Teledyne DALSA • 880 Rue McCaffrey • St-Laurent, Québec, H4T 2C7 • Canada
https://www.teledynedalsa.com/imaging/products/software/

SAP-AN0009: Using Sapera LT with OpenCV Processing API

Sapera LT Image Acquisition into OpenCV Processing Buffers

Overview

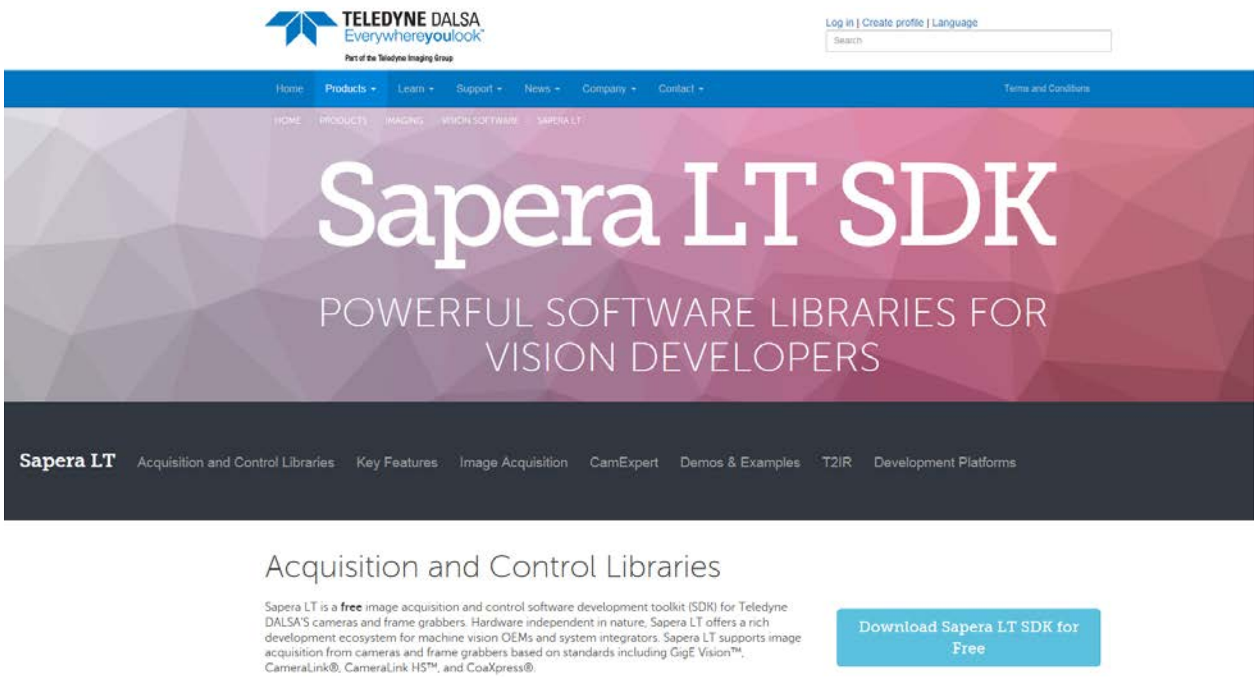
This application note describes how to export image data from a SapBuffer object for use in a third-party image processing API such as OpenCV. This document is only intended to give a general idea as to how the SapBuffer object can be used to create a third-party compatible image.

This document assumes the user is familiar with the Sapera LT SapBuffer class and the Mat class provided by the OpenCV library.

The examples provided within this document are demonstrated in C++ using the Sapera LT ++ API and OpenCV (Open Source Computer Vision Library) as an example third-party image processing API. These concepts are also compatible with C# and the Sapera LT .NET API. Usage of OpenCV and other third-party libraries and their algorithms is outside the scope of this document.

ℹ	**Note:** This document is for illustrative purposes only. OpenCV is not a product of Teledyne DALSA and thus no support for OpenCV shall be given. Support for OpenCV may be obtained through https://opencv.org.

For more information on processing images within callbacks or using multiple threads, contact Teledyne DALSA.

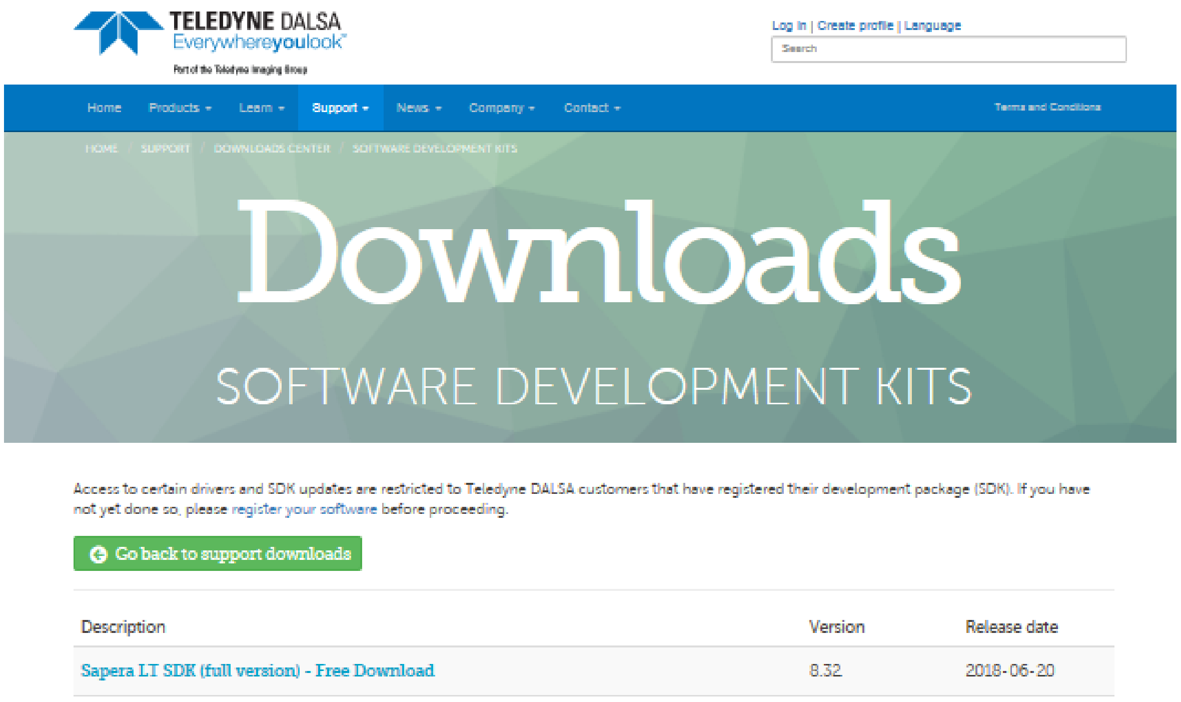


Acquisition and Control Libraries

Sapera LT is a **free** image acquisition and control software development toolkit (SDK) for Teledyne DALSA'S cameras and frame grabbers. Hardware independent in nature, Sapera LT offers a rich development ecosystem for machine vision OEMs and system integrators. Sapera LT supports image acquisition from cameras and frame grabbers based on standards including GigE Vision™, CameraLink®, CameraLink HS™, and CoaXpress®.

Sapera LT SDK (full version), the image acquisition and control SDK for Teledyne DALSA cameras and frame grabbers is available for download from the Teledyne DALSA website:

http://teledynedalsa.com/imaging/support/downloads/sdks/


TELEDYNE DALSA
Everywhereyoulook
Part of the Teledyne Imaging Group
Log in | Create profile | Language
Search
Home
Products
Learn
Support
News
Company
Contact
Terms and Conditions
HOME / SUPPORT / DOWNLOADS CENTER / SOFTWARE DEVELOPMENT KITS
Downloads
SOFTWARE DEVELOPMENT KITS
Go back to support downloads

Access to certain drivers and SDK updates are restricted to Teledyne DALSA customers that have registered their development package (SDK). If you have not yet done so, please register your software before proceeding.

Description	Version	Release date
Sapera LT SDK (full version) - Free Download	8.32	2018-06-20

Note: Administrative privileges are required to perform the Sapera LT software installation described in this application note.

Installation Prerequisites

The following Teledyne DALSA software is required:

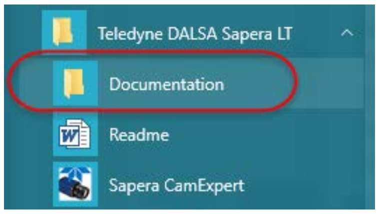
- Sapera LT SDK (full version 8.32.00.1847 or higher)
- For applications that use Teledyne DALSA frame grabbers, the device driver must be installed.

Sapera LT SDK and all device drivers are available for free download from the Teledyne DALSA website:

https://www.teledynedalsa.com/en/support/downloads-center/software-development-kits/

https://www.teledynedalsa.com/en/support/downloads-center/device-drivers/

Key documentation provided with the installation of the Sapera LT SDK includes the **Getting Started Manual For Frame Grabbers** or **Getting Start Manual for GigE Vision Cameras**.



For detailed information on Sapera LT buffers please see the “Working with Buffers” section of the Sapera LT User’s Manual and the SapBuffer class documentation in the Sapera LT ++ Programmer’s Manual (or Sapera LT .NET Programmer’s Manual for C# developers).

OpenCV Version

OpenCV 2.4.13 (or higher)

Example Program

An example program based on the Sapera Console Grab Example (GrabCPP) provided with the standard Sapera LT SDK installation can be downloaded from the link below. ftp://ftp.dalsa.com/Private/p_ProductSupport/TechSupportWiki/GrabConsole_OpenCV.zip

This example exports SapBuffer object data to an OpenCV Mat object and displays the Mat object image in a separate window. The example code for exporting SapBuffer data to Mat objects can be found in the *ExportToOpenCV_Copy()* and *ExportToOpenCV_Direct()* functions in the GrabCPP.cpp file.

ℹ	This example has been developed in C++ using Visual Studio 2012. If using a different version of Visual Studio update the header file paths and linker paths to include the paths to the OpenCV headers and libraries.

SapBuffer Functions and Properties

The following table shows the SapBuffer functions and properties you should be familiar with to export an SapBuffer object to a third-party image object.

SapBuffer Functions and Properties		
C++	C#	Description
GetWidth()	Width	Gets the image width in pixels (# of columns).
GetHeight()	Height	Gets the image height in pixels (# of rows).
GetFormat()	Format	Data format of the image (Mono8/10/12, RGB888, etc.) See the SapBuffer constructor documentation for a complete list of formats.
GetPixelDepth()	PixelDepth	Gets the number of significant bits .
GetBytesPerPixel()	BytesPerPixel	Gets the number of bytes required to store a single pixel.
Read()	Read()	Allows reading pixel data from the SapBuffer into an external data array.
GetAddress()	GetAddress()	Initiates direct buffer access to the raw pixel data. ReleaseAddress() may need to be called when direct access is no longer needed.
ReleaseAddress()	ReleaseAddress()	Ends direct buffer access obtained from GetAddress().
ColorConvert()	ColorConvert()	Converts a color image (for example, Bayer format) to RGB format.

These functions and parameters are fundamental to being able to export the SapBuffer to a third-party image object. All functions in the table above may not be required to perform the export. The *ColorConvert()* function is mentioned here for SapBuffer objects containing Bayer encoded images. If your image needs to be converted from Bayer to RGB format prior to exporting you can create a RGB SapBuffer and use the *ColorConvert()* function to perform the Bayer to RGB conversion.

SapBuffer Export

The SapBuffer objects can be exported to third-party utilities using one of two methods:

- using the raw memory directly of the SapBuffer object
- using a copy of the SapBuffer object data

Both methods are demonstrated below using the OpenCV Mat class. The Mat class is used to represent an image in the OpenCV library and can be used with many OpenCV classes and functions to process the image.

Creating a Mat Object Using SapBuffer Memory Directly

The following source code illustrates creating an OpenCV Mat object that uses the SapBuffer raw memory directly and displaying the Mat object image.

To obtain the SapBuffer memory address that contains the image data, use the SapBuffer::GetAddress function. Then create an instance of the Mat object (exportImg) using the constructor that accepts the dimensions of the image (width and height), image type, and a pointer to the existing image data. The Mat object is now ready for image processing.

Note that any changes made to the image data using the Mat object will also be reflected in the SapBuffer data since they are both sharing the same image data.

```
void ExportToOpenCV_Direct( SapBuffer *pSapBuf )
{
        if( pSapBuf == NULL )
                return;

        SapFormat sapFormat = pSapBuf->GetFormat();
        int       OpenCV_Type = 0;

        switch( sapFormat )
                {
                case SapFormatMono8:
                        OpenCV_Type = CV_8UC1;
                        break;
                case SapFormatMono16:
                        OpenCV_Type = CV_16UC1;
                        break;
                case SapFormatRGB888:
                        OpenCV_Type = CV_8UC3;
                        break;
                case SapFormatRGB161616:
                        OpenCV_Type = CV_16UC3;
                        break;
                default:
                        sapFormat = SapFormatUnknown;
                        break;
                }

        if( sapFormat != SapFormatUnknown )
        {
                // Export to OpenCV Mat object using SapBuffer data directly
                void    *pBuf = NULL;
                pSapBuf->GetAddress( &pBuf );
                Mat exportImg(  pSapBuf->GetHeight(), pSapBuf->GetWidth(), OpenCV_Type, pBuf );

                namedWindow( OPENCV_WINDOW_NAME, WINDOW_NORMAL | CV_WINDOW_KEEPRATIO );

                // Display OpenCV Image
                imshow( OPENCV_WINDOW_NAME, exportImg );
                pSapBuf->ReleaseAddress( &pBuf );

                waitKey(1);
        }
}
```

Creating a Mat Object Using a Copy of SapBuffer Memory

The following source code illustrates creating an OpenCV Mat object that makes a copy of the SapBuffer memory and displaying the Mat object image.

To create a Mat object with a copy of the data contained in the SapBuffer, use the Mat::create function. This function sets the image dimensions (width and height) and image type and allocates the appropriate amount of memory for the image data. The SapBuffer::Read function reads the data out of the SapBuffer and into the Mat object's data buffer (exportImg.data). The Mat object (exportImg) now has its own independent copy of the image data that is ready for image processing.

```
void ExportToOpenCV_Copy( SapBuffer *pSapBuf )
{
        if( pSapBuf == NULL )
                return;

        SapFormat sapFormat = pSapBuf->GetFormat();
        int      OpenCV_Type = 0;

        switch( sapFormat )
                {
                case SapFormatMono8:
                        OpenCV_Type = CV_8UC1;
                        break;
                case SapFormatMono16:
                        OpenCV_Type = CV_16UC1;
                        break;
                case SapFormatRGB888:
                        OpenCV_Type = CV_8UC3;
                        break;
                case SapFormatRGB161616:
                        OpenCV_Type = CV_16UC3;
                        break;
                default:
                        sapFormat = SapFormatUnknown;
                        break;
                }

        if( sapFormat != SapFormatUnknown )
        {
                // Export to OpenCV Mat object copying SapBuffer data to Mat object
                Mat             exportImg;

                exportImg.create( pSapBuf->GetHeight(), pSapBuf->GetWidth(), OpenCV_Type );
                pSapBuf->Read( 0, pSapBuf->GetWidth() * pSapBuf->GetHeight(), exportImg.data );

                namedWindow( OPENCV_WINDOW_NAME, WINDOW_NORMAL | CV_WINDOW_KEEPRATIO );

                // Display OpenCV Image
                imshow( OPENCV_WINDOW_NAME, exportImg );

                waitKey(1);
        }
}
```