pixel of an image. Consequently, the fractional part of the coordinates of the center of a pixel is ".5". This is a change of convention with respect to the legacy **`ECodedImage`** class. This convention is better suited for the representation of sub-pixel coordinates.
- In **`ECodedImage`** class, the **`Feature`** enumeration used in the **`ECodedImage`** object is renamed as **`ELegacyFeature`**. The **`EFeature`** name now belongs to a new enumeration used by **`ECodedImage2`**.
- The following features have disappeared: **`OBJ_GRAVITY_CENTER`**, **`OBJ_LIMIT`**, **`OBJ_LIMIT45`**, **`OBJ_ELLIPSE`** and **`OBJ_CENTROID`**. These were just convenience features when drawing objects.
- **`FeretBoxXXX`** (**`ECodedImage`**) are replaced by **`MinimumEnclosingRectangleXXX`** (**`ECodedImage2`**).
- **`LimitAngledXXX`** (**`ECodedImage`**) are replaced by **`FeretBoxXXX`** (**`ECodedImage2`**).
- **`LimitXXX`** (**`ECodedImage`**) are replaced by **`BoundingBoxXXX`** (**`ECodedImage2`**).

- Accordingly to the mathematical conventions, the angles are measured clockwisely in **ECodedImage2**: this allows bringing the X-axis on the Y-axis with a positive 90° rotation, which is not the case in **ECodedImage**.
- **Limit22XXX** (**ECodedImage**) are replaced by **FeretBox68XXX** (**ECodedImage2**). Because of the change in the angle measurements (cf. above), **Limit22Width** becomes **Limit68Height**, and **Limit22Height** becomes **Limit68Width**.
- **Limit45XXX** (**ECodedImage**) are replaced by **FeretBox45XXX** (**ECodedImage2**).
- **Limit68XXX** (**ECodedImage**) are replaced by **FeretBox22XXX** (**ECodedImage2**). Because of the change in the angle measurements (cf. above), **Limit68Width** becomes **Limit22Height**, and **Limit68Height** becomes **Limit22Width**.
- **Limit45Width** and **Limit45Height** are reduced by "1/sqrt(2)" (for implementation problem in **ECodedImage**).
- 1 is subtracted from **LimitWidth** (resp. **LimitHeight**), in order **FeretBoxWidth** (resp. **FeretBoxHeight**) to give the same result as **LimitWidth** when the Feret angle is set to zero.
- Previously, **ECodedImage::SetThreshold** had a default value for its argument. Now, you must provide an argument to **ECodedImage::SetThreshold**. Calling it with **EThresholdMode_MinResidue** is equivalent to calling it without argument in Open eVision 1.0.1.

EasyMatch

- The enum values for **EMatchContrastMode** are renamed as **EMatchContrastMode_Normal**, **EMatchContrastMode_Inverse**, **EMatchContrastMode_Any**, and **EMatchContrastMode_Unknown**.
- [C++, .NET] The enumeration **E_CORRELATION_MODE** is renamed as **ECorrelationMode**.
- [C++, .NET] The enumeration **MCH_CONTRAST_MODE** is renamed as **EMatchContrastMode**.
- [C++, .NET] The enumeration **MCH_FILTERING_MODE** is renamed as **EFilteringMode**.
- [.NET] **Matcher.CreateBW8PatternCopy** and **Matcher.CreateC24PatternCopy** have been removed, and are replaced by **EMatcher.CopyLearntPattern(EImageBW8& image)** and **EMatcher.CopyLearntPattern(EImageC24& image)**.
- [C++] **EMatch::CreateBW8PatternCopy** and **EMatch::CreateC24PatternCopy** have been removed, and are replaced by **EMatcher::CopyLearntPattern(EImageBW8& image)** and **EMatcher::CopyLearntPattern(EImageC24& image)**.
- [C++] **EMatchPosition* EMatch::GetPosition** is replaced by **EMatchPosition EMatcher::GetPosition**.

EasyFind

- [C++, .NET] The **PatternFinder** class is renamed as **EPatternFinder**.
- [C++, .NET] The **FoundPattern** class is renamed as **EFoundPattern**.
- [C++] The **EFoundPattern**, **EPatternFinder** classes were using properties; they now use getters and setters.
- [C++, .NET] **PatternFinder::CreateBW8PatternCopy** is renamed as **EPatternFinder::CopyLearntPattern(EImageBW8& image)**.
- [C++] **FoundPattern::Center** was of **Point** type; it is now of **EPoint** type. Thus, you must use **EFoundPattern::GetX**, **EFoundPattern::GetY**, **EFoundPattern::SetX**, or **EFoundPattern::SetY** to access X and Y.

- [C++] **`FoundPattern::DrawFeaturePoints`** has been removed. You may select to draw the features points using **`EFoundPattern::(Get/Set)DrawFeaturesPoints`**.
- [C++] **`FoundPattern::LearningDone`** is renamed as **`EPatternFinder::GetLearningDone`**.
- [C++] The arguments of **`Learn`** were references; they are now pointers.
- [C++] The argument of **`Find`** was a reference; it is now a pointer.
- Three new contrast modes have been added. These new modes are distinguished by the substring "PointByPoint". They compute a mathing score instead of a global fashion. The default value of the **`EFindContrastMode`** remains Normal.
- [C++, .NET] The enumeration **`EasyFind::Contrast::Type`** is renamed as **`EFindContrastMode`**.
- [C++, .NET] The enumeration **`EasyFind::LocalSearchMode::Type`** is renamed as **`ELocalSearchMode`**.
- [C++, .NET] The enumeration **`EasyFind::PatternType::Type`** is renamed as **`EPatternType`**.
- [C++, .NET] The enumeration **`EasyFind::ReductionMode::Type`** is renamed as **`EReductionMode`**.
- [C++, .NET] The enumeration **`EasyFind::ThinStructureMode::Type`** is renamed as **`EThinStructureMode`**.

EasyGauge

- [C++] In the **`EPoint`** and **`EXXXGauge`** classes, the 2-arguments overload **`SetCenter(FLOAT32 x, FLOAT32 y)`** is renamed as **`SetCenterXY(FLOAT32 x, FLOAT32 y)`**; the single-argument overload **`SetCenter(EPoint center)`** remains unchanged.
- **`EPoint::Set(x, y)`** is replaced by **`EPoint::SetCenterXY(x, y)`**.
- **`EFrame::Set(centerX, centerY, angle, scale)`** is suppressed. It is replaced by **`SetCenterXY(centerX, centerY)`**; **`SetAngle(angle)`**; **`SetScale(amplitude)`**.
- **`EFrame::Set(center, angle, scale)`** is suppressed. It is replaced by **`SetCenter(center)`**; **`SetAngle(angle)`**; **`SetScale(scale)`**.
- **`EFrameShape::Set(centerX, centerY, angle, scale)`** is suppressed. It is replaced by **`SetCenterXY(centerX, centerY)`**; **`SetAngle(angle)`**; **`SetScale(amplitude)`**.
- **`EFrameShape::Set(center, angle, scale)`** is suppressed. It is replaced by **`SetCenter(center)`**; **`SetAngle(angle)`**; **`SetScale(scale)`**.
- **`ELine::Set(center, length, angle)`** is suppressed. It is replaced by **`SetCenter(center)`**; **`SetLength(length)`**; **`SetAngle(angle)`**.
- **`ELine::Set(origin, end)`** is renamed as **`SetFromTwoPoints(origin, end)`**.
- **`ELineShape::Set(centerX, centerY, length, angle)`** is suppressed. It is replaced by **`SetCenterXY(centerX, centerY)`**; **`SetLength(length)`**; **`SetAngle(angle)`**.
- **`ELineShape::Set(origin, end)`** is renamed as **`SetFromTwoPoints(origin, end)`**.
- **`ELineGauge::Set(centerX, centerY, length, angle)`** is suppressed. It is replaced by **`SetCenterXY(centerX, centerY)`**; **`SetLength(length)`**; **`SetAngle(angle)`**.
- **`ELineGauge::Set(origin, end)`** is renamed as **`SetFromTwoPoints(origin, end)`**.
- **`ERectangle::Set(centerX, centerY, sizeX, sizeY, angle)`** is suppressed. It is replaced by **`SetCenterXY(centerX, centerY)`**; **`SetSize(sizeX, sizeY)`**; **`SetAngle(angle)`**.
- **`ERectangle::Set(origin, end)`** is renamed as **`SetFromTwoPoints(origin, end)`**.
- **`ERectangle::Set(origin, middle, end)`** is renamed as **`SetFromOriginMiddleEnd(origin, middle, end)`**.

- **ERectangleShape::Set(centerX, centerY, sizeX, sizeY, angle)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetSize(sizeX, sizeY)**; **SetAngle(angle)**.
- **ERectangleShape::Set(origin, end)** is renamed as **SetFromTwoPoints(origin, end)**.
- **ERectangleShape::Set(origin, middle, end)** is renamed as **SetFromOriginMiddleEnd(origin, middle, end)**.
- **ERectangleGauge::Set(centerX, centerY, sizeX, sizeY, angle)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetSize(sizeX, sizeY)**; **SetAngle(angle)**.
- **ERectangleGauge::Set(origin, end)** is renamed as **SetFromTwoPoints(origin, end)**.
- **ERectangleGauge::Set(origin, middle, end)** is renamed as **SetFromOriginMiddleEnd(origin, middle, end)**.
- **ECircle::Set(centerX, centerY, diameter, angle, amplitude)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetDiameter(diameter)**; **SetAngle(angle)**; **SetAmplitude(amplitude)**.
- **ECircle::Set(origin, end, bDirect)** is renamed as **SetFromTwoPoints(origin, end, bDirect)**.
- **ECircle::Set(origin, middle, end, bFull)** is renamed as **SetFromOriginMiddleEnd(origin, middle, end, bFull)**.
- **ECircleShape::Set(centerX, centerY, diameter, angle, amplitude)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetDiameter(diameter)**; **SetAngle(angle)**; **SetAmplitude(amplitude)**.
- **ECircleShape::Set(origin, end, bDirect)** is renamed as **SetFromTwoPoints(origin, end, bDirect)**.
- **ECircleShape::Set(origin, middle, end, bFull)** is renamed as **SetFromOriginMiddleEnd(origin, middle, end, bFull)**.
- **ECircleShape::Set(circle)** is renamed as **SetCircle(circle)**.
- **ECircleGauge::Set(centerX, centerY, diameter, angle, amplitude)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetDiameter(diameter)**; **SetAngle(angle)**; **SetAmplitude(amplitude)**.
- **ECircleGauge::Set(origin, end, bDirect)** is renamed as **SetFromTwoPoints(origin, end, bDirect)**.
- **ECircleGauge::Set(origin, middle, end, bFull)** is renamed as **SetFromOriginMiddleEnd(origin, middle, end, bFull)**.
- **ECircleGauge::Set(circle)** is renamed as **SetCircle(circle)**.
- **EWedge::Set(centerX, centerY, diameter, breadth, angle, amplitude)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetDiameters(diameter, breadth)**; **SetAngle(angle)**; **SetAmplitude(amplitude)**.
- **EWedge::Set(origin, end, breadth, bDirect)** is renamed as **SetFromTwoPoints(origin, end, breadth, bDirect)**.
- **EWedge::Set(origin, middle, end, breadth, bFull)** is renamed as **SetFromOriginMiddleEnd(origin, middle, end, breadth, bFull)**.
- **EWedgeShape::Set(centerX, centerY, diameter, breadth, angle, amplitude)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetDiameters(diameter, breadth)**; **SetAngle(angle)**; **SetAmplitude(amplitude)**.
- **EWedgeShape::Set(origin, end, breadth, bDirect)** is renamed as **SetFromTwoPoints(origin, end, breadth, bDirect)**.

- **`EWedgeShape::Set(origin, middle, end, breadth, bFull)`** is renamed as **`SetFromOriginMiddleEnd(origin, middle, end, breadth, bFull)`**.
- **`EWedgeShape::Set(wedge)`** is renamed as **`SetWedge(wedge)`**.
- **`EWedgeGauge::Set(centerX, centerY, diameter, breadth, angle, amplitude)`** is suppressed. It is replaced by **`SetCenterXY(centerX, centerY)`**; **`SetDiameters(diameter, breadth)`**; **`SetAngle(angle)`**; **`SetAmplitude(amplitude)`**.
- **`EWedgeGauge::Set(origin, end, breadth, bDirect)`** is renamed as **`SetFromTwoPoints(origin, end, breadth, bDirect)`**.
- **`EWedgeGauge::Set(origin, middle, end, breadth, bFull)`** is renamed as **`SetFromOriginMiddleEnd(origin, middle, end, breadth, bFull)`**.
- **`EWedgeGauge::Set(wedge)`** is renamed as **`SetWedge(wedge)`**.
- The 2-arguments overload **`EPointGauge::SetTolerance(tolerance, angle)`** is renamed as **`EPointGauge::SetTolerances(tolerance, angle)`**.
  The following methods are now available:
  `EPointGauge::SetTolerance(tolerance)` and **`EPointGauge::SetToleranceAngle(angle)`**.
- The **`EShape::GetNameUnicode`** method does not exist anymore.
- The **`ELandmark`** class is now documented.
- **`EWorldShape::HitLandMark`** is renamed as **`HitLandMarks`**.
- The default argument of the following methods is suppressed:
  ```
  void ECircleGauge::SetCircle(const ECircle& circle = ECircle(EPoint(0, 0), 2));
  void ELineGauge::SetLine(const ELine& line = ELine(EPoint(0, 0), 2));
  void EWedgeGauge::SetWedge(const EWedge& wedge = EWedge(EPoint(0, 0), 1, 1));
  ```
- The following method:
  ```
  UINT32 EWorldShape::RebuildGrid(FLOAT32 colPitch, FLOAT32 rowPitch, UINT32
  centerIndex = ~0, const EPoint& worldCenter = EPoint(0, 0), BOOL direct = TRUE);
  ```
  is now split in two sub-methods:
  - **`UINT32 RebuildGrid(FLOAT32 colPitch, FLOAT32 rowPitch, UINT32 centerIndex = ~0, BOOL direct = TRUE);`**
  - **`UINT32 RebuildGrid(FLOAT32 colPitch, FLOAT32 rowPitch, const EPoint& worldCenter, UINT32 centerIndex = ~0, BOOL direct = TRUE);`**
    This second sub-method is named **`RebuildGridTranslated`** into the ActiveX API.

- Using Open eVision with Visual Basic 6.0 requires that the member names differ regardless of the case. To keep a universal naming, the following changes have been made in both C++ and .NET:

Old name	New name
DragHandle_Tol_X0	EDragHandle_Tol_XX0
DragHandle_Tol_X1	EDragHandle_Tol_XX1
DragHandle_Tol_Y0	EDragHandle_Tol_YY0
DragHandle_Tol_Y1	EDragHandle_Tol_YY1
DragHandle_Tol_A0	EDragHandle_Tol_AA0
DragHandle_Tol_A1	EDragHandle_Tol_AA1
DragHandle_Tol_R0	EDragHandle_Tol_RR0
DragHandle_Tol_R1	EDragHandle_Tol_RR1
DragHandle_Edge_X	EDragHandle_Edge_XX
DragHandle_Edge_Y	EDragHandle_Edge_YY
DragHandle_Edge_A	EDragHandle_Edge_AA
DragHandle_Edge_R	EDragHandle_Edge_RR
DragHandle_Tol_X0	EDragHandle_Tol_XX0
DragHandle_Tol_X1	EDragHandle_Tol_XX1
DragHandle_Tol_Y0	EDragHandle_Tol_YY0
DragHandle_Tol_Y1	EDragHandle_Tol_YY1
DragHandle_Tol_A0	EDragHandle_Tol_AA0
DragHandle_Tol_A1	EDragHandle_Tol_AA1
DragHandle_Tol_R0	EDragHandle_Tol_RR0
DragHandle_Tol_R1	EDragHandle_Tol_RR1
DragHandle_Edge_X	EDragHandle_Edge_XX
DragHandle_Edge_Y	EDragHandle_Edge_YY
DragHandle_Edge_A	EDragHandle_Edge_AA
DragHandle_Edge_R	EDragHandle_Edge_RR

EasyOCR

- The number of available classes is now 31 instead of 32 (the last bit is now reserved for special purposes). This means that the **EOcrClass._31** (.NET) and **EOcrClass__31** (C++) enumeration value have been removed (formerly **OCR_CLASS_31** in C++).
- The output of **OCR:ReadText** and **OCR::Recognize** was passed by reference; it is now returned by the method.
- The unicode version of **EOCR::ReadText** and **EOCR::Recognize** are now called **EOCR::ReadTextWide** and **EOCR::RecognizeWide**.
- **EBW8* GetPatternBitmap(INT32 index)** is replaced by **EImageBW8* GetPatternBitmap(INT32 index)**. This method now returns a reference to the pattern image. This reference should not be deleted.

EasyOCV

- [C++] The **EOCV**, **EOCVChar**, **EOCVText** classes had member variables. They have been replaced by getters and setters.
- The **EOCV::m_TemplateImage** member variable can now be accessed through the **EOCV::GetLearnedTemplateImage** method. Since modifying this variable does not make sense, no **EOCV::SetLearnedTemplateImage** is supplied.

EasyBarCode

- **EBarCode::Set(centerX, centerY, sizeX, sizeY, angle)** is suppressed. It is replaced by **SetCenterXY(centerX, centerY)**; **SetSize(sizeX, sizeY)**; **SetAngle(angle)**.
- **EBarCode::Set(rectangle)** is renamed as **SetRectangle(rectangle)**.

EasyMatrixCode

- [C++] The **EMatrixCode**, **EMatrixCodeReader** classes were using properties; they now use getters and setters.
- **SearchParamsType** was a struct; it is now a class.
- **SearchParamsType** had 4 properties (vectors); it is now replaced by **ESearchParamsType** that has 16 methods **AddXXX**, **RemoveXXX**, **GetXXXCount**, and **GetXXX**.
- The enum values for **EMatrixCodeContrastMode** are renamed as **EMatrixCodeContrastMode_BlackOnWhite**, and **EMatrixCodeContrastMode_WhiteOnBlack**.
- The enumeration **EasyMatrixCode::Family::Type** is renamed as **EFamily**.
- The enumeration **EasyMatrixCode::Flipping::Type** is renamed as **EFlipping**.
- The enumeration **EasyMatrixCode::LearnParam::Type** is renamed as **ELearnParam**.
- The enumeration **EasyMatrixCode::LogicalSize::Type** is renamed as **ELogicalSize**.
- The enumeration **EasyMatrixCode::Contrast::Type** is renamed as **EMatrixCodeContrastMode**.