













PATH=C:\Users\X-Studio\Downloads\opencv-4.6.0\opencv-4.6.0\build\install\x64\vc16\bin;$(PATH)

#include <opencv2/opencv.hpp>

#include <pylon/PylonIncludes.h>

#ifdef PYLON\_WIN\_BUILD

# include <pylon/PylonGUI.h>

#endif

// Namespace for using pylon objects.

using namespace Pylon;

// Namespace for using cout.

using namespace std;

// Number of images to be grabbed.

static const uint32\_t c\_countOfImagesToGrab = 100;

int main(int /\*argc\*/, char\* /\*argv\*/[])

{

// The exit code of the sample application.

int exitCode = 0;

// Before using any pylon methods, the pylon runtime must be initialized.

PylonInitialize();

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().GetModelName() << endl;

// 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;

// 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);

// This smart pointer will receive the grab result data.

CGrabResultPtr ptrGrabResult;

CImageFormatConverter formatConverter;

formatConverter.OutputPixelFormat = PixelType\_BGR8packed;

CPylonImage pylonImage;

cv::Mat frame;

// Camera.StopGrabbing() is called automatically by the RetrieveResult() method

// when c\_countOfImagesToGrab images have been retrieved.

while (camera.IsGrabbing())

{

// 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;

formatConverter.Convert(pylonImage, ptrGrabResult);

frame = cv::Mat(ptrGrabResult->GetHeight(), ptrGrabResult->GetWidth(), CV\_8UC3, (uint8\_t \*)pylonImage.GetBuffer());

/\*cout << "Gray value of first pixel: " << (uint32\_t)pImageBuffer[0] << endl << endl;\*/

#ifdef PYLON\_WIN\_BUILD

// Display the grabbed image.

/\*Pylon::DisplayImage(1, ptrGrabResult);\*/

cv::imshow("frame", frame);

cv::waitKey(30);

#endif

}

else

{

cout << "Error: " << std::hex << ptrGrabResult->GetErrorCode() << std::dec << " " << ptrGrabResult->GetErrorDescription() << endl;

}

}

}

catch (const GenericException& e)

{

// Error handling.

cerr << "An exception occurred." << endl

<< e.GetDescription() << endl;

exitCode = 1;

}

// Comment the following two lines to disable waiting on exit.

cerr << endl << "Press enter to exit." << endl;

while (cin.get() != '\n');

// Releases all pylon resources.

PylonTerminate();

return exitCode;

}