Release Notes mViz 4.5

The release features a new powerful tool called Snake. It lets you perform inspections along a path of arbitrary shape. Try the utility `mViz Snake` and play with the image `SnakeTest.bmp` and the model file `SnakeTest.sng`.

The libraries compiled with Visual Studio 2019 are now available.

Notice that this is the last release to come before the version 5.

mViz Snake

There is new utility application called `mViz Snake`. It lets you create, edit and test a SnakeGauge interactively. A SnakeGauge has the capability to inspect an edge of arbitrary shape, made of a sequence of line segments, circular arcs or splines. The deviation from the ideal shape can be assessed in different ways. The SnakeGauge can also inspect the width of some ribbon following this path, and detect anomalies.

After design, the layout of a snake can be saved to disk for reuse in your application. You can also create your own interactive editor.

General

The method `Image::Read` was causing memory leaks when reading `.png` files. This has been fixed.

The class `Graph` now supports the method `Bezier` to draw a cubic Bezier arc.

Image processing

The methods `Operator::DeBayer` for Bayer-pattern reconstruction, `Kernel::Convolve`, `Operator::Equalize` and `Geometry::Transform` did not support not working in-place (source = destination). This has been changed.

The method `Morpho::Distance` now supports 16 bits values, allowing to process large images (distances up to 32767 pixels). Check the new argument `Large`. When set to `true`, the result image is of the type `Gray16`.

A method `XY::Circumcenter` has been added. It computes the center of the circumscribed circle by three given points, and only that.

The method EdgeMap::Step edges takes a new argument ShowPolarity that enables an output mode compatible with the convention white=edge, black=non-edge.

The value of the noise amplitude in Histogram::Noise has been adjusted to correspond to the standard deviation in case of Gaussian noise.

An overload of the function Profile::EdgeDetect is now available. Instead of returning all detected edges, it returns the strongest.

The objects of the classes Profile, Path, Poly and Region now have Write and Read methods for file archival.

The class Poly now has the same methods as some already found in Path: Smooth, LineFit, CircleFit, Center, LineDeviation, CircleDeviation.

The method Region::Invert is now available to get the complement of a region with respect to a rectangle (image window).

Appending to a Profile without first initializing it (Profile::Append) was not possible. This has been changed.

The method Geometry::Transform to apply a precomputed geometric transformation was inaccurate in the interpolation mode. This has been fixed.

The filtering method Profile::BoxHighPass was performing a rescaling of the signal. This rescaling has been removed. The computed values will be impacted.

Blob Analysis

When computing the feature BlobConvexAreaRatio, the convex areas and the ratio were wrongly computed as negative values. This has been fixed. The computation of the feature BlobConvexArea alone did not have this problem.

If this a problem for you, please contact the technical support.

Gauging

CAUTION: we changed the BestSpan policy in the EdgePoint gauge to select the shortest rather than the longest span. This is more logical as edges with a long span are of little interest. This will change the behavior of existing code that uses this feature.

Bar Code Reading

The flag SymbologyIdentifier has been added to enable/disable the insertion of the FNC1-in-first-position marker "]C1".

Dot Code Reading

Due to a wrong internal transfer of information, the Code2DReader was not able to process the DotCode symbology. This has been fixed.

The Code2DReader was causing a memory leak. It has been fixed.

mVizNET

The following symbolic constants were missing and have been added (in the class Constant):

Constant::Auto

Constant::NotInitialized

Constant::MinShort, Constant::MaxShort

`Constant::MinInt`, `Constant::MaxInt`

`Constant::MinFloat`, `Constant::MaxFloat`

mViz+

The context menu of an image now provides a command `Copy Window to Clipboard`, which copied the content of the window (with annotations) rather than the bare image.

The name of the `ClippingMode` in the scripts was sometimes wrong. This has been fixed.

The `Regional Minima` in `Morphology` were not displaying, contrary to the `Maxima`. This has been fixed.

Release Notes mViz 4.4

This release features a nice improvement in the `EdgeLine`: you can now fit a line against an edge rather than over it, so that the fitted line wholly remains on a side of the edge. This is useful for gauging of edges with cracks.

The `Blobs` detector supports two new features for shape assessment.

Image processing

The class `Poly` now has a `ConvexHull` method. It works the same way as that in `Path`.

The comparison operators `Operator::Equal/Greater` Test were returning wrong results on RGB images (these functions are marginally useful).

Creating a `Region` with a circular `Quad` object then getting the region outline could result in an infinite loop because of an algorithmic flaw. This has been fixed.

The methods `Profile::Plot` and `Histogram::Plot` were swapping the red and blue components. This has been fixed. It only affected the display, while the stored values were correct.

The method `Histogram::Bin` has a new overload taking a `Component` argument, to retrieve the frequencies when the image type was `Rgb`.

The methods `Operator::EqualTest` and `Operator::Greater` test were not working correctly with color images. This has been fixed.

The method `Region::Follow` was not resetting the `Path` argument, and this has to be done externally. This behavior has been changed.

In some particular cases, the method `Region::QuadAppend` could cause a later call to `Fill` to hang. This has been fixed.

Two models have been added to the calibration tool: `Mirroring` and `Reflection`. These are the equivalents of `Rotation` and `Similarity`, but with inverted coordinate system. Check the enumeration `RegistrationModels`.

Blob Analysis

The features `FilledArea` and `FillingRatio` have been added. They allow to detect “hollow” blobs.

The arithmetic operations on `Regions` (`Unite`, `Intersect`, `Subtract`) could fail when the regions were obtained from blobs because of interference with the blob labels. This has been fixed.

Template matching

The `Locator` could hang on images containing little features. This has been fixed.

The ShapeMatcher was not correctly handling the recognition based on selected blobs. This has been fixed.

Gauging and Calibration

The EdgeLine gauge has a new property Fitting that specifies if the fitting must be made against the edge (one of two sides) or over it. Check the values of the enumeration FittingSide.

The Edge… gauges were ignoring profiles extending past the image edges. This behavior has been changed and partial profiles are now possible.

The parameters of the Similarity transform (Geometry::ModelFit) could be computed wrongly. This has been fixed.

Bar Code Reading

The detection of the Start/Stop delimiting patterns of the ITF symbols was too permissive (since mViz 4.3), resulting in many false detections in valid barcodes when this symbology was enabled. This has been fixed.

Dot Code Reading

QR codes: when a Kanji field is met, the reader will now generate a \000020 ECI, followed by the decoded data in the Shift JIS encoding. (Without such a delimiter, string interpretation by the application would have been impossible.)

QR codes: the reader was emitting the SymbolPosition, SymbolTotal and Parity in the returned string, in case on an (unbuffered) Structured Append. This has been suppressed, and the values are reported in data members instead.

OCR

The reader will now return a relative contrast value in addition to the matching score. This is useful for text quality assessment. Check the member AverageContrast (global) and method CharContrast (per character).

Switching between the NoLayout anf FlexibleLayout modes could cause malfunctions because of wrong internal updates. This has been fixed.

Reading a font after a call to DeleteFontChar could cause a freeze. This has been fixed.

mViz+

Scripts generated with scalar arguments returned by reference were wrongly inserted in the scripts. This has been fixed.

The integration of the ShapeMatcher was very poor. This has been improved.

Release Notes mViz 4.3

This release includes three important enhancements:

- The code reader now supplies the Quality Indicators for the QR codes;
- The locator has been enhanced to allow the detection of overlapping template instances.
- mViz is now available for the Vision Component Nano-Z smart camera, on request.

General

Some sections of the User's Manual were missing. This has been fixed.

Image processing

`Path::Draw` now allows to specify the size of the vertices rather than presence/absence, following the size convention of `Graph::Dot`. Caution: the argument `VertexSize` was previously a Boolean type.

All morphological filters now perform a straight image copy for a structuring element of zero size. This eases comparison with the original image.

The method `Region::Box` has been added to obtain window limits around a defined Region. This is useful to optimize the processing cost for filtering in a region.

Now instead of, for instance,

```
// Apply the Prewitt, using an image of the same size
Prewitt0.Set(Src);
Kernel::Canny(Src, Prewitt0, Canny2Gradient);

// Transfer to the source image, inside the wedge
Wedge.Fill(Prewitt0, Src);
```

you can use

```
// Get the region bounding box, with a 1 pixel margin
Limits L= Wedge.Box(1);

// Apply the Prewitt filter in the bounding box
Prewitt0.Set(Src);
Src.Window(L); Prewitt0.Window(L);
Kernel::Canny(Src, Prewitt0, Canny2Gradient);
Src.Window(); Prewitt0.Window();

// Transfer to the source image, inside the wedge
Wedge.Fill(Prewitt0, Src);
```

The comparison operators Operator::CompareEqual and .

Blob Analysis

The computation of the `Centroid` feature could cause a division by zero exception in case of a black blob. This has been fixed.

Template matching

A property `Locator::MaxOverlap` has been added to allow detection of touching or even overlapping template instances.

The method `Locator::EdgePoints` was returning wrong coordinates in the degrees angular mode. This has been fixed.

The `ScoreType` property is no more supported. The default value, `NGCScore`, is always used. If this is an issue, contact the Technical Support.

Peephole improvements have been made.

Dot Code Reading

The full set of Quality Indicators is now available for the QR Codes. (Do not forget to set the `CheckQuality` flag.)

The indicators are, by order,

- **Overall Grade**: this takes the worst of the grades below, giving a global idea of the printing quality.
- **Symbol Contrast**: indicates how different the foreground and background intensities are. The larger the better, as noise, dirt or other degradation can modify these intensities and complicate location/decoding.
- **Print Growth (Horizontal and Vertical)**: a measure of the relative thickness of the black/white cells, which can be adverse because of orverinkg/underinking or similar effects due to the marking process.
- **Axial Non Uniformity**: indicates if the size of the cells differs along one axis compared to the other. This is often an indication of a mismatch in marking speeds across and along the substrate.
- **Grid Non Uniformity**: indicates if the cells are aligned on a regular grid, with no local deformation. This is often an indication of a mechanical disturbance in the marking motion.
- **Unused Error**: the 2D codes have a built-in error decoding capability, enabled by redundancy in the content. The more error correction capacity is consumed, the more likely a reading failure can arise.
- **Modulation**: is a measure of the uniformity of the reflectance of the light and dark cells respectively. Uneven modulation can be caused by irregular ink supply.
- **Reflectance Margin**: is a measure of how well each cell is distinguishable as light or dark in comparison to the global threshold. Insufficient margin can be caused by irregular ink supply or lack of contrast.
- **Fixed Pattern Damage**: detects various anomalies in the fixed parts of the code (finder pattern). Damage in these areas can make the symbol undecodable by lack of proper geometric references. The causes can be diverse.

Two extra indicators are QR Code-specific:

- **Format Information Damage**: checks if the three Format fields (alongside the three finder patterns) are unaltered. Wrong format infomation makes the QR unreadable.
- **Version Information Damage**: checks if the two Version fields (near two of the finder patterns) are unaltered. Wrong version infomation makes the QR unreadable.

In some rare circumstances, Data Matrix codes with a complex geometry could cause a freeze. This has been fixed.

The accuracy of some Quality Indicators of the Data Matrix has been improved.

Bar Code Reading

The readability of the Interleaved 2 of 5 codes has been well improved.

Due to wrong coding, multiple barcodes side by side had a low reading rate. This has been fixed.

Release Notes mViz 4.2

This release fixes a number of minor issues. Powerful line and circle fitting functions have been added, as well as assessment of linearity or circularity. They operate on `Path` data. Some blur and edge detection filters have been added.

A new utility program, `mViz Code2D`, has been added to ease the setting of the dot code reader for difficult cases and optimize the parameters.

General

We recall that all angles, either user-input or computed by mViz are expressed in Degrees by default. You can change this behavior by setting

```
Status::AngleUnit= Status::FromRadians;
Status::UnitAngle= Status::ToRadians;
```

Both assignments are required.

The `Image` class now provides a method `Area` which returns the number of pixels in the current `Window` or whole image.

The `Status::License` function has been improved for easier assessment of plain/temporary license and presence/absence of a dongle. Note that for security reasons, the full documentation of this function is only delivered upon request to the Technical Support.

Image processing

Several new fast filters have been added to support Gaussian and Canny filtering. They appear as `Kernel::Binomial/5/7/9` lowpass filters, and as `Canny/1/15/2` for the `GradientType` of the `Kernel::Canny` and `EdgeMap::GradientVector/GradientMaxima/StrongEdges` methods.

A filter `Kernel::CannyPhase` has been added, supporting one the above `Canny GradientType`. It returns the orientation angle of the gradient vector as a value in range `[0..255]`, corresponding to the angular ranges `[0..360]` or `[0..180]` (polarity invariant).

Blob Analysis

The computation of features that require the gray values from the source image, namely `Mass`, `Centroid`, `Ellipsoid`, `Average`, `Deviation`, `Minimum`, `Maximum` and `Histogram` required the `Window` of the source image to be reset, for proper computation. This is no more required.

Code Reading

The Extended Code 39 barcodes could fail to be properly decoded in case they included a non-terminating asterisk * character. This has been fixed.

Gauging

The `Path` object now supports fitting of a line or circle model to the points. This can be made in a standard or robust way, i.e. when there are alien “garbage” points in the data. Check the methods `Path::FitLine` and `Path::FitCircle`. Notice that these advanced functions require the `Gauging` or `Calibration` license.

The methods `Path::LineDeviation` and `Path::CircleDeviation` (for linearity and circularity assessment) have been enhanced to support explicit specification of the reference line/circle position by the user. The latter can also be obtained automatically by the above `FitLine/FitCircle` methods. This change will impact the code using them.

The negative indexing (i.e. n^{th} from end) was not working correctly in the Decode method of the Edge gauges when a single edge polarity (`Raising` or `Falling`) was chosen. This has been fixed.

Template matching

The `Locator` object now exposes the method `EdgePoints`, which retrieves the outline of a given instance of the template and stores them in a `Path` object. This method is meant for display purposes.

Release Notes mViz 4.1

This is an intermediate release with miscellaneous improvements.

General

The `Limits` objects (rectangle/window limits) can now save their settings and retrieve them from a file. Check the methods `Read` and `Write`.

The `Quad` object now has a `SetCenter` method with less arguments, to create a circle with no worries on the arguments.

Blob Analysis

The `Blobs` context now has a method to pick the blob with the smallest or largest feature value. Check the method `Blobs::ByFeatureFirst`.

The `Centroid` and `Ellipsoid` features of a completely black blob (all pixel values equal zero) cannot be computed. This was causing an arithmetic condition. It is now handled correctly and the corresponding features are left as not-computed.

Image processing

The method `Histogram::ShowNoise` has been added to the image quality functions. It gives visual feedback about the intensity and distribution of noise in an image.

Methods `Path::LineDeviation` and `Path::CircleDeviation` have been added to assess straightness (smoothness) of straight and curves edges fond in a path.

New Gradient types have been added. They correspond to Canny filters of different strengths. Check the `GradientTypes` enumeration.

The `Morpho::Watershed` image segmentation method now accepts images of type vector gradient (rather than grayscale gradient), such as those computed by the class `EdgeMap`.

Calibration

The fitting of a `Scaling` model could result in wrong calibration parameters. This has been fixed.

Code Reading

The 2D code reader now supports an “inking correction” feature. When the cells are too thin or too fat, morphological processing can be applied internally to improve the decoding rate. Check the property `InkingCorrection`. It should be clear that this feature is also usable with direct part marking.

The detection of the QR codes has been enhanced. Some cases such that one of the finder pattern (corner) is damaged can now be decoded.

The Grid grading Quality Indicators were not properly computed in mViz 4.0. This has been fixed.

Gauging

The `EdgeRectangle` widget has a new working mode to measure thicknesses. The new property `FourSides` is true by default, which fits a rectangle. But when set to false, it will only fit two parallel sides, so that their distance can be obtained.

The EdgeArc object was not saving all required position properties with the methods `Read`/`Write`.

The functions `EdgeRectangle::HitHandle` and `EdgeRectangle::Drag` were not handling correctly the `LengthHandle` (sizing of the space aroung the rectangle edges). This has been fixed.

mVizNET

The Value objects `Site` and XY were missing many methods available in the native libraries. They have been restored.

mViz+

Some floating-point parameters such as `CharReader::GapFraction` or `Geometry::CalibrateTarget::OriginRow/Column` could not be set because of bad behavior of the widget. This has been fixed.

The method `Geometry::Undistort` following `Geometry::CalibrateTarget` was not operating. This has been fixed.

The method `Blobs::ByFeatureFirst` has been integrated.

The copy of a constant to a profile via `Process > Point to Point > Profile Arithmetic…` was not possible. This has been fixed.

Some operations on profiles could cause a fatal error because of wrong memory management. This has been fixed.

Note that problems that appear in mViz+ do not necessarily reflect problems in the library mViz.

Release Notes mViz 4.0

The major features of this version of mViz is that the installer becomes 64 bits by default. And more Microsoft compilers are supported: Visual Studio 2008, 2010, 2012, 2013, 2015, and 2017.

The .NET wrapper is now compiled for the framework versions 3.5, 4.0 and 4.6.

The image reader/writer libraries have been upgraded to the most current versions: Png 1.6.37, Jpeg 9c and Tiff 4.0.10.

General

The settings of several mViz objects can now be saved to/retrieved from a disk file (all code readers, all measurement gauges).

The following file extensions are now accepted as aliases when saving/loading images: `jpg`/`jpeg`, `tif`/`tiff`.

The working range of the `Image::HitHandle` function was too large on a zoomed image. This has been fixed.

Blob Analysis

The `Gray1` image type (binary) is now supported for segmentation. Use the overload

```
int Blobs::Segment(const Image& Src, bool Above, bool Connexity8, int MinimumArea, int MaximumArea)
```

The method `Blobs::Segment` working with adaptive thresholding (`int Size` argument) and a mask was not using the mask and applied to the whole image. This has been fixed.

```
void Blobs::Segment(int Size, int Noise, const Region& Mask, const Image& Src, bool Above,
bool Connexity8, int MinimumArea, int MaximumArea);
```

Due to a typo, sorting decreasingly on an integer feature was malfunctioning. This has been fixed.

Image processing

Several statistical functions were limited to an image size of 2^{23}= 8 388 608 pixels. This limit has been increased to 2^{31}= 2 147 483 648. The row width is limited to 2^{15}-1=32767 pixels.

The method `Histogram::Normalize` has been added. It allows to transform an image by a gain/offset transformation so that its gray mean and standard deviation take specified values. This is useful to deal with images of wildly varying intensity or contrast.

Code Reading

The barcode and 2D reader objects (Code1D/Code2D Reader) can now save their settings and retrieve them from a file.

The `Code2DReader` was not appending a null byte at the end of the decoded string when `SymbologyIdentitifer` was activated (though the string length was correct). This has been changed.

Character Reading

The flag `VariableWidth` was not saved correctly to the font files, and was always treated as `true` upon loading. This has been fixed.

The segmentation results of the `CharsSegment` and `CharsRead` methods were slightly different in the `Dotted` modes. This has been fixed.

Gauging

All gauge objects (Edge Point/Line/Arc/Rectangle) can now save their settings and retrieve them from a file. Check the methods `Read` and `Write`.

Classification

To avoid name clashes, the data access members `Classifier::Bool/Gray/Gray16/Rgb/Int/-Float/Dble` had to be renamed with a suffix `Value` and are now `Classifier::BoolValue/-GrayValue/Gray16Value/RgbValue/IntValue/FloatValue/DbleValue`.

mVizNET

The `Buffer` property of an image was not exposed. This has been changed.

Conversions between .NET and mViz Images, and conversely, are now available for all supported types. Note that .NET supports the 16 bits grayscale images very poorly, unlike mViz.

The ref class objects now have a destructor that deallocates unmanaged memory. Use them to force memory deallocation and avoid heap overflow when you create/destroy such objects frequently.

mViz+

The Filter parameter of `Geometry::DownSample` was set to `false` by default. This was not the intended behavior, it has been changed.

The operations reading from/writing to a file, now available for several objects, are now accessed via their respective `Storage…` menu entries.

mViz OCR

Many new font files are available. Check the `Images\Fonts` folder.