Release Notes Open eVision 1.1.5.5321
March 29, 2010

Contents

What's new?

Open eVision 1.1.5

Open eVision 1.1.5 adds the following features:

Operating Systems and Processor Architecture

Windows 7 (32-bit and 64-bit versions) is now supported.

Licensing

The License Manager now supports online license returns through a command-line flag. See the License Manager documentation for more information.

Open eVision 1.1

EasyImage

The newly included features are:

- **Flexible Masks**: applying a mask on an image restricts the processing to unmasked pixels of the image. The Open eVision masks are *flexible* by comparison to the rectangular ROIs; they support complex and disconnected shapes. EasyImage supports flexible masks as an argument for selected functions.
- **Canny Edge Detector**: known as an optimal edge detector, the Canny edge detector operates on a grayscale BW8 image and delivers a black-and-white BW8 image where pixels have only 2 possible values: 0 and 255.
- **Harris Corner Detector**: the Harris corner detector is popular due to its strong invariance to rotation, illumination variation and image noise. This implementation of the Harris corner detector operates exclusively on grayscale BW8 images.
- **Hit-and-Miss Transform**: this morphological operator detects a particular pattern of foreground and background pixels in an image. The EasyImage implementation of the hit-and-miss filter operates on grayscale and color images.

EasyObject

The EasyObject library has been **entirely redesigned**. The main **`ECodedImage`** class has been superseded by the **`ECodedImage2`** class.

- The main concepts for blob analysis are now represented by distinct classes instead of a single, monolithic class.
  - The new object-oriented API has separate classes for a clean separation between the concepts (encoding, object selection, object feature extraction, etc.).
  - The features of the objects have an improved access. The objects and the holes can be efficiently accessed randomly (i.e. in an index-based fashion).

- **Flexible masks** restrict the blob analysis to complex and/or disconnected regions of the image. They are available as an extra argument to the **`Encode`** method. It is also possible to generate a flexible mask from a coded image or from a selection of objects through the **`RenderMask`** method.
- The EasyObject algorithms now behave much better when the image size and/or the number of object runs increases. EasyObject is now **globally faster**.
- Other features:
  - Support of the encoding of BW1, BW8, BW16 and C24 source images.
  - Addition of the ability to compute an object *eccentricity*.
- For maintenance purpose, the legacy **`ECodedImage`** class is still available and documented in a dedicated section.

EasyFind

- EasyFind now features a new **scoring method** for the *consistent edges* operating mode that improves both speed and robustness in case of large occlusions. The new scoring method relies on a comparison of the similarity of the feature points taken independently, rather than in a global way.
- There are three new **contrast modes** to configure this new scoring method.

EasyGauge

- New methods to retrieve the **position of the samples** for line, circle, wedge, and rectangle gauges.
- New methods to retrieve the **samples peaks** for wedges and rectangle gauges.

Supported Image File Types

- Newly supported image file types: **JPEG-2000** and **PNG**.

Thread Safety

Open eVision is thread-safe. This means that it is designed to support simultaneous execution by multiple threads. It is particularly suitable if your application includes independent tasks able to be scheduled independently.

Error Reporting

From Open eVision 1.1 on, all methods throw exceptions instead of returning error codes. The error code is still supplied in the exception class instance.

Changes in the Drawing Methods

The drawing methods now provide default colors similar to the ones used in Open eVision Eval and Open eVision Studio. You do not have to select a pen if you want to use those colors.

To allow custom pens (for custom color, patterns or pen widths, for instance), a **`DrawWithCurrentPen`** method is now available.

Operating Systems and Processor Architecture

Windows Vista and Windows Server 2008 are now supported.

Please note that Open eVision does not support Windows 2000 anymore.

Integrated Development Environments

Microsoft Visual Studio .NET 2008, CodeGear C++ Builder 2009 and CodeGear Delphi 2009 are now supported.

Borland C++ Builder 2006, Borland Delphi 6.0, and Borland Delphi 2006 are not supported anymore.

Since Open eVision 1.1, the core DLL of the eVision libraries is common to all supported IDEs and languages. This avoids unexpected differences of behavior between the various languages/IDE.

An Application Can Now Use Multiple Versions of Open eVision

It is now allowed to install several versions of Open eVision on the same machine. For instance, Open eVision 1.1 can be installed alongside Open eVision 1.0.1.

Moreover, it is possible to mix several Open eVision (or eVision) versions in the same program. To allow differentiating between objects, they now belong to a version-specific namespace, from Open eVision 1.1 on. For instance, the API objects of Open eVision X.Y belong to the **`Euresys::Open_eVision_X_Y`** namespace.

New Open eVision Learning Accessories

- **Restructured documentation**: Open eVision comes with a comprehensive and re-structured documentation per programming interface (C++, .NET and ActiveX). Each of these three guides is split into a Functional Guide and a Programming Guide. The Programming Guide contains a comprehensive reference to the API elements, as well as code snippets that demonstrate the concepts and techniques explained in the Functional Guide.
- **Practical and didactic project and application samples**: the **project samples** illustrate how to use the Open eVision libraries with a particular IDE. The **application sample programs** illustrate the combined use of different libraries in a specific application. A variety of combination and applications are represented.

Compatibility Issues with Previous Versions

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_1`** 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 1.1 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 1.1 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**.

Fixes and Improvements

Fixes and Improvements in Release 1.1.5.5321

Open eVision Studio/Open eVision Eval

Script generation errors

There were errors in the Visual Basic 6.0 code generated by Open eVision Studio. They have been fixed.

EasyMatrixCode

Memory leaks when calling the Learn or Read method of EMatrixCodeReader under .NET

A small amount of memory was leaked when using the EMatrixCodeReader class under .NET. This has been fixed.

Fixes and Improvements in Release 1.1.5.4980

Open eVision Studio/Open eVision Eval

Inconsistent ROI OrgX and OrgY values

The OrgX and OrgY values for an ROI display 0 (zero) even when the ROI origin is outside of its parent image limits. This has been solved.

EasyGauge

Handle leaks when calling the ECircleGauge drawing methods

GDI objects were leaked when calling some drawing methods. This has been solved.

EasyMatrixCode

Memory leaks when calling the Learn or Read method of EMatrixCodeReader

Under some circumstances, memory buffers were kept allocated in subsequent calls of these methods. This has been solved.

Fixes and Improvements in Release 1.1.0.4690

Open eVision Studio/Open eVision Eval

Internal error when drawing EROIBW8

Exception sometimes occurred when calling the Draw method of EROIBW8 instance, when using the Open eVision .NET assembly. This has been solved.

ECodedImage2 and EHarrisDetector results draw slowly when there are many results

This has been solved.

Help files not available through Open eVision Studio menus

This has been solved.

Possible crash when no Open eVision Studio license available

This has been solved.

Display problems under Japanese Windows version

Under a Japanese version of Windows XP or Windows Vista, the EWorldShape dialog box of Open eVision Studio or Open eVision Eval features check boxes that are displayed incorrectly (camera and empirical model). This has been solved.

Problems in the E...Gauge dialog boxes

The following problems did apply to all gauging tools:

- when "Labeled" was checked, a crash occurred;
- the "Draw Nominal" and "Draw Actual" check boxes did not work.

This has been solved.

Script language selection has been improved (available via the "Welcome Screen")

A toolbar button and a menu entry has been added to allow changing the script language (C++, C# or Visual Basic 6).

Clicking on an encoded image did not select objects if the image was zoomed and/or scrolled.

This has been solved.

Drag & drop of an image to Open eVision Studio/Eval did not work anymore

This has been solved.

Menu entries greyed out in Open eVision Studio

The License Manager Help menu entry was greyed out in Open eVision Studio.

This has been solved. Also, menu entries have been improved and rearranged in Open eVision Eval.

Lines were added to the script window even when script generation was disabled.

This has been solved.

Found pattern display issues

The found pattern overlay was sometimes not refreshed after changing EPatternFinder options. This has been solved.

The color system setting is not refreshed in the Get/Set Component dialog

This has been solved.

Selecting "ISH" inside the color thresholding dialog had no effect

This has been solved.

Internal Error when comparing two images in the Arithmetic and Logic tool

This has been solved.

The enumeration names are not correct with the EasyObject tool

This has been solved.

The "Information..." menu entry is grayed out in the image popup menu

This has been solved.

The EasyMatrixCode overlay vanishes after saving the model.

This has been solved.

The instance name was missing from the result window title bar.

This has been solved.

The EasyObject overlay vanishes after selection.

This has been solved.

Installer and Licensing

Licensing service uninstalled by even when another version still present

The licensing service, which is a critical component of the licensing system, was removed when uninstalling previous Open eVision 1.1 versions. This made it impossible to use another installed Open eVision version such as 1.0 or 1.0.1.

This has been solved.

Prompt not displayed at the end of installation

Sometimes the command-line prompt was not displayed after running the License Manager from the command-line. This has been solved.

Crash when using the License Manager in the installation directory under Windows Vista

When executing the License Manager from the command line in the installation directory, a crash could occur. This has been solved.

ActiveX not completely uninstalled

There were sometimes leftovers from an ActiveX component installation. This has been solved.

ActiveX

Default arguments were needed

The default argument mechanism was malfunctioning for some methods. This required all arguments to be specified. This has been solved.

Basic Types

Argument for StopTiming

The Easy.StopTiming method default argument is now correctly handled.

Issues when copying EROI... and EImage... classes

There were sometimes issues when using the following methods on EROI... and EImage... classes:

- The copy constructor (C++)
- The assignment operator (C++)
- The CopyTo method (all API)

Under some circumstances, the actual image data was not copied.

This has been solved.

Documentation fixes

The reference documentation has been solved for:

- EOCR.LearnPatterns
- The EasyImage, EasyColor and EasyObject static classes.

EasyImage

Wrong EayImage::Threshold behavior with a relative threshold of 100%

A relative threshold of 100% is now correctly reported as an error by Open eVision and triggers an exception.

Renames to avoid Visual Basic keyword clashes

- EHarrisCornerDector.Scale has been renamed to EHarrisCornerDector.IntegrationScale
- ECannyEdgeDetector.Scale has been renamed to ECannyEdgeDetector.SmoothingSCale

Wrong EHarrisInterestPoints::Draw with default zoomY value

This has been solved.

EasyObject

Wrong layer computation for holes

The reported layer index was sometimes wrongly computed for *nested* holes. This has been solved.

EasyFind

Problem upon model loading in EasyFind

Sometimes an exception was thrown when loading a model with the PatternFinder.Load method. This has been solved.

Wrong license required

The PatternFinder.Load method sometimes requested an EasyGauge license. This has been solved.

Syntax error in Visual Basic 6 script

There were syntax errors in the generated Visual Basic code for the EasyFind library. This has been solved.

Memory leaks when loading an EasyFind model

This has been solved.

EasyGauge

Dot grid calibration issues

Under some circumstances (when image contains noisy areas), the dot grid calibration did fail. The dot grid cliabration behaviour has been greatly improved.

Heap corruption on object destruction in EasyGauge

When an array that contains E...Gauge objects gets destroyed, exceptions sometimes occurred. This has been solved.

Open eVision Studio and coded application give differents results using ECircleGauge

This has been solved.

Empty file when saving a gauge object

When saving an E...Gauge model under .NET, this resulting file was empty and locked by the process (i.e. did not close correctly). This has been solved.

Open eVision Studio: rectangle gauge plots do not appear immediately

This has been solved

EasyMatrixCode

Wrong MatrixCode result string

Under some circumstances, parts of a MatrixCode result string were garbled appended with bogus characters. This has been solved

Fixes and Improvements in Release 1.1.0.4276

Open eVision License Manager

Wrong error message during offline activation

Preparing a portable memory device for offline activation requires an Internet connection. If no connection is available, the application reports an error.

The error message is now more explicit.

Open eVision Studio / Open eVision Eval

Wrong results displayed in case of matrix code decoding failure (EasyMatrixCode)

When a matrix code cannot be read, an error message is displayed. However, the results of the last decoded matrix code are still shown.

This has been solved.

Returned coordinates of the sampled points are wrong (EasyGauge)

In case of a model fitting gauge, the returned CenterX and CenterY do not correspond to the sampled point specified by the index.

This has been solved.

EasyGauge

EFrameShape.Load and EFrameShape.Save are missing

This has been solved.

EasyOCR

EOCR.GetPatternBitmap returns the wrong type

The type of the returned value is BW8, whereas an EImageBW8 is expected.

This has been solved.

EasyMatrixCode

Wrong 0x00 character decoding

If the 0x00 character is in the first place, the matrix code decoding phase never ends, and the reading process always ends with a time out message.

This has been solved.

Known Issues

Basic Types

- 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.

EasyObject

ECodedImage2 and EHarrisDetector results draw slowly when there are many results.

EasyMatch

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

- 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.

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, add manually 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

- 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.