AVT Image Transform

Programmer's Manual

V1.3
2014-08-12

Allied Vision Technologies GmbH
Taschenweg 2a
D-07646 Stadtroda / Germany

ALLIED Vision Technologies

Legal Notice

Trademarks

Unless stated otherwise, all trademarks appearing in this document of Allied Vision Technologies are brands protected by law.

Warranty

The information provided by Allied Vision Technologies is supplied without any guarantees or warranty whatsoever, be it specific or implicit. Also excluded are all implicit warranties concerning the negotiability, the suitability for specific applications or the non-breaking of laws and patents. Even if we assume that the information supplied to us is accurate, errors and inaccuracy may still occur.

Copyright

All texts, pictures and graphics are protected by copyright and other laws protecting intellectual property. It is not permitted to copy or modify them for trade use or transfer, nor may they be used on websites.

Allied Vision Technologies GmbH 08/2014

Managing Director: Mr. Frank Grube
Tax ID: DE 184383113

Headquarters:

Taschenweg 2a
D-07646 Stadtroda, Germany
Tel.: +49 (0)36428 6770
Fax: +49 (0)36428 677-28
e-mail: info@alliedvisiontec.com

Contents

1 Contacting Allied Vision Technologies

Note

- **Technical Information**
  http://www.alliedvisiontec.com
- **Support**
  support@alliedvisiontec.com

Allied Vision Technologies GmbH (Headquarters)
Taschenweg 2a
07646 Stadtroda, Germany
Tel.: +49 36428-677-0
Fax.: +49 36428-677-28
Email: info@alliedvisiontec.com

Allied Vision Technologies Canada Inc.
101-3750 North Fraser Way
Burnaby, BC, V5J 5E9, Canada
Tel: +1 604-875-8855
Fax: +1 604-875-8856
Email: info@alliedvisiontec.com

Allied Vision Technologies Inc.
38 Washington Street
Newburyport, MA 01950, USA
Toll Free number +1 877-USA-1394
Tel.: +1 978-225-2030
Fax: +1 978-225-2029
Email: info@alliedvisiontec.com

Allied Vision Technologies Asia Pte. Ltd.
82 Playfair Road
#07-02 D'Lithium
Singapore 368001
Tel. +65 6634-9027
Fax:+65 6634-9029
Email: info@alliedvisiontec.com

Allied Vision Technologies (Shanghai) Co., Ltd.
2-2109 Hongwell International Plaza
1602# ZhongShanXi Road
Shanghai 200235, China
Tel: +86 (21) 64861133
Fax: +86 (21) 54233670
Email: info@alliedvisiontec.com

2 Introduction

2.1 Document history

Version	Date	Changes
1.0	2013-03-22	Initial version
1.2	2013-06-18	Small corrections, layout changes
1.3	2014-08-12	Rework of the whole document

2.2 Conventions used in this manual

To give this manual an easily understood layout and to emphasize important information, the following typographical styles and symbols are used:

2.2.1 Styles

Style	Function	Example
Bold	Programs, inputs or highlighting important things	**bold**
Courier	Code listings etc.	`Input`
Upper case	Constants	CONSTANT
Italics	Modes, fields, features	*Mode*
Blue and/or parentheses	Links	(Link)

2.2.2 Symbols

Note

This symbol highlights important information.

Caution

This symbol highlights important instructions. You have to follow these instructions to avoid malfunctions.

www

This symbol highlights URLs for further information. The URL itself is shown in blue.
Example: http://www.alliedvisiontec.com

3 General aspects of the Library

The purpose of the AVT Image Transform library is to transform images received via Vimba APIs into common image formats. It is part of Vimba and doesn't require a separate installation.

Applications using Vimba C or C++ APIs may need to integrate AVT Image Transform library as an additional library, whereas a subset of it is seamlessly integrated into the AVT Vimba .NET API via method `Frame.Fill()`. (See the Vimba.NET Programmer's Manual)

3.1 Technologies

The AVT Image Transform Library is available in two versions: The standard version executes a method call in a single thread. The OpenMP version is recommended for achieving a high performance with multi-core PCs, since the function calls are distributed over the cores of the processor. Although additional time is required to schedule the sub tasks, its performance roughly scales by the number of free cores.

Both libraries have the same interface and the same name, but are delivered in different folders in the Vimba installation, allowing you to use the library you want.

The OpenMP version of the library can be found in the subdirectory `OpenMP` beneath the standard library.

3.2 Supported image data formats

In contrast to image *file* formats, image *data* formats contain no embedded meta-data, but only pure images as rectangular arrays of pixels. These pixels have certain characteristics, e.g. a certain color or bit depth and the endianness for a bit depth of more than eight, and they may appear in a certain order. All this is described in a pixel format name.

Note

All implemented image transformations only accept multi-byte pixel formats in little-endian byte order.

For this library, pixel formats from the interface specifications IIDC and GigE Vision as well as common display formats on Windows and Linux platforms have been selected. The names of the pixel formats mainly match the list of values of the "PixelFormat" feature in the Standard Features Naming Convention of GenICam.

Since the following pixel formats span a few bytes or in some cases several lines, their width and height or size (in pixels) must be divisible by the following values:

Pixel format	Width	Height	Size
`Mono12Packed`	1	1	2
`Bayer....`	2	2	1
`Yuv411`	4	1	1
`Yuv422`	2	1	1

3.3 Supported transformations

Source \ Target	Mono8	Mono16	Rgb8	Bgr8	Argb8	Bgra8	Yuv422
Mono8	✔	-	✔	✔	✔	✔	✔
Mono10	✔	✔	✔	✔	✔	✔	-
Mono12	✔	✔	✔	✔	✔	✔	-
Mono12Packed	✔	-	✔	✔	✔	✔	-
Mono14	✔	✔	✔	✔	✔	✔	-
Mono16	✔	✔	✔	✔	✔	✔	-
BayerXX[1]8	✔	✔	✔	✔	✔	✔	✔
BayerXX[1]10	✔	✔	✔	✔	✔	✔	-
BayerXX[1]12	✔	✔	✔	✔	✔	✔	-
BayerXX[1]12Packed	✔	✔	✔	✔	✔	✔	-
BayerXX[1]16	✔	✔	✔	✔	✔	✔	-
Rgb8	✔	-	✔	✔	✔	✔	-
Bgr8	✔	-	✔	✔	✔	✔	-
Argb8	✔	-	✔	✔	✔	✔	-
Bgra8	✔	-	✔	✔	✔	✔	-
Yuv411	✔	-	✔	✔	✔	✔	-
Yuv422	✔	-	✔	✔	✔	✔	✔
Yuv444	✔	-	✔	✔	✔	✔	-

[1] BayerXX is any of BayerGR, BayerRG, BayerGB or BayerBG

4 API usage

4.1 General

Every concrete image transformation must specify its three parameters:

- The fully-qualified input image (including complete format specification)
- The complete output format (plus a pointer to enough memory to hold the transformed image)
- The transformation itself (including transformation options if there is more than one possible way from the input format to the output format)

`VmbImageTransform()` is the main function, which covers all three transformation parameters. It uses two `VmbImage` pointers to specify the source and destination image and a list of `VmbTransformInfo` structs to specify the transformation.

To ease filling the structs that are needed for a successful call of `VmbImageTransform()`, several helper functions for initializing them are available:

- Methods for filling the necessary format information of either the input or the output image (described in chapter Filling Image Information).
- Methods for filling optional transformation parameters to invoke special and additional functionality, as can be seen in chapter Specifying Transformation Options.

4.2 Filling Image Information

There are three ways to fill `VmbImage` structs:

1. Filling the fields explicitly one by one (not recommended)
2. Using the function `VmbSetImageInfoFromPixelFormat()` by providing a `VmbPixelFormat_t` value and the size
3. Using the function `VmbSetImageInfoFromString()` by providing a string that describes the pixel format and the size

With the first approach, you have to specify many fields (that are important for the transformation process, but very possibly not for you), while the latter two approaches only require knowing either the name or the enumeration value of the image format.

4.3 Specifying Transformation Options

Depending on your application and the desired image characteristics, you can choose between several transformation options. A common way to transform color images is using a 3x3 matrix with method `VmbSetColorCorrectionMatrix3x3()`. The Matrix you have to specify is a 3x3 row order float matrix.

$$\begin{pmatrix} rr & rg & rb \\ gr & gg & gb \\ br & bg & bb \end{pmatrix}$$

Every transformation of a Bayer image requires defining how to interpolate or combine the color pixels. This can done with the help of an additional transformation parameter to `VmbImageTransform()`, which can be filled with the help of function `VmbSetDebayerMode()` (using a `VmbDebayerMode` as input). `VmbDebayerMode` can be one of the following values:

Value	Description
`VmbDebayerMode2x2`	2x2 with green averaging
`VmbDebayerMode3x3`	3x3 with equal green weighting per line
`VmbDebayerModeLCAA`	Debayering with horizontal local color anti-aliasing
`VmbDebayerModeLCAAV`	Debayering with horizontal and vertical local color anti-aliasing
`VmbDebayerModeYUV422`	Debayering with YUV422-alike sub-sampling

4.4 Transforming images

When you've prepared all the parameters of a transformation, use the pre-filled structs and call `VmbImageTransform()`.

5 Transformation examples

5.1 Debayering into an Rgb8 image

Example for a transformation of a BayerGR8 image with 640x480 pixels into an Rgb8 image of the same size:

```
VmbImage                sourceImage;
VmbImage                destinationImage;
VmbTransformInfo        info;

// set size member for verification inside API
sourceImage.Size        = sizeof ( sourceImage );
destinationImage.Size   = sizeof ( destinationImage );

// attach the data buffers
sourceImage.Data        = pInBuffer;
destinationImage.Data   = pOutBuffer;

// fill image info from pixel format
VmbSetImageInfoFromPixelFormat ( VmbPixelFormatBayerGR8,
                                 640,
                                 480,
                                 &sourceImage);

// fill image info from pixel format string
std::string name ( "RGB8" );
VmbSetImageInfoFromString ( name.c_str(),
                            static_cast<VmbUint32_t>( name.size() ) ,
                            640,
                            480,
                            &destinationImage );

// set the debayering algorithm to simple 2 by 2
VmbSetDebayerMode ( VmbDebayerMode2x2, &info );

// perform the transformation
VmbImageTransform ( &sourceImage, &destinationImage, &info, 1 );
```

5.2 Debayering into a Mono8 image

Example for preparing a transformation of a BayerRG8 image with 640x480 pixels into a Mono8 image of the same size:

```
VmbImage                sourceImage;
VmbImage                destinationImage;
VmbTransformInfo        info;

// set size member for verification inside API
sourceImage.Size        = sizeof ( sourceImage );
destinationImage.Size   = sizeof ( destinationImage );

// attach the data buffers
sourceImage.Data        = pInBuffer;
```

```
destinationImage.Data   = pOutBuffer;

// fill image info from pixel format string
std::string name ( "PixelFormatBayerRG8" );
VmbSetImageInfoFromString ( name.c_str(),
                            static_cast<VmbUint32_t>( name.size() ),
                            640,
                            480,
                            &sourceImage );

// fill image info from pixel format
VmbSetImageInfoFromPixelFormat ( VmbPixelFormatMono8,
                                 640,
                                 480,
                                 &destinationImage );

// set the debayering algorithm to 3 by 3
VmbSetDebayerMode ( VmbDebayerMode3x3, &info );

// perform the transformation
VmbImageTransform ( &sourceImage, &destinationImage, &info, 1 );
```

5.3 Applying a color correction to an Rgb8 image

Example for applying a color correction to an Rgb8 image with 640x480 pixels:

```
VmbImage           sourceImage;
VmbImage           destinationImage;
VmbTransformInfo   info;
const VmbFloat_t   mat[] = { 1.0f, 0.0f, 0.0f,
                             0.0f, 1.0f, 0.0f,
                             0.0f, 0.0f, 1.0f };

// set size member for verification inside API
sourceImage.Size        = sizeof ( sourceImage );
destinationImage.Size   = sizeof ( destinationImage );

// attach the data buffers
sourceImage.Data        = pInBuffer;
destinationImage.Data   = pOutBuffer;

// fill image info from pixel format
VmbSetImageInfoFromPixelFormat ( VmbPixelFormatRgb8,
                                 640,
                                 480,
                                 &sourceImage );

// fill image info from pixel format
VmbSetImageInfoFromPixelFormat ( VmbPixelFormatRgb8,
                                 640,
                                 480,
                                 &destinationImage );

// set the transformation matrix
```

```
VmbSetColorCorrectionMatrix3x3 ( mat, &info );

// perform the transformation
VmbImageTransform ( &sourceImage, &destinationImage, &info, 1 );
```

5.4 Debayering a 12-bit image into a Mono16 image with additional color correction

Example for preparing a transformation of a BayerGR12 image with 640x480 pixels into a Mono16 image of the same size:

```
VmbImage           sourceImage;
VmbImage           destinationImage;
VmbTransformInfo   info[2];
const VmbFloat_t   mat[] = { 1.0f, 0.0f, 0.0f,
                             0.0f, 1.0f, 0.0f,
                             0.0f, 0.0f, 1.0f };

// set size member for verification inside API
sourceImage.Size        = sizeof ( sourceImage );
destinationImage.Size   = sizeof ( destinationImage );

// attach the data buffers
sourceImage.Data        = pInBuffer;
destinationImage.Data   = pOutBuffer;


// fill image info from pixel format
VmbSetImageInfoFromPixelFormat ( VmbPixelFormatBayerGR12Packed,
                                 640,
                                 480,
                                 &sourceImage );

// fill image info from pixel format string
std::string name ( "MONO16" );
VmbSetImageInfoFromString ( name.c_str(),
                            static_cast<VmbUint32_t>( name.size() ) ,
                            640,
                            480,
                            &destinationImage );

// set the debayering algorithm to 2 by 2
VmbSetDebayerMode ( VmbDebayerMode2x2, &info[0] );

// set the transformation matrix
VmbSetColorCorrectionMatrix3x3 ( mat, &info[1] );

// perform the transformation
VmbImageTransform ( &sourceImage, &destinationImage, &info, 2 );
```

6 Function reference

6.1 Information

6.1.1 VmbGetVersion()

Inquire the library version.

	Type	Name	Description
out	`VmbUint32Ptr_t`	Value	Contains the library version (Major,Minor,Sub,Build)

6.1.2 VmbGetErrorInfo()

Translate Vimba error codes into a human-readable string.

	Type	Name	Description
in	`VmbError_t`	ErrorCode	The error code to get a readable string for
out	`VmbANSIChar_t*`	InfoString	Pointer to a zero terminated string that will contain the error information on return
in	`VmbUint32_t`	MaxInfoLength	The length of the InfoString buffer

6.1.3 VmbGetApiInfoString()

Get information about the currently loaded Vimba ImageTransform API.

	Type	Name	Description
in	`VmbAPIInfo_t`	InfoType	Type of information to return
out	`VmbANSIChar_t*`	Info	Pointer to a zero terminated string that will contain the information on return
in	`VmbUint32_t`	MaxInfoLength	The length of the Info buffer

InfoType may be one of the following values:

Value	Description
`VmbAPIInfoAll`	Returns all information about the API
`VmbAPIInfoPlatform`	Returns information about the platform the API was built for (x86 or x64)
`VmbAPIInfoBuild`	Returns info about the API built (debug or release).
`VmbApiInfoTechnology`	Returns info about the supported technologies the API was built for (OpenMP or OpenCL).

6.2 Transformation

6.3 VmbImageTransform()

Images are transformed using the `VmbImageTransform()` function. The transformation is defined by the images used and the desired transformation. If a transformation is not supported, `VmbErrorBadParameter` is returned.

	Type	Name	Description
in	`VmbImage*`	source	Image to transform
out	`VmbImage*`	destination	Destination image
in	`VmbTransformInfo*`	parameter	Optional transform parameters
in	`VmbUint32_t`	parameterCount	Number of transform parameters

6.4 Helper functions

6.4.1 VmbSetImageInfoFromPixelFormat()

Set image info member values in `VmbImage` from pixel format or string.

	Type	Name	Description
in	`VmbPixelFormat_t`	PixelFormat	PixelFormat describes the pixel format used by the image data member.
in	`VmbUint32_t`	width	Width of the image in pixels
in	`VmbUint32_t`	height	Height of the image in pixels
in/out	`VmbImage*`	image	Pointer to Vimba image to set the info to

Note

`VmbPixelFormat_t` can be obtained from Vimba C/C++ APIs frame or from the *PixelFormat* feature. For displaying images, it is suggested to use `VmbSetImageInfoFromString()` or to look up a matching `VmbPixelFormat`

6.4.2 VmbSetImageInfoFromString()

Set image info member values in `VmbImage` from pixel format or string.

	Type	Name	Description
in	`VmbANSIChar_t*`	imageFormat	Image format as a (const) case insensitive string that is either a PixelFormat (Vmb is optional) or a pixel struct name
in	`VmbUint32_t`	stringLength	The length of the pixel format string
in	`VmbUint32_t`	width	Width of the image in pixels
in	`VmbUint32_t`	height	Height of the image in pixels
in/out	`VmbImage*`	image	Pointer to Vimba image to set the info to

6.4.3 VmbSetDebayerMode()

Set transformation options to a predefined debayering mode.

	Type	Name	Description
in	`VmbDebayerMode_t`	debayerMode	The mode used for debayering the source raw image, default mode is 2x2 debayering. Debayering modes only work for image widths and heights divisible by two.
in/out	`VmbTransformInfo_t*`	transformInfo	Parameter that contains information about special transform functionality.

Note

Debayering is only applicable to image formats with both an even width and an even height.

6.4.4 VmbSetColorCorrectionMatrix3x3()

Set transformation options to a 3x3 color matrix transformation.

	Type	Name	Description
in	`VmbFloat_t*`	matrix	Color correction matrix
in/out	`VmbTransformInfo_t*`	transformInfo	Parameter that is filled with information about special transform functionality.

7 Structs

7.1 VmbImage

`VmbImage` encapsulates image data for the transformation function.

Struct entry	Purpose
`VmbUint32_t size`	Size of the structure
`void* data`	Pointer to the payload received from AVT Vimba C/C++ API or to the display image data
`VmbImageInfo imageInfo`	Internal information data used for mapping the images to the correct transformation, imageInfo data can be set with `VmbSetImageInfo()` helper functions from `VmbPixelFormat_t` or format string

7.2 VmbImageInfo

`VmbImageInfo` contains image information needed for the transformation function.

Struct entry	Purpose
`VmbUint32_t width`	The width of the image in pixels. For macro pixel formats like YUV, it is the width in sub pixels
`VmbUint32_t height`	The height of the image in pixels
`VmbInt32_t stride`	The offset from the current line to the next line, a value not equal to width is currently not supported
`VmbPixelInfo pixelInfo`	Information about the pixel format

7.3 VmbPixelInfo

`VmbPixelInfo` describes the pixel format of an image.

Struct entry	Purpose
`VmbUint32_t bitsPerPixel`	Number of bits for one image pixel, or sub pixel in macro pixel formats
`VmbUint32_t bitsUsed`	Number of bits used per pixel, e.g. RGB12 has 48 bits per pixel and 36 bits used
`VmbAlignment_t alignment`	For image formats where bitsPerPixel is not equal to bitsUsed, the alignment specifies the bit layout of the pixel
`VmbEndianness_t endianness`	Specifies the endianness of pixels that are larger than one byte
`VmbPixelLayout_t pixelLayout`	Describes the layout of the pixel component, e.g., RGB or BGR layout
`VmbBayerPattern_t bayerPattern`	For raw image data, this field specifies the color filter array layout of the sensor

7.4 VmbTransformInfo

Transformation parameters given as a pointer to a `VmbTransformInfo` struct are used to invoke special and additional functionality, which is applied while interpreting the source image (debayering) or transforming the images (color correction).

Struct entry	Purpose
`VmbTransformType_t transformType`	Transform info type, may be VmbTransformTypeDebayerMode or VmbTransformTypeColorCorrectionMatrix
`VmbTransformParameter parameter`	Transform info variant, may be VmbTransformParameterDebayer or VmbTransformParameterMatrix3x3