APPLICATION NOTE

```
// Create an instant camera object with the firs
Camera_t camera( CTlFactory::GetInstance().Creat

// Register an image event handler that accesses
camera.RegisterImageEventHandler( new CSampleIma
Ownership_TakeOwnership);

// Open the camera.
camera.Open();
```

Getting Started with pylon and OpenCV

Applicable to all Basler USB3 Vision and GigE Vision cameras

Document Number: AW001368
Version: 03   Language: 000 (English)
Release Date: 22 March 2021
Software Version (pylon): 6.x

BASLER
the power of sight

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany

Tel. +49 4102 463 515
Fax +49 4102 463 599

support.europe@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA

Tel. +1 610 280 0171
Fax +1 610 280 7608

support.usa@baslerweb.com

Asia-Pacific

Basler Asia Pte. Ltd.
35 Marsiling Industrial Estate Road 3
#05–06
Singapore 739257

Tel. +65 6367 1355
Fax +65 6367 1255

support.asia@baslerweb.com

www.baslerweb.com

Table of Contents

1 Introduction 2

2 Requirements 2

2.1 Basler pylon Camera Software Suite 2
2.2 OpenCV 2
2.3 Microsoft Visual Studio 2

3 OpenCV Installation 3

4 Common Settings for Building Applications with pylon and OpenCV in Visual Studio 4

5 Sample Code to Run pylon and OpenCV 7

Revision History 10

1 Introduction

OpenCV is an open-source computer vision library that allows you to perform image processing in combination with Basler Machine Vision cameras. This application note provides information on how to install and use OpenCV together with the Basler pylon Camera Software Suite in Microsoft Visual Studio on Windows.

OpenCV does not support Machine Vision standards such as USB3 Vision or GigE Vision. Therefore, Basler does not recommend grabbing images using OpenCV API functions. Instead, you should use the pylon Camera Software Suite drivers and APIs, e.g., the pylon C++ API, to grab images and convert them to OpenCV images.

This document provides further information on the integration of OpenCV functions for image display, image saving, and video recording into your pylon source code.

2 Requirements

2.1 Basler pylon Camera Software Suite

The procedures described in this application note are based on the pylon Camera Software Suite version 6.2.0 (64 bit).

You can download the pylon Camera Software Suite version 6.2.x or higher from the **Downloads Software** section of the Basler website: www.baslerweb.com.

For information about how to install the pylon Camera Software Suite, see the *Installation and Setup Guide for Cameras Used with pylon for Windows* (AW000611) and the *pylon Deployment Guide* (AW001362). You can download the documents from the **Download Documents** section of the Basler website: www.baslerweb.com.

2.2 OpenCV

This document uses OpenCV version 4.5.0 as an example. The procedures described also apply to older or newer versions. For versions different from the one used in this document, make adjustments as necessary. You can download OpenCV for Windows from www.opencv.org.

2.3 Microsoft Visual Studio

The procedures described in this application note apply to Microsoft Visual Studio 2017 on a Windows 10 64-bit operating system and serve as an example.

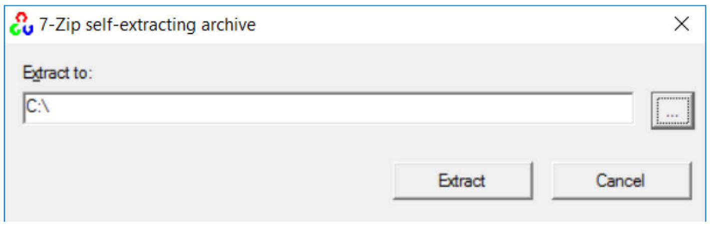
In principle, the procedures can also be applied to older versions of Microsoft Visual Studio, for example Microsoft Visual Studio 2015.

3 OpenCV Installation

ℹ	Code samples in the instructions below and data to be entered in the relevant fields in the screenshots assume that Visual Studio 2017, Windows 10 64-bit, and OpenCV 4.5.0 are used. Adjust as necessary if your setup differs. For example, assuming you use OpenCV 4.5.0: Replace `vc15` by `vc14` if you're using Visual Studio 2015.

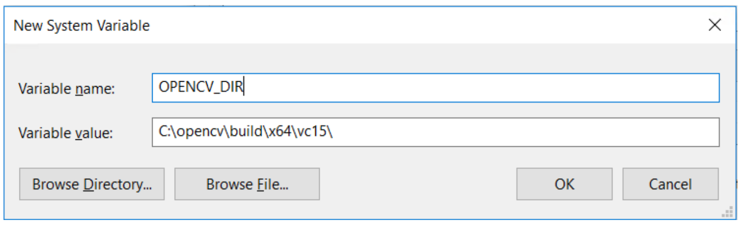
To set up the environment variables for OpenCV:

1. Extract the OpenCV files to C:\.



C:\

2. In the Windows search bar, type 'environment variable' and select **Edit the system environment variables**.
3. On the **Advanced** tab of the **System Properties** dialog box, click **Environment Variables**.
4. In the **System Variables** area of the **Environment Variables** dialog box, scroll to **Path** and select it.
5. Click the **Edit** button below the **System Variables** area.
6. In the **Edit Environment Variable** dialog box, klick **New** and add the following:
   `C:\opencv\build\x64\vc15\bin`
7. Click **OK**.
8. Create an OpenCV environment variable to speed up your work. The variable will hold the path to the build directory of the OpenCV library:
   a. Click the **New** button below the **System Variables** area.
   b. In the **New System Variable** dialog box, type `OPENCV_DIR` in the **Variable name** field.
   c. In the **Variable value** field, type: `C:\opencv\build\x64\vc15\`



Variable name: OPENCV_DIR
Variable value: C:\opencv\build\x64\vc15\

4 Common Settings for Building Applications with pylon and OpenCV in Visual Studio

The steps below outline how to configure Visual Studio in order to use OpenCV with the pylon C++ API.

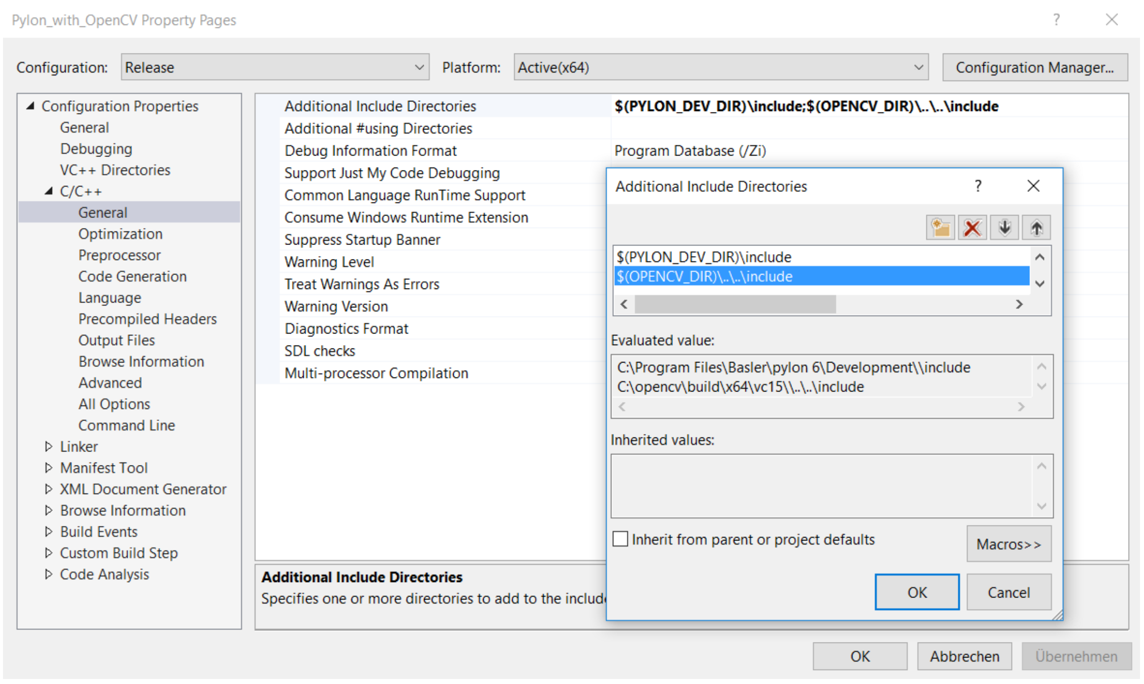
For more information about how to set up your Microsoft Visual Studio project for building applications with pylon, refer to the *Programmer's Guide and API Reference for pylon for Windows*. This guide is part of the pylon Camera Software Suite installation (**Start** > **Basler** > **C++ Programmer's Guide and API Reference Documentation** or, in the **pylon Viewer**, click **Help** > **C++ Programmer's Guide**).

This application note assumes that you have already read the *Programmer's Guide and API Reference for pylon for Windows* (especially the topic *Programmer's Guide* and its section *Common Settings for Building Applications with pylon*) and have set up the PYLON_DEV_DIR environment variable for building a 64-bit application in your project. The project used in this document is named "Pylon_with_OpenCV".

Sections 2.2 and 2.3 list the assumptions made for the instructions below.

To configure the Visual Studio project for OpenCV:

1. Open your project in Visual Studio 2017, go to **Project** and select **Properties**:
2. In the **Property Pages** window, select **C/C++** > **General**.
3. From the **Additional Include Directories** drop down, select **Edit** and add the OpenCV include directory: `$(OPENCV_DIR)\..\..\include`



Configuration: Release
Platform: Active(x64)

Additional Include Directories	$(PYLON_DEV_DIR)\include;$(OPENCV_DIR)\..\..\include
Additional #using Directories	
Debug Information Format	Program Database (/Zi)
Support Just My Code Debugging	
Common Language RunTime Support	
Consume Windows Runtime Extension	
Suppress Startup Banner	
Warning Level	
Treat Warnings As Errors	
Warning Version	
Diagnostics Format	
SDL checks	
Multi-processor Compilation	

Additional Include Directories
Specifies one or more directories to add to the includ

Additional Include Directories

$(PYLON_DEV_DIR)\include
$(OPENCV_DIR)\..\..\include

Evaluated value:

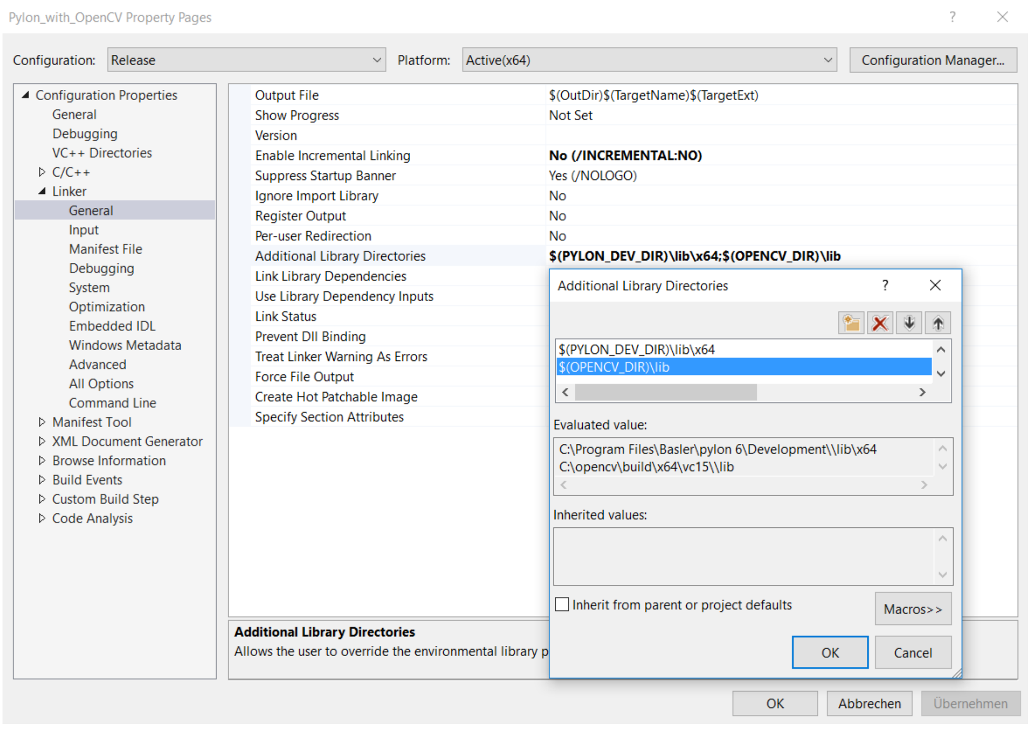
C:\Program Files\Basler\pylon 6\Development\\include
C:\opencv\build\x64\vc15\\..\..\include

Inherited values:

☐ Inherit from parent or project defaults

4. Click **OK**.

5. In the **Property Pages** window, select **Linker** > **General**.
6. From the **Additional Library Directories** drop down, select **Edit** and add the OpenCV library directory: `$(OPENCV_DIR)\lib`


Pylon_with_OpenCV Property Pages

Configuration: Release Platform: Active(x64)

Output File	$(OutDir)$(TargetName)$(TargetExt)
Show Progress	Not Set
Version	
Enable Incremental Linking	**No (/INCREMENTAL:NO)**
Suppress Startup Banner	Yes (/NOLOGO)
Ignore Import Library	No
Register Output	No
Per-user Redirection	No
Additional Library Directories	**$(PYLON_DEV_DIR)\lib\x64;$(OPENCV_DIR)\lib**

Additional Library Directories

$(PYLON_DEV_DIR)\lib\x64
$(OPENCV_DIR)\lib

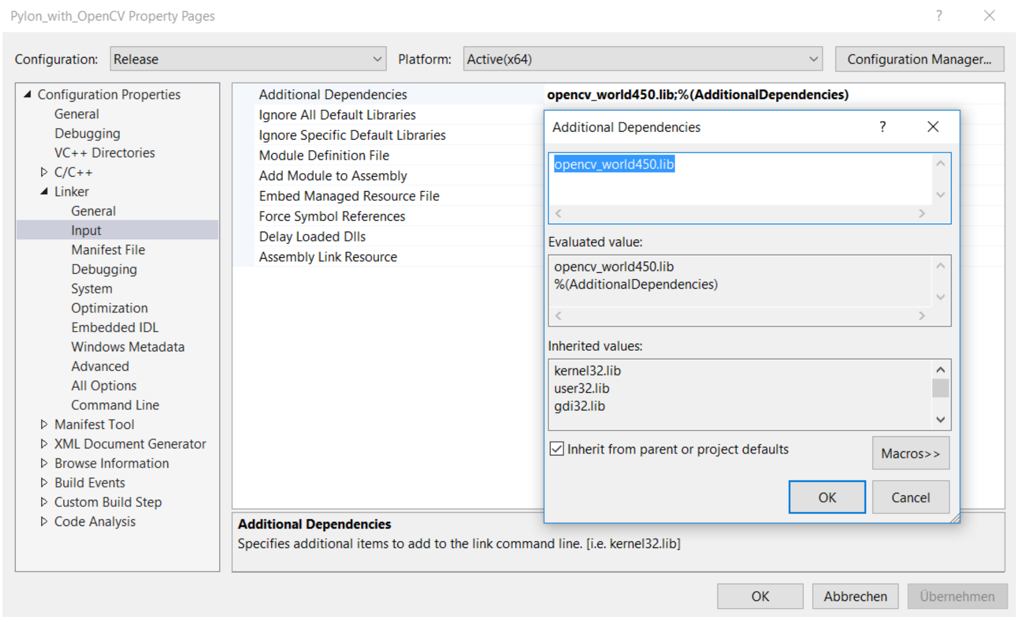
Evaluated value:

C:\Program Files\Basler\pylon 6\Development\\lib\x64
C:\opencv\build\x64\vc15\\lib

Inherited values:

Additional Library Directories
Allows the user to override the environmental library p

7. Click **OK**.
8. In the **Property Pages** window, select **Linker** > **Input**.
9. From the **Additional Dependencies** drop down, select **Edit** and add the following OpenCV lib file:
   - `opencv_world450.lib`


Pylon_with_OpenCV Property Pages

Configuration: Release Platform: Active(x64)

Additional Dependencies	**opencv_world450.lib;%(AdditionalDependencies)**
Ignore All Default Libraries	
Ignore Specific Default Libraries	
Module Definition File	
Add Module to Assembly	
Embed Managed Resource File	
Force Symbol References	
Delay Loaded Dlls	
Assembly Link Resource	

Additional Dependencies

opencv_world450.lib

Evaluated value:

opencv_world450.lib
%(AdditionalDependencies)

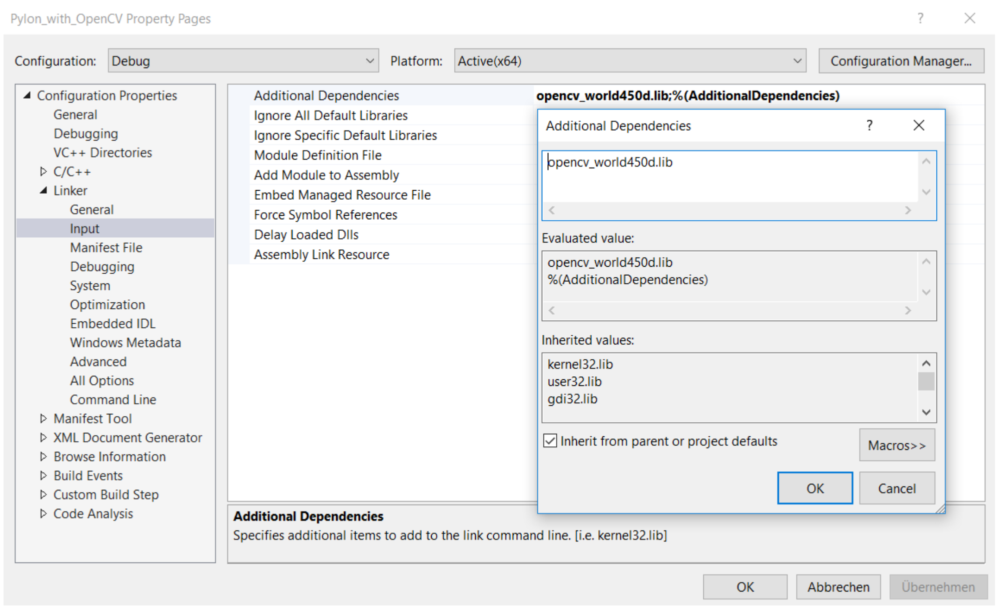
Inherited values:

kernel32.lib
user32.lib
gdi32.lib

Inherit from parent or project defaults

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

10. Click **OK**.
11. Click **Apply**.
12. To configure your project for **Debug** mode, do the following:
    a. In the **Configuration** drop down box, select **Debug**.
    b. Repeat steps 2 to 8.
    c. In step 9, add the following OpenCV debug lib file:
       - `opencv_world450d.lib`


Pylon_with_OpenCV Property Pages

Configuration: Debug Platform: Active(x64)

Additional Dependencies: opencv_world450d.lib;%(AdditionalDependencies)

Additional Dependencies

opencv_world450d.lib

Evaluated value:

opencv_world450d.lib
%(AdditionalDependencies)

Inherited values:

kernel32.lib
user32.lib
gdi32.lib

☑ Inherit from parent or project defaults

Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]

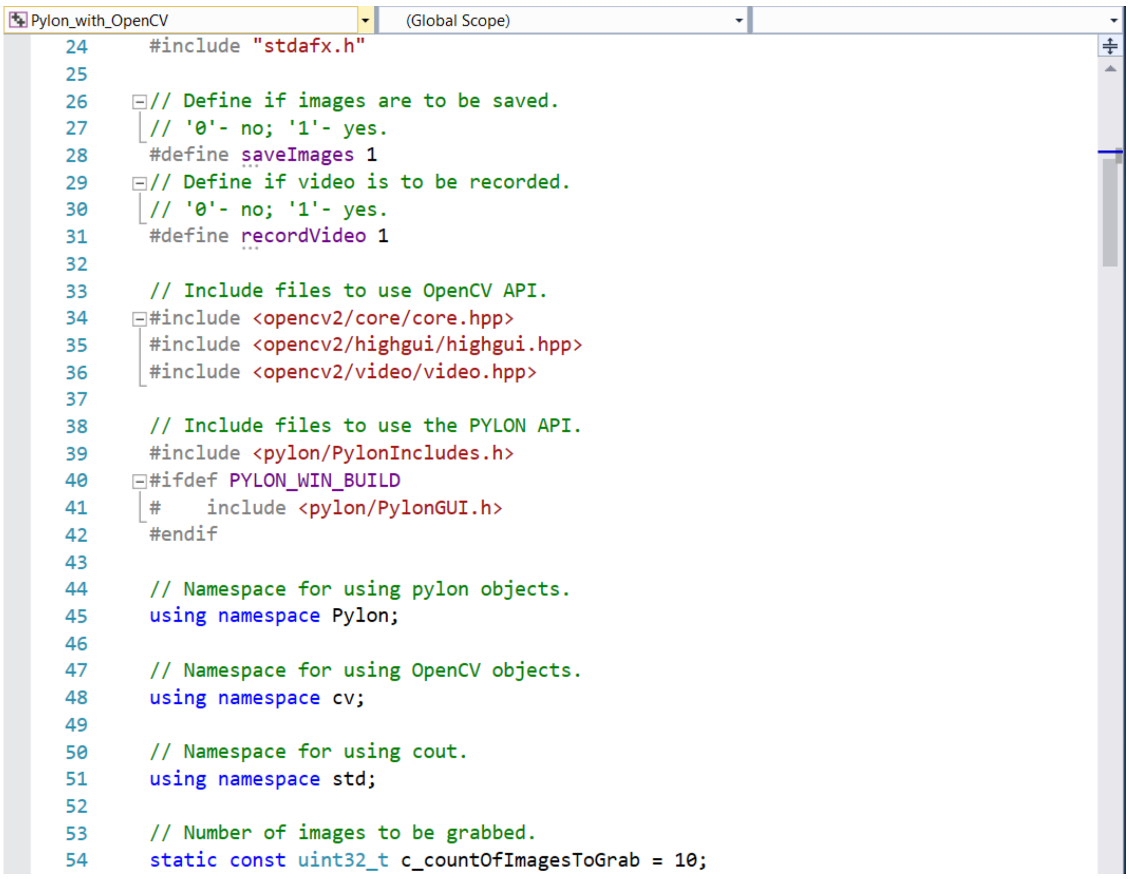
    d. Click **OK**.
    e. Click **Apply**.

5 Sample Code to Run pylon and OpenCV

The following sample code demonstrates how to accomplish these tasks:

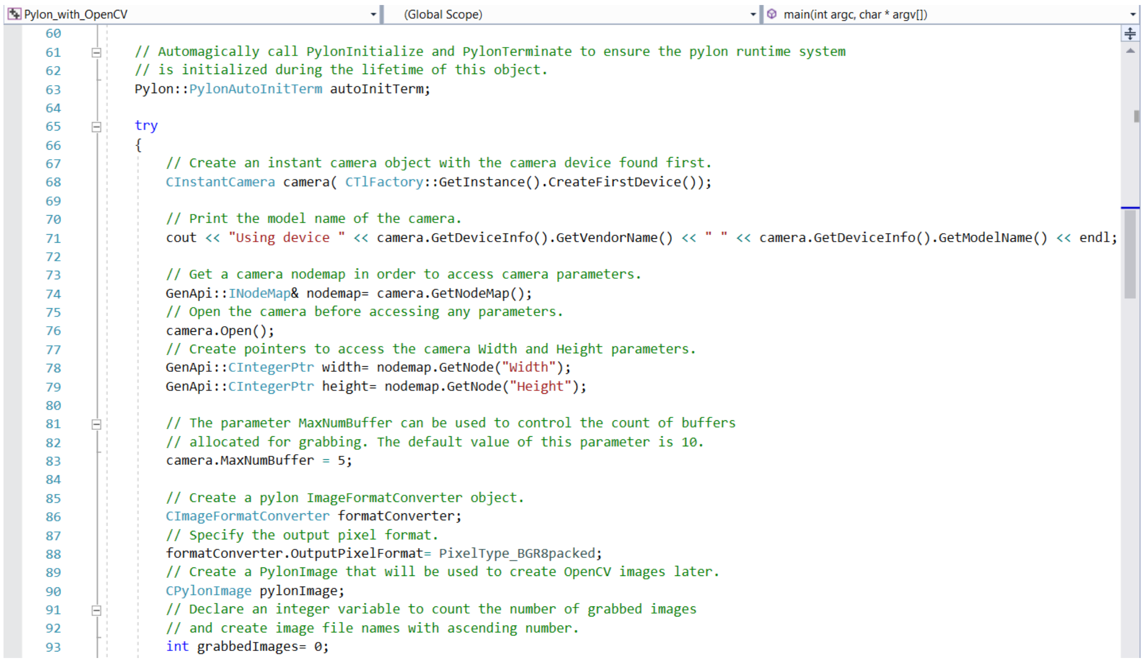
- Acquire images with pylon C++ API functions.
- Convert the acquired pylon images to OpenCV images.
- Display and save images with OpenCV functions.
- Record video files with OpenCV functions.

1. Include all necessary OpenCV and pylon API header files, declare required namespaces, and define some global variables needed throughout the sample.



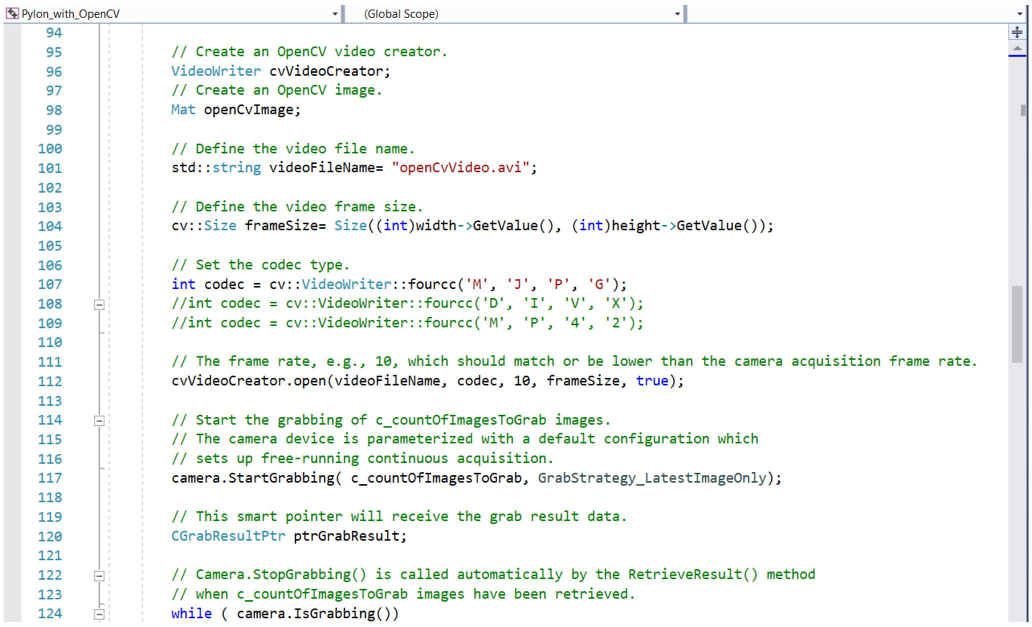
```
#include "stdafx.h"

// Define if images are to be saved.
// '0'- no; '1'- yes.
#define saveImages 1
// Define if video is to be recorded.
// '0'- no; '1'- yes.
#define recordVideo 1

// Include files to use OpenCV API.
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/video/video.hpp>

// Include files to use the PYLON API.
#include <pylon/PylonIncludes.h>
#ifdef PYLON_WIN_BUILD
#    include <pylon/PylonGUI.h>
#endif

// Namespace for using pylon objects.
using namespace Pylon;

// Namespace for using OpenCV objects.
using namespace cv;

// Namespace for using cout.
using namespace std;

// Number of images to be grabbed.
static const uint32_t c_countOfImagesToGrab = 10;
```

2. Create the first camera found regardless of its interface, i.e., GigE Vision or USB3 Vision. Get the camera's GenICam *nodemap* to access the camera's *Width* and *Height* parameters, which are needed for the initialization of the OpenCV video creator. Set up a pylon image format converter object and define the desired output pixel format, which is needed to convert the grabbed pylon image buffer to a pylon image. Convert the pylon image to an OpenCV image.



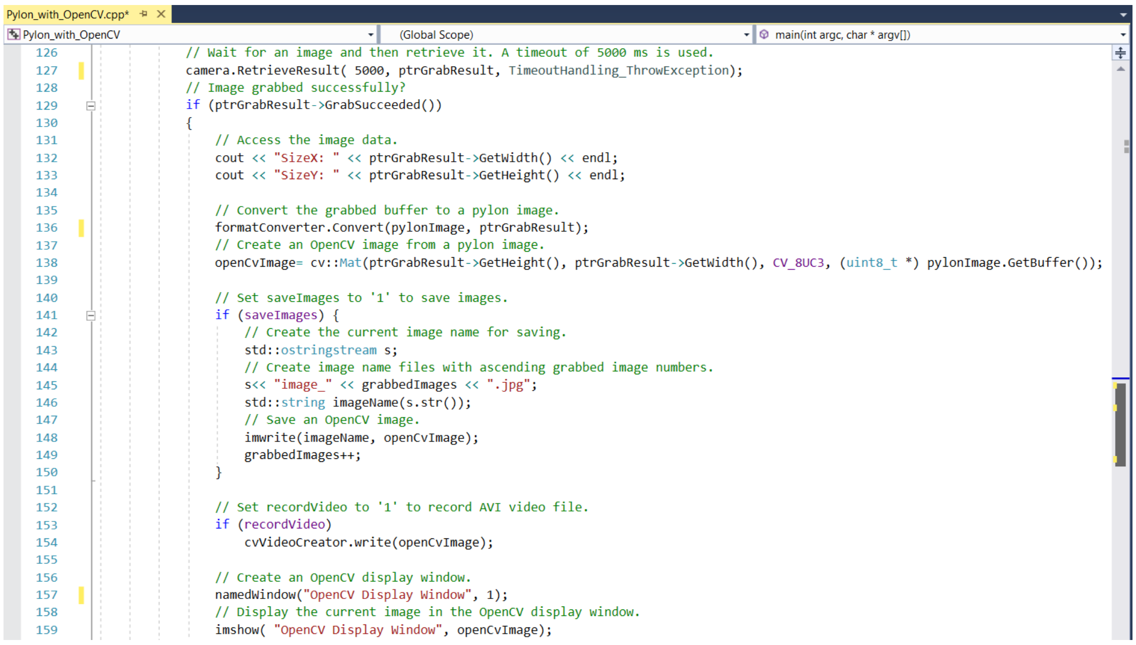
```

    // Automagically call PylonInitialize and PylonTerminate to ensure the pylon runtime system
    // is initialized during the lifetime of this object.
    Pylon::PylonAutoInitTerm autoInitTerm;

    try
    {
        // Create an instant camera object with the camera device found first.
        CInstantCamera camera( CTlFactory::GetInstance().CreateFirstDevice());

        // Print the model name of the camera.
        cout << "Using device " << camera.GetDeviceInfo().GetVendorName() << " " << camera.GetDeviceInfo().GetModelName() << endl;

        // Get a camera nodemap in order to access camera parameters.
        GenApi::INodeMap& nodemap= camera.GetNodeMap();
        // Open the camera before accessing any parameters.
        camera.Open();
        // Create pointers to access the camera Width and Height parameters.
        GenApi::CIntegerPtr width= nodemap.GetNode("Width");
        GenApi::CIntegerPtr height= nodemap.GetNode("Height");

        // The parameter MaxNumBuffer can be used to control the count of buffers
        // allocated for grabbing. The default value of this parameter is 10.
        camera.MaxNumBuffer = 5;

        // Create a pylon ImageFormatConverter object.
        CImageFormatConverter formatConverter;
        // Specify the output pixel format.
        formatConverter.OutputPixelFormat= PixelType_BGR8packed;
        // Create a PylonImage that will be used to create OpenCV images later.
        CPylonImage pylonImage;
        // Declare an integer variable to count the number of grabbed images
        // and create image file names with ascending number.
        int grabbedImages= 0;
```

3. Declare and initialize the OpenCV video creator and OpenCV image objects. Use pylon API functions to start image acquisition and retrieve the grabbed images:



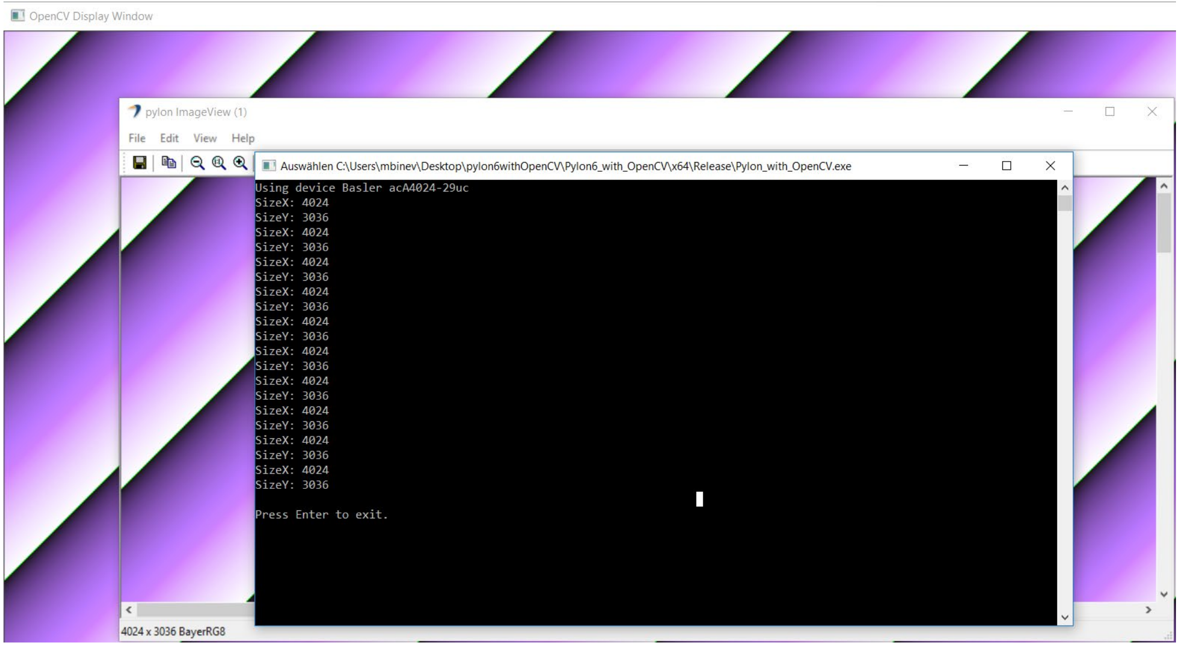
```

        // Create an OpenCV video creator.
        VideoWriter cvVideoCreator;
        // Create an OpenCV image.
        Mat openCvImage;

        // Define the video file name.
        std::string videoFileName= "openCvVideo.avi";

        // Define the video frame size.
        cv::Size frameSize= Size((int)width->GetValue(), (int)height->GetValue());

        // Set the codec type.
        int codec = cv::VideoWriter::fourcc('M', 'J', 'P', 'G');
        //int codec = cv::VideoWriter::fourcc('D', 'I', 'V', 'X');
        //int codec = cv::VideoWriter::fourcc('M', 'P', '4', '2');

        // The frame rate, e.g., 10, which should match or be lower than the camera acquisition frame rate.
        cvVideoCreator.open(videoFileName, codec, 10, frameSize, true);

        // Start the grabbing of c_countOfImagesToGrab images.
        // The camera device is parameterized with a default configuration which
        // sets up free-running continuous acquisition.
        camera.StartGrabbing( c_countOfImagesToGrab, GrabStrategy_LatestImageOnly);

        // This smart pointer will receive the grab result data.
        CGrabResultPtr ptrGrabResult;

        // Camera.StopGrabbing() is called automatically by the RetrieveResult() method
        // when c_countOfImagesToGrab images have been retrieved.
        while ( camera.IsGrabbing())
```

4. Convert the grabbed image buffer to a pylon image, which in turn is converted to an OpenCV image. The OpenCV image is used for image display, image saving, and video file recording using OpenCV functions:



```
// Wait for an image and then retrieve it. A timeout of 5000 ms is used.
camera.RetrieveResult( 5000, ptrGrabResult, TimeoutHandling_ThrowException);
// Image grabbed successfully?
if (ptrGrabResult->GrabSucceeded())
{
    // Access the image data.
    cout << "SizeX: " << ptrGrabResult->GetWidth() << endl;
    cout << "SizeY: " << ptrGrabResult->GetHeight() << endl;

    // Convert the grabbed buffer to a pylon image.
    formatConverter.Convert(pylonImage, ptrGrabResult);
    // Create an OpenCV image from a pylon image.
    openCvImage= cv::Mat(ptrGrabResult->GetHeight(), ptrGrabResult->GetWidth(), CV_8UC3, (uint8_t *) pylonImage.GetBuffer());

    // Set saveImages to '1' to save images.
    if (saveImages) {
        // Create the current image name for saving.
        std::ostringstream s;
        // Create image name files with ascending grabbed image numbers.
        s<< "image_" << grabbedImages << ".jpg";
        std::string imageName(s.str());
        // Save an OpenCV image.
        imwrite(imageName, openCvImage);
        grabbedImages++;
    }

    // Set recordVideo to '1' to record AVI video file.
    if (recordVideo)
        cvVideoCreator.write(openCvImage);

    // Create an OpenCV display window.
    namedWindow("OpenCV Display Window", 1);
    // Display the current image in the OpenCV display window.
    imshow( "OpenCV Display Window", openCvImage);
```

The image below shows the running application with a test image grabbed with a Basler ace USB3 Vision camera in the OpenCV and pylon image display windows:


OpenCV Display Window
pylon ImageView (1)
File
Edit
View
Help
Auswählen C:\Users\mbinev\Desktop\pylon6withOpenCV\Pylon6_with_OpenCV\x64\Release\Pylon_with_OpenCV.exe
4024 x 3036 BayerRG8

```
Using device Basler acA4024-29uc
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036
SizeX: 4024
SizeY: 3036

Press Enter to exit.
```

If you want to get the complete sample source code, get in contact with your local Basler support team as listed at the beginning of this document.

Revision History

Document Number	Date	Changes
AW001368**01**000	18 Nov 2015	Initial release version of this document.
AW001368**02**000	03 Jan 2019	Removed all reference to IEEE 1394. Added reference to document AW001362. Introduced generalizations to account for higher software versions (Windows, Microsoft Visual Studio, OpenCV, pylon). Added some information about modifications necessary for higher software versions.
AW001368**03**000	22 Mar 2020	Updated the software requirements: pylon version 6.2, OpenCV version 4.5.0, Microsoft Visual Studio 2017, and Windows 10 64-bit.