



OPEN CV



GIỚI THIỆU OPENCV

- **OpenCV** là viết tắt của **Open Source Computer Vision** , là thư viện xử lý ảnh mã nguồn mở hoàn toàn miễn phí của Intel. OpenCV là một thư viện mở gồm các hàm được xây dựng phục vụ cho việc xử lý thị giác máy thời gian thực (Real time computer vision). Các thuật toán xử lý ảnh thông thường lẫn cao cấp đều được tối ưu hóa bởi các nhà phát triển thư viện thành các hàm đơn giản và rất dễ sử dụng. Các version thường xuyên được cập nhật theo thời gian.
- Các version: 1.0, 1.1, 2.0, 2.1, 2.2 ,3.x
- Hiện nay đã có bản Open CV 4.x (2018)
- *OpenCV có thể sử dụng trên nhiều hệ điều hành Windows, Linux, Mac OS, iOS and Android*
- *OpenCV dùng với nhiều ngôn ngữ lập trình như C, C++, C#, Visual C++, Python, Java*

GIỚI THIỆU OPENCV

- <https://opencv.org/>

- HĐH Android

[/opencv-android/3.4.0/opencv-3.4.0-android-sdk.zip](https://opencv.org/opencv-android/3.4.0/opencv-3.4.0-android-sdk.zip)

- HĐH Windows

[opencv-win/3.4.0/opencv-3.4.0-vc14_vc15.exe](https://opencv.org/opencv-win/3.4.0/opencv-3.4.0-vc14_vc15.exe)

- HĐH IOS

[/opencv-ios/3.4.0/opencv-3.4.0-ios-framework.zip](https://opencv.org/opencv-ios/3.4.0/opencv-3.4.0-ios-framework.zip)

- HĐH UNIX UBUNTU

[/opencv-unix/3.4.0/opencv-3.4.0.zip](https://opencv.org/opencv-unix/3.4.0/opencv-3.4.0.zip)

- Tài liệu quan trọng để học Open CV là quyển **Learning OpenCV** của Garry Bradski và Adrian Kaehler, nhà xuất bản O'Reilly Media Inc, 2008, 2016
- <http://opencv.willowgarage.com/documentation/cpp/index.html>

Những điểm đặc trưng

- **Image and Video I/O**
 - Những giao diện này sẽ giúp bạn đọc được dữ liệu ảnh từ file hoặc trực tiếp từ video. Bạn cũng có thể tạo các file ảnh và video với giao diện này
- **Thị giác máy và các thuật toán xử lý ảnh (General computer-vision and image-processing algorithms (mid – and low level APIs))**
 - Sử dụng những giao diện này, bạn có thể thực hành với rất nhiều chuẩn thị giác máy mà không cần phải có mã nguồn của chúng.
- **Modul thị giác máy ở cấp độ cao**
 - OpenCV gồm một vài áp dụng ở cấp độ cao. Thêm vào nhận dạng mặt, dò tìm, theo dõi. Nó bao gồm luồng thị giác (sử dụng camera di động để xác định cấu trúc 3D), kích cỡ camera và âm thanh nổi.

Những điểm đặc trưng

- **AI and machine-learning**
 - Các ứng dụng của thị giác máy thường yêu cầu máy móc phải học (machine learning) hoặc các hình thức trí tuệ nhân tạo khác. Một vài trong số chúng là có sẵn trong gói OpenCV
- **Lấy mẫu ảnh và phép biến đổi**
 - Nó thường rất tốt cho quá trình xử lý một nhóm phần tử ảnh như là một đơn vị. OpenCV bao gồm lấy tách ra, lấy mẫu ngẫu nhiên, phục chế, xoay ảnh, làm cong ảnh (warping), thay đổi hiệu ứng của ảnh.
- **Cách thức tạo và phân tích ảnh nhị phân**
 - Ảnh nhị phân thường xuyên được dùng trong các hệ thống kiểm tra có khuyết điểm hình dạng hoặc các bộ phận quan trọng. Sự biểu diễn ảnh cũng rất thuận tiện khi chúng ta biết rõ vật thể cần bắt.

Những điểm đặc trưng

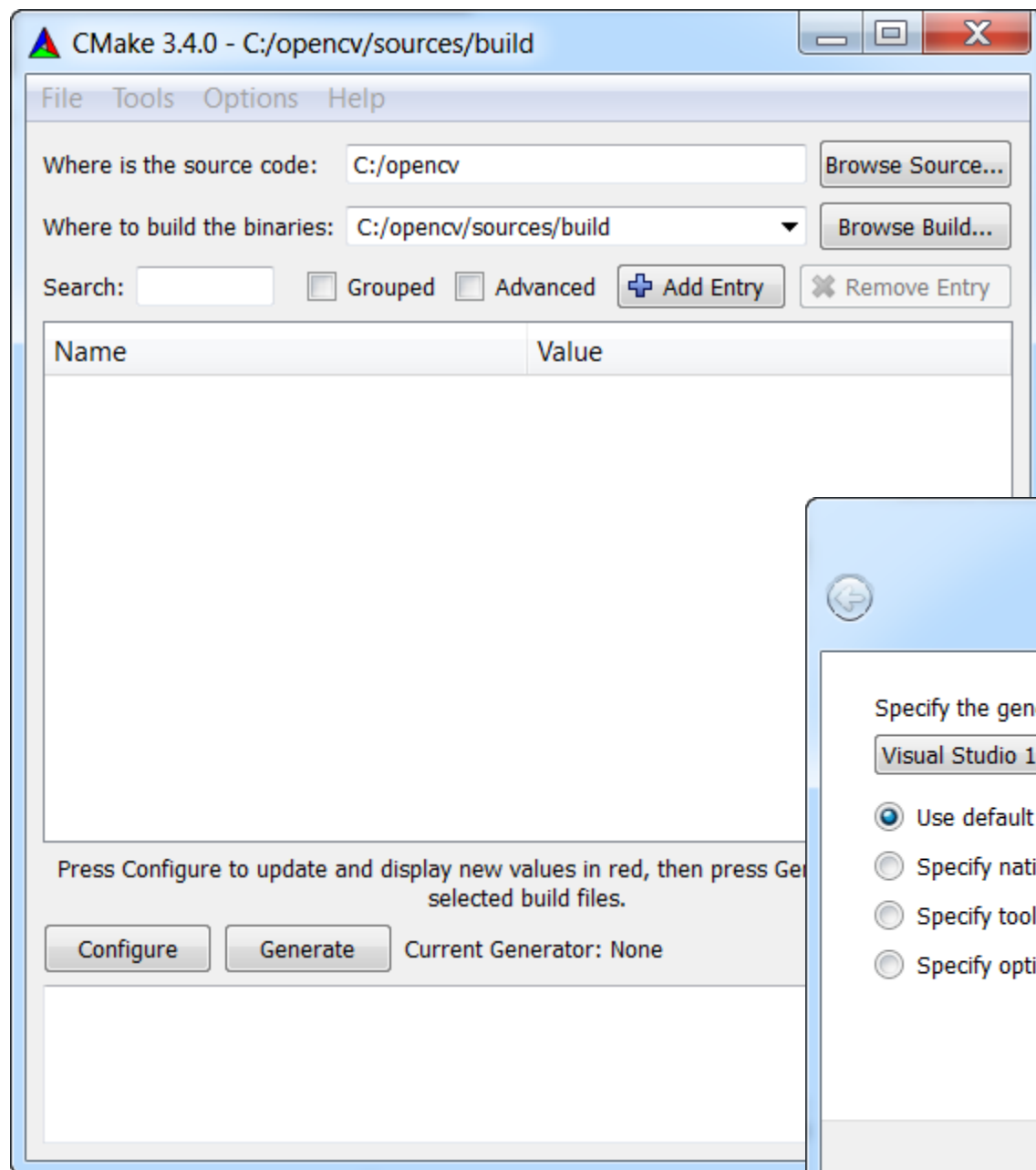
- **Các phương pháp tính toán thông tin 3D**
 - Những hàm này rất có ích khi cần sắp xếp và xác định một khối lập thể (with a stereo rig) hoặc với không gian nhìn phức tạp (multiple views) từ một camera riêng.
- **Các phép toán cho xử lý ảnh, thị giác máy và biểu diễn ảnh**
 - OpenCV sử dụng các phép toán phổ biến như: đại số học, thống kê và tính toán hình học
- **Đồ họa**
 - Những giao diện này giúp bạn viết chữ và vẽ trên hình ảnh. Thêm vào đó những chức năng này được sử dụng nhiều trong ghi nhãn và đánh dấu. Ví dụ nếu bạn viết một chương trình cần nhận dạng nhiều đối tượng thì nó sẽ rất có ích cho tạo nhãn ảnh (label image) với kích thước và vị trí.

Những điểm đặc trưng

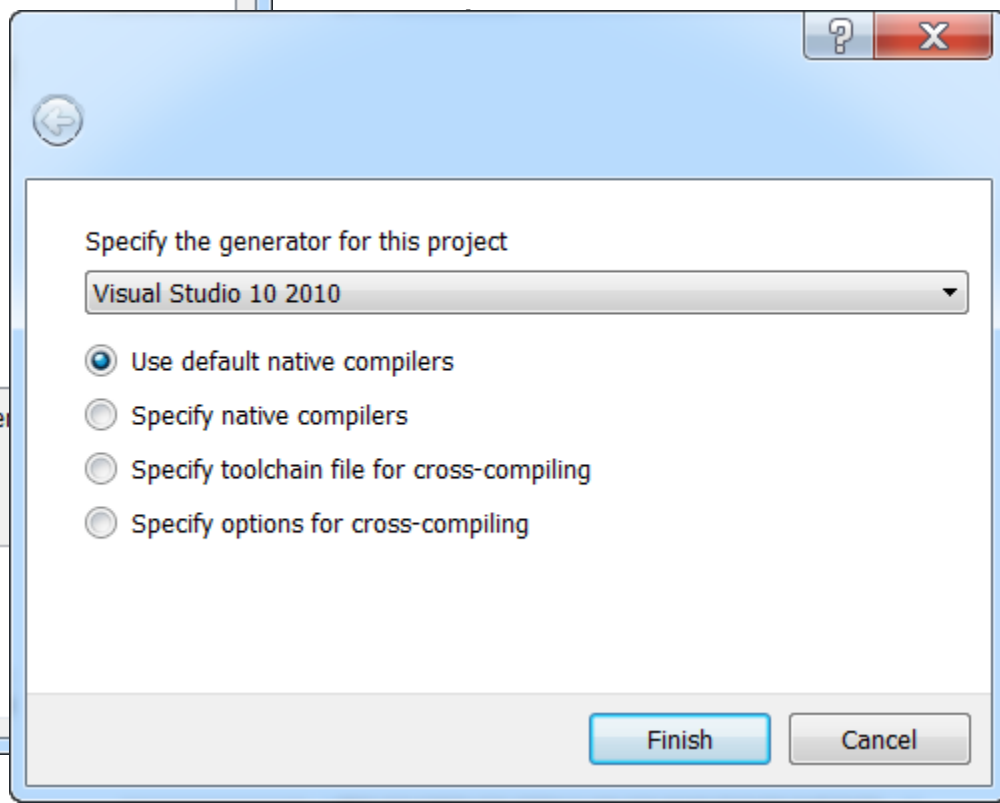
- **Phương thức GUI Giao diện người dùng (Graphic user Interface)**
 - OpenCV bao gồm cửa sổ giao diện của chính bản thân nó. Trong khi đó những giao diện này được so sánh giới hạn với khả năng có thể thực hiện trong mỗi môi trường. Chúng cung cấp những môi trường API đa phương tiện và đơn giản để hiển thị hình ảnh, cho phép người dùng nhập dữ liệu thông qua chuột, bàn phím và điều khiển quá trình.
- **Cấu trúc dữ liệu và giải thuật**
 - Với những giao diện này bạn có thể giữ lại, tìm kiếm, lưu và cách danh mục điều khiển, các tuyến tập (cũng như các tập hợp lệnh được gọi), đồ họa và sơ đồ nhánh một cách hiệu quả.

CÀI ĐẶT OPENCV CHO VISUAL STUDIO 2010 X86

- Bước 1: Download OpenCV và giải nén thư viện vào thư mục được định sẵn. Giả sử đường dẫn của mã nguồn là C:\opencv, sẽ có hai thư mục build và sources, Thư mục build chứa những tập tin của phần mã nguồn được biên dịch sẵn. Thư mục sources được sử dụng cho cài đặt manual, có chứa các file mẫu.
- Bước 2: để cài đặt tiếp ta download và cài đặt cmake <https://cmake.org/files/v3.4/cmake-3.4.0-win32-x86.exe>
- Cmake là công cụ dùng để cấu hình file mã nguồn mở cho nhiều ứng dụng khác nhau
- Bước 3: Mở CMake. Trong phần Where is the source code sẽ trở đến thư mục sources đã đề cập lúc trước. Phần Where to build the binaries là nơi chứa phần thư viện được biên dịch. Trong ví dụ này đường dẫn là C:\opencv\source\build



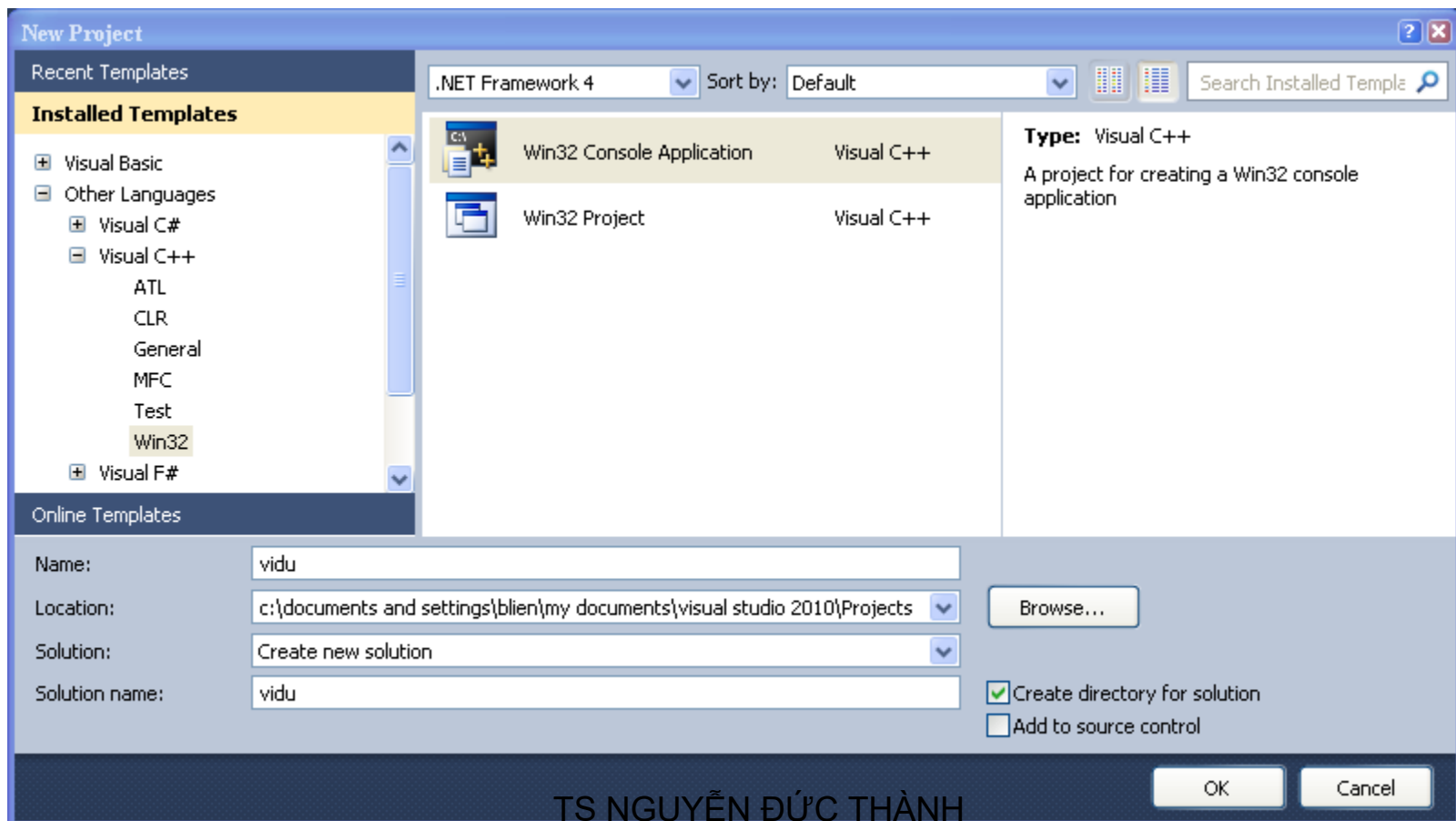
Sau đó bấm
Configure Chọn
Visual Studio phù
hợp rồi chọn
Generate. Sau khi
quá trình phát sinh
mã nguồn hoàn tất,
trong thư mục build
sẽ xuất hiện các
Project/Solution khác

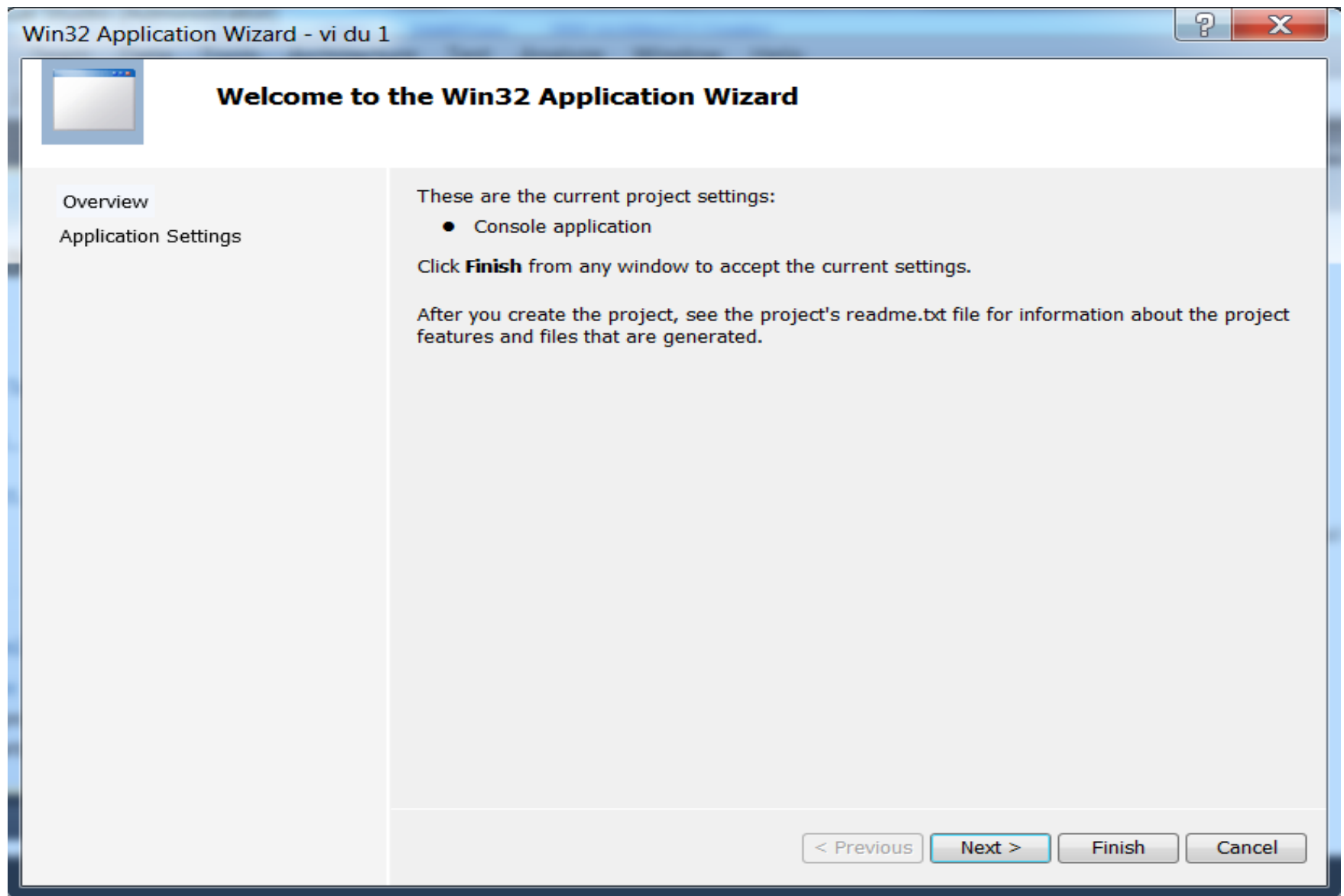


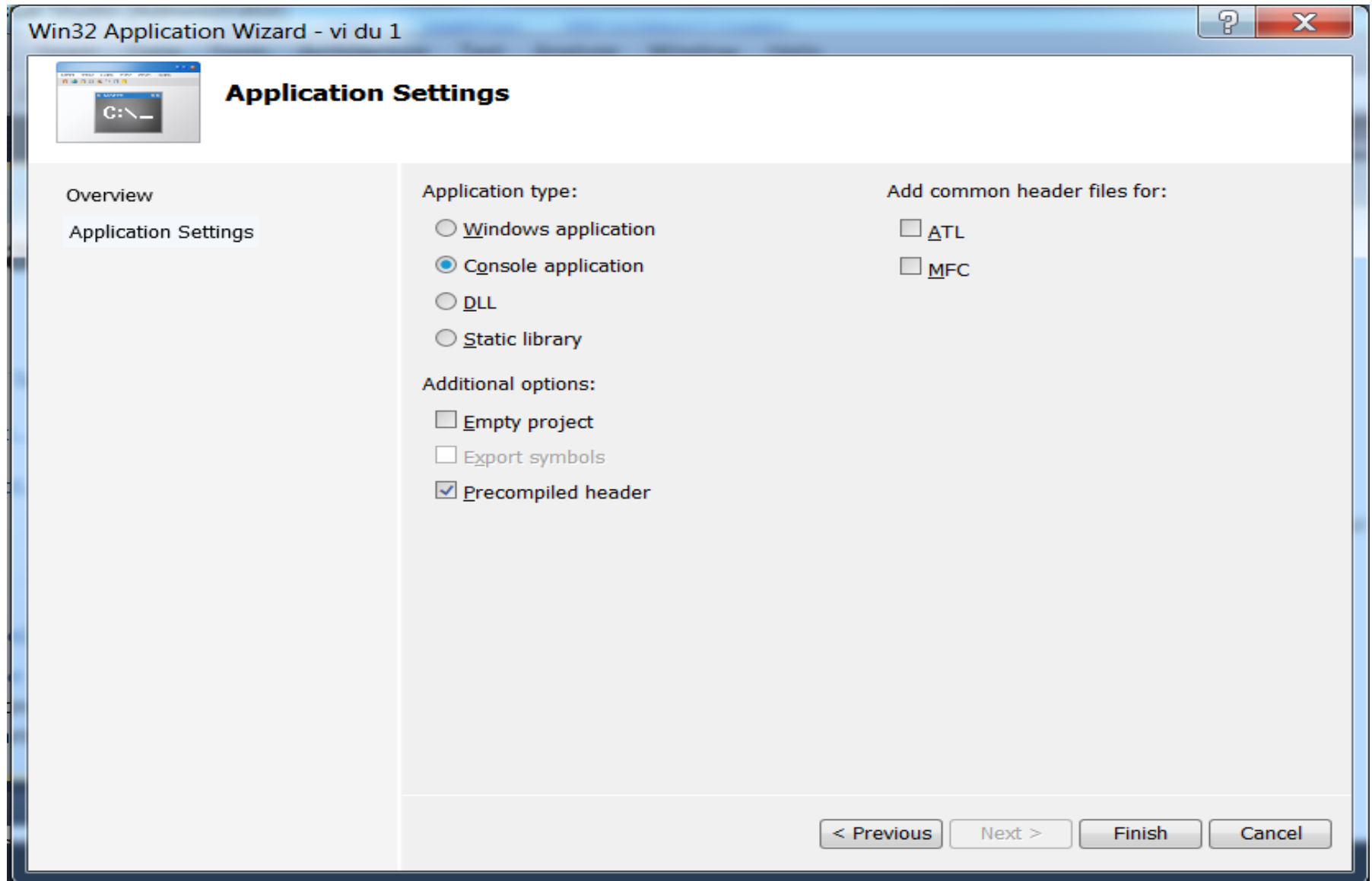
- Bước 4: Dùng Visual Studio để mở solution có tên là OPENCV.sln. Lưu ý: bản VS này phải trùng với bản VS đã configure trong cmake. Quá trình biên dịch cần thực hiện qua 2 bước: (1) biên dịch thư viện cho quá trình Debug và (2) biên dịch thư viện cho quá trình Release. Chỉ cần click chọn Target sẽ build (x86, x64) rồi sau đó nhấn F7 Sau khi biên dịch, ta sẽ có 2 thư mục cần quan tâm nằm trong build: thư mục lib chứa toàn bộ thư viện opencv có đuôi .lib; thư mục bin chứa file dll cần thiết để chương trình sau khi build có thể chạy được.
- Thư mục include chứa các file header khai báo các hàm, trong đó header .h dùng cho c và header .hpp dùng cho C++

OPENCV VÀ VS2010

- Vào File – New Project- Visual C++ Win32- Win32 Console Application tạo Project vidu1



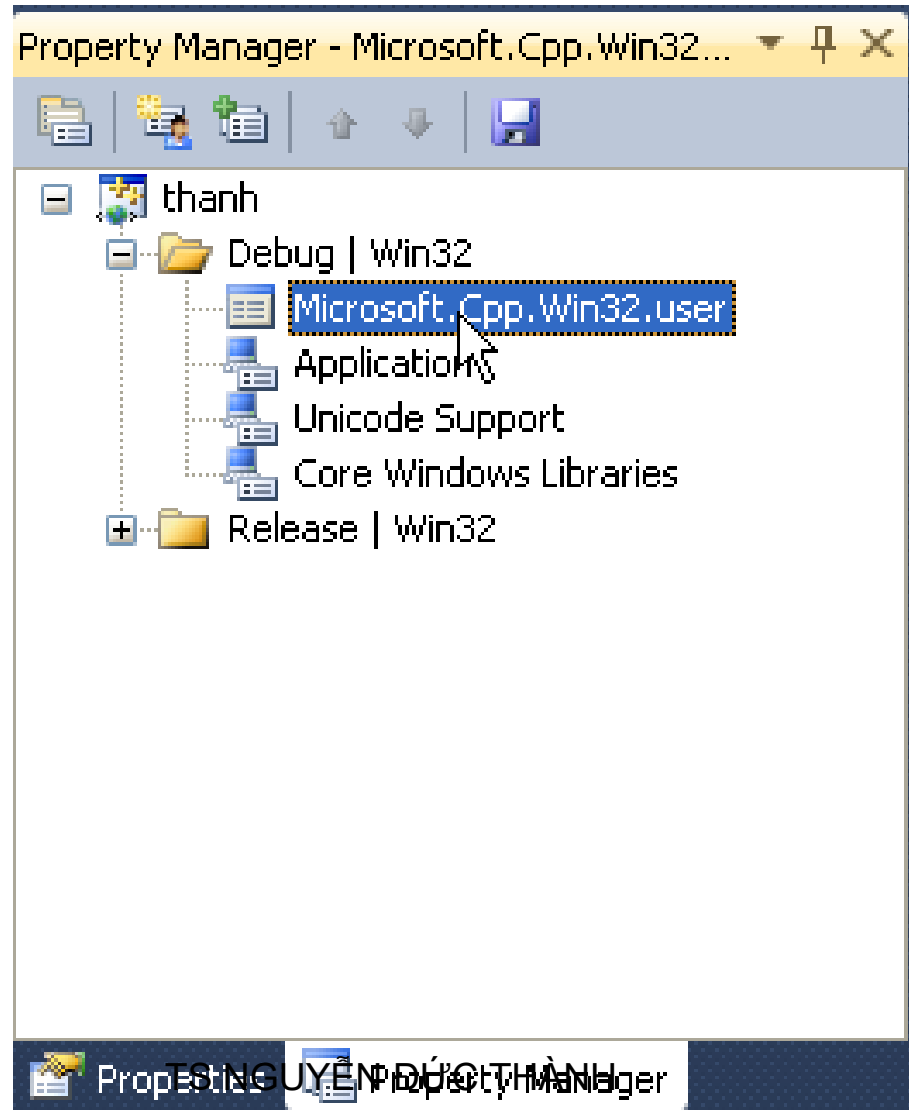




OPENCV VÀ VS2010

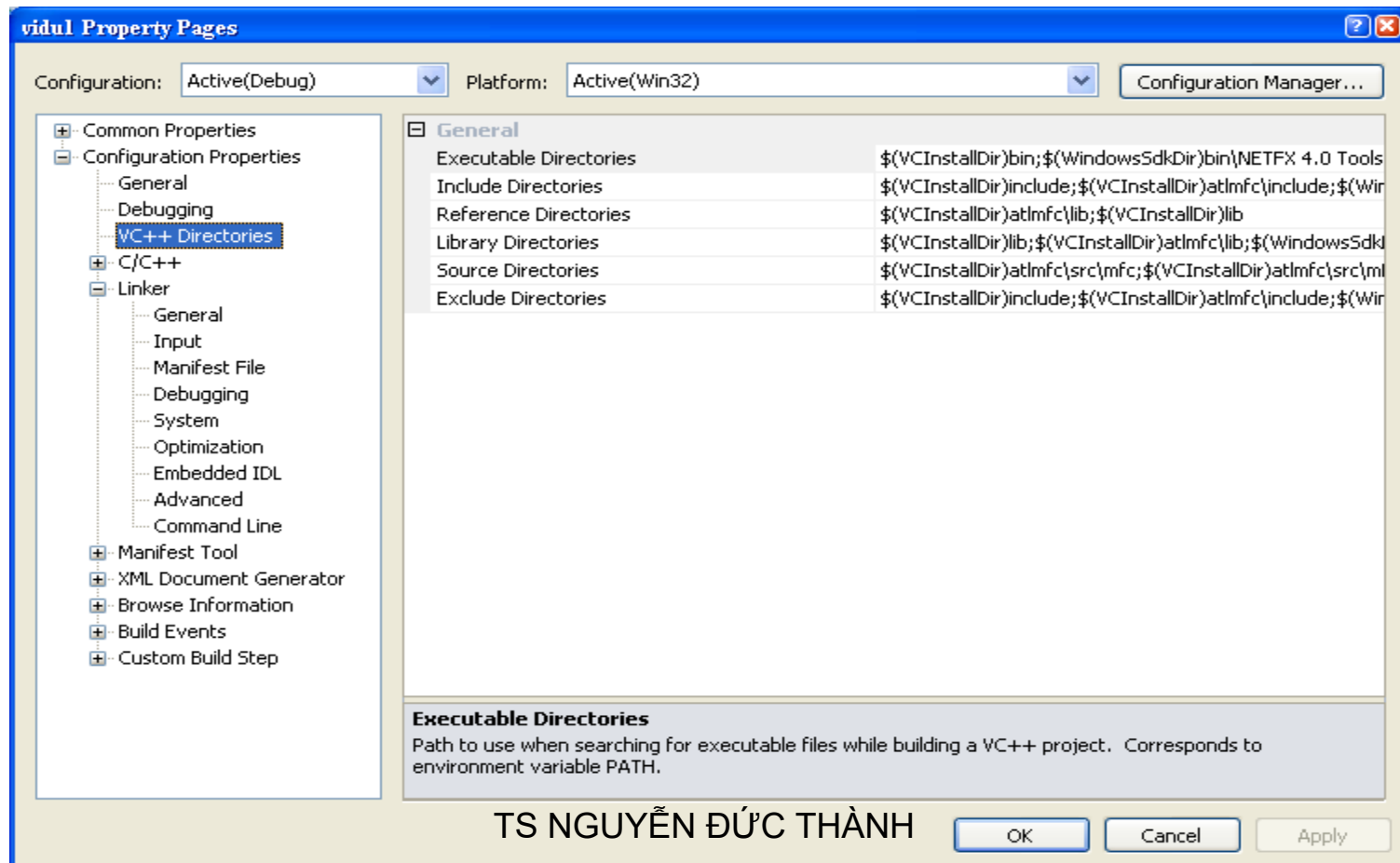
- Mở menu View- Other Windows- Property Manager – Debug/Win32 – bấm chuột kép vào dòng Microsoft.Cpp.Win32.user Sau đó thực hiện các bước sau. Sau đó lưu cấu hình lại . Khi ta tạo Project mới, VC++ sẽ biên dịch thành công các lệnh của OpenCV
- Làm tương tự với Property Manager – Release/Win32

OPENCV VÀ VS2010

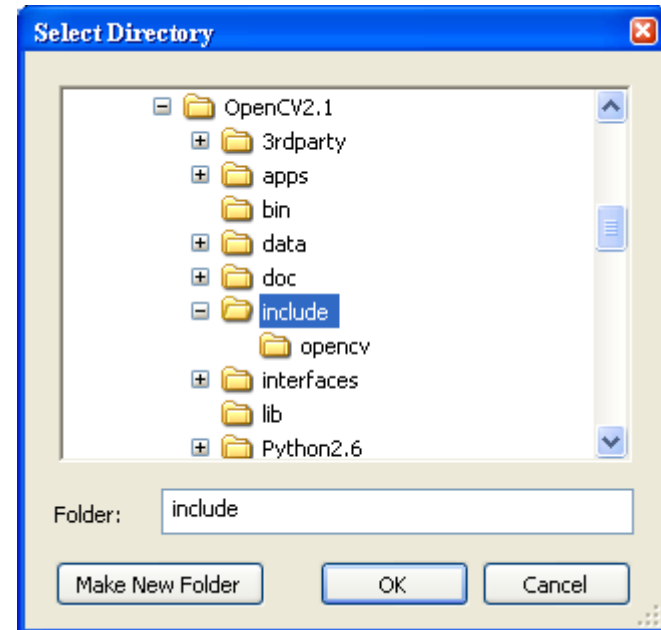
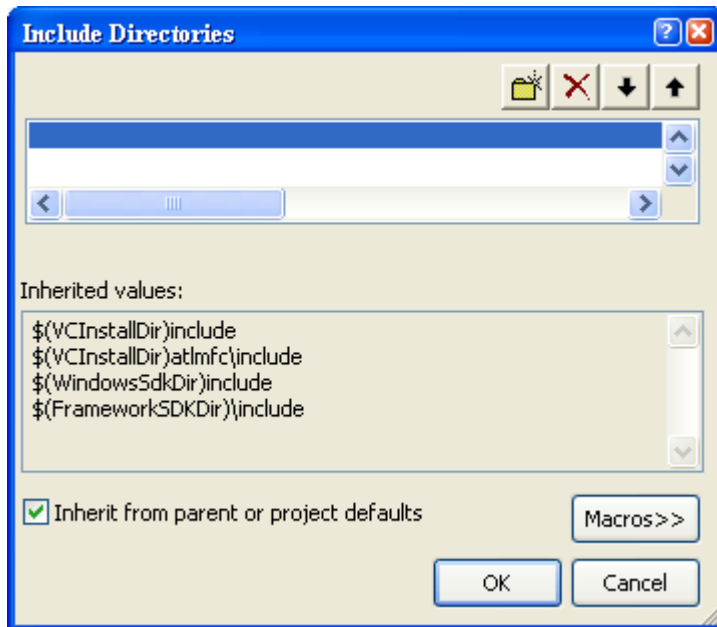


OPENCV VÀ VS2010

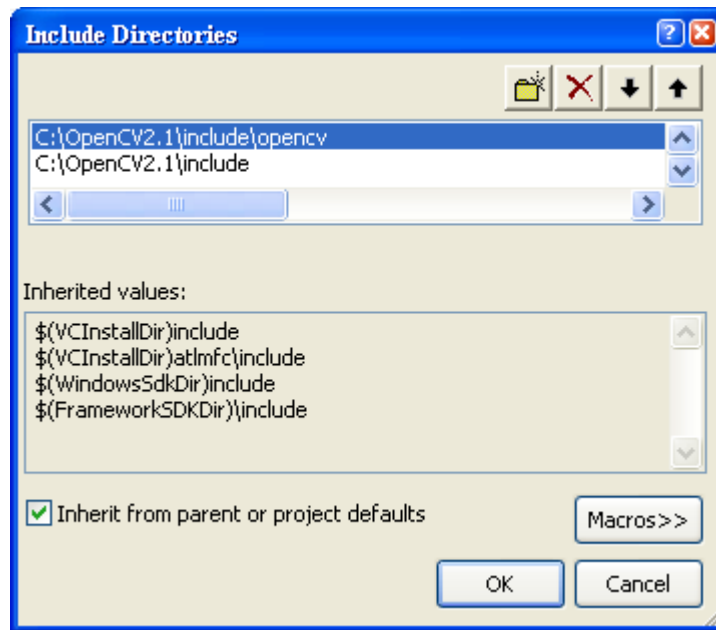
- Vào Project- vidu1 Properties – VC++Directories khai báo các đường dẫn



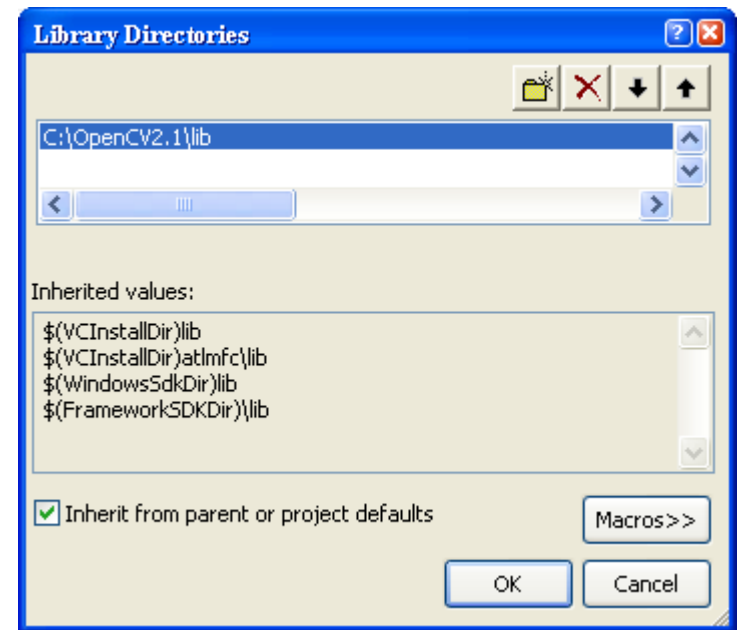
OPENCV VÀ VS2010



OPENCV VÀ VS2010

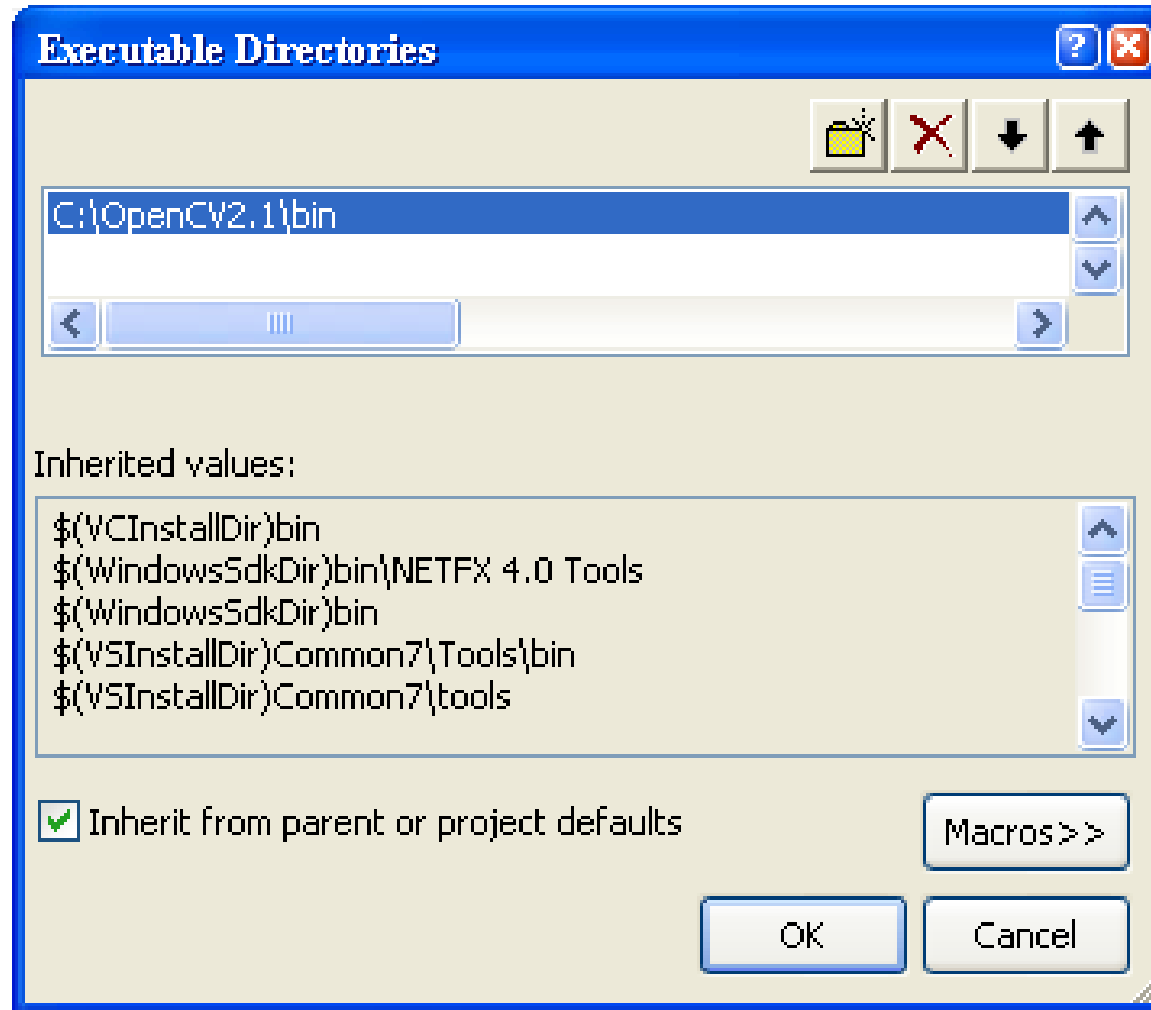


C:\opencv2.2\include
C:\\opencv2.2\\include\\opencv
C:\\opencv2.2\\include\\opencv2

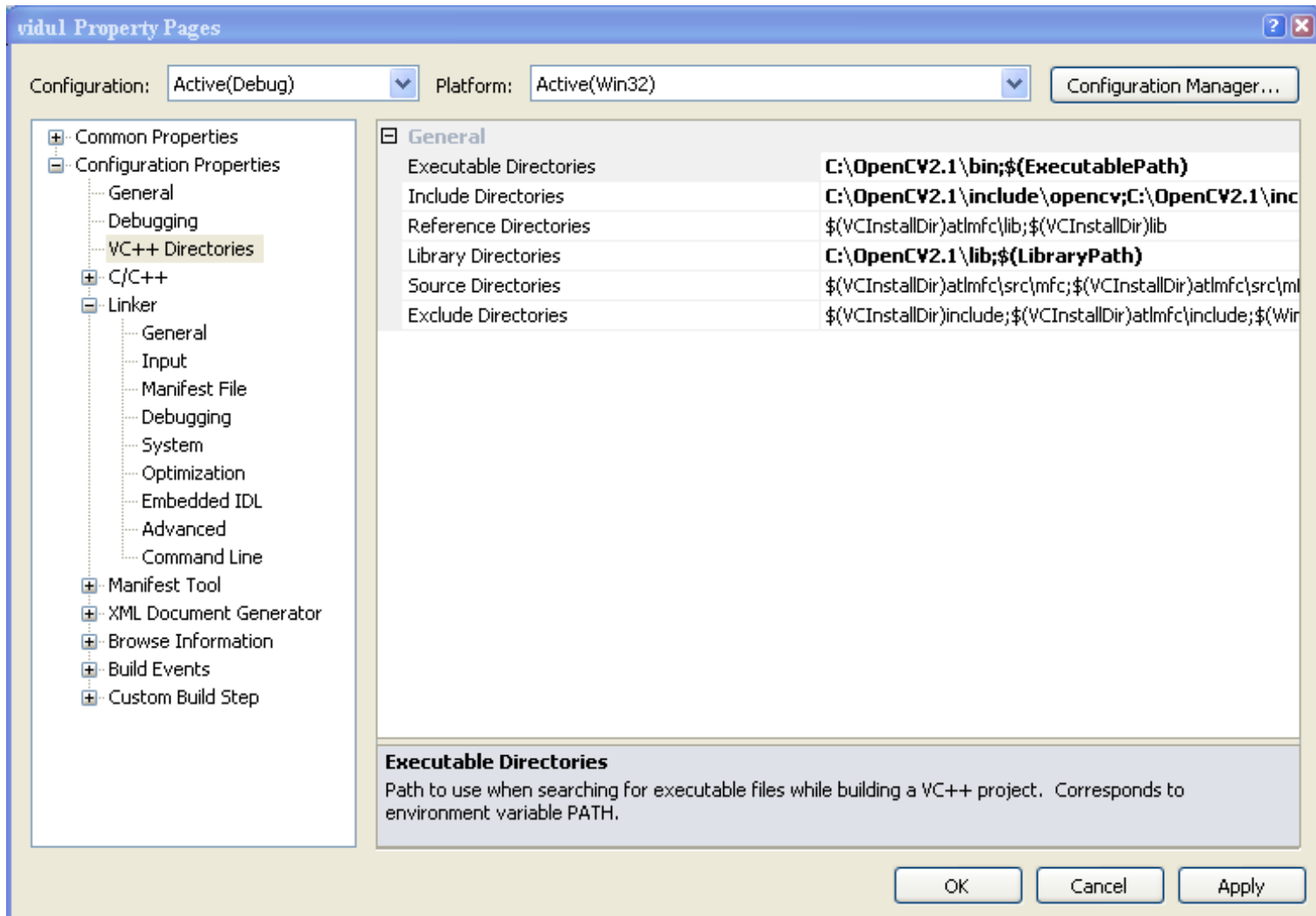


C:\opencv2.2\lib

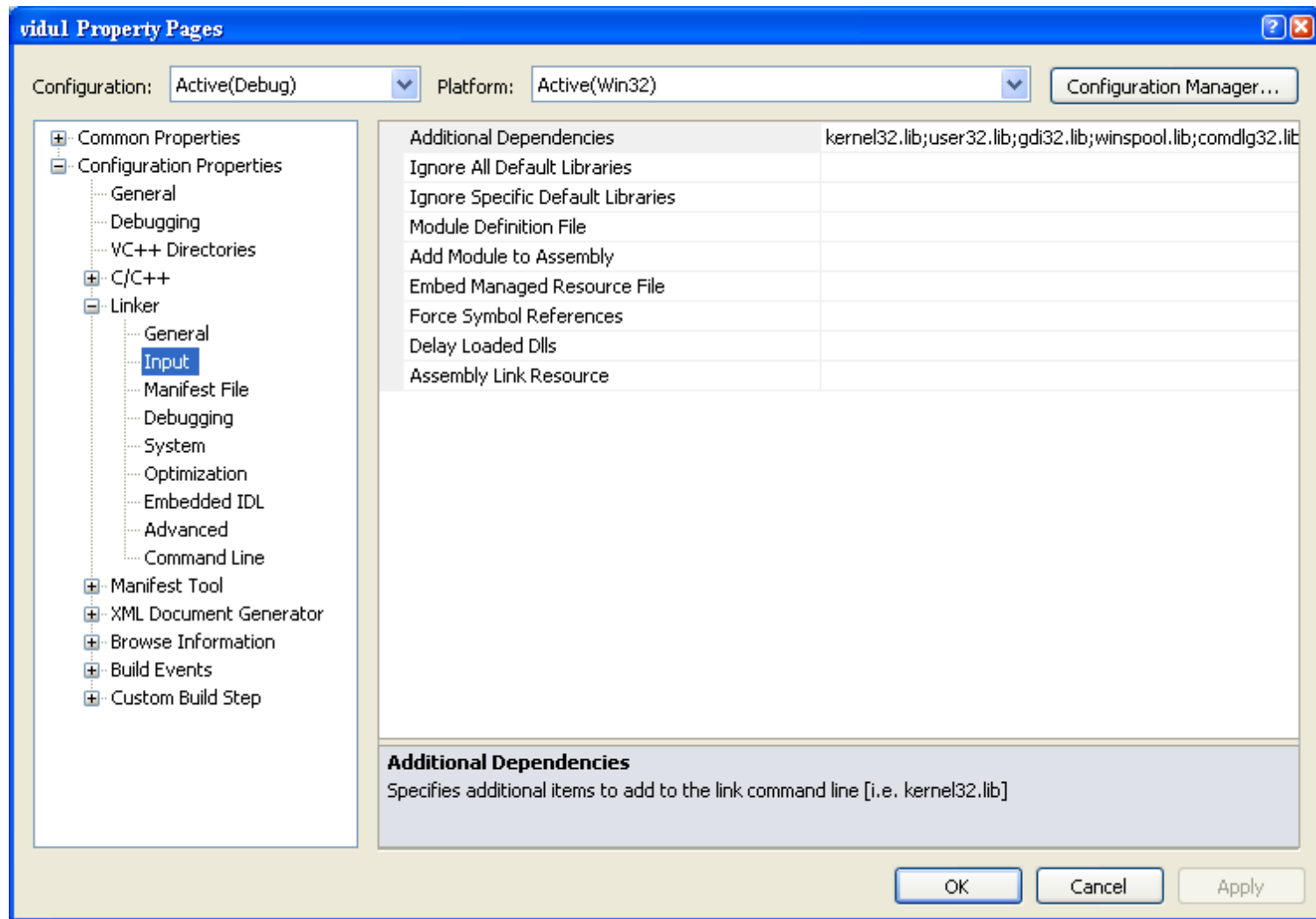
OPENCV VÀ VS2010



OPENCV VÀ VS2010

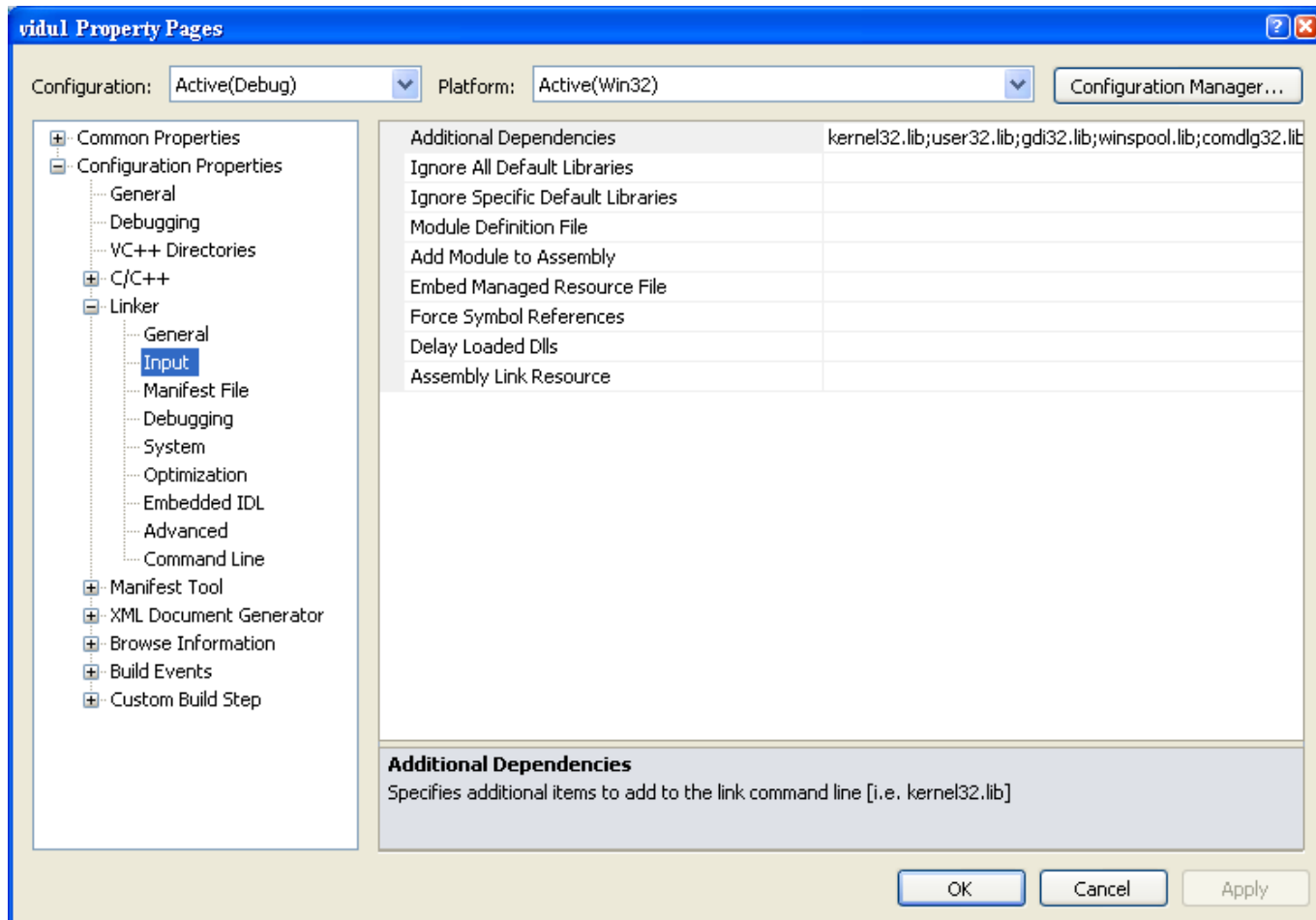


OPENCV VÀ VS2010

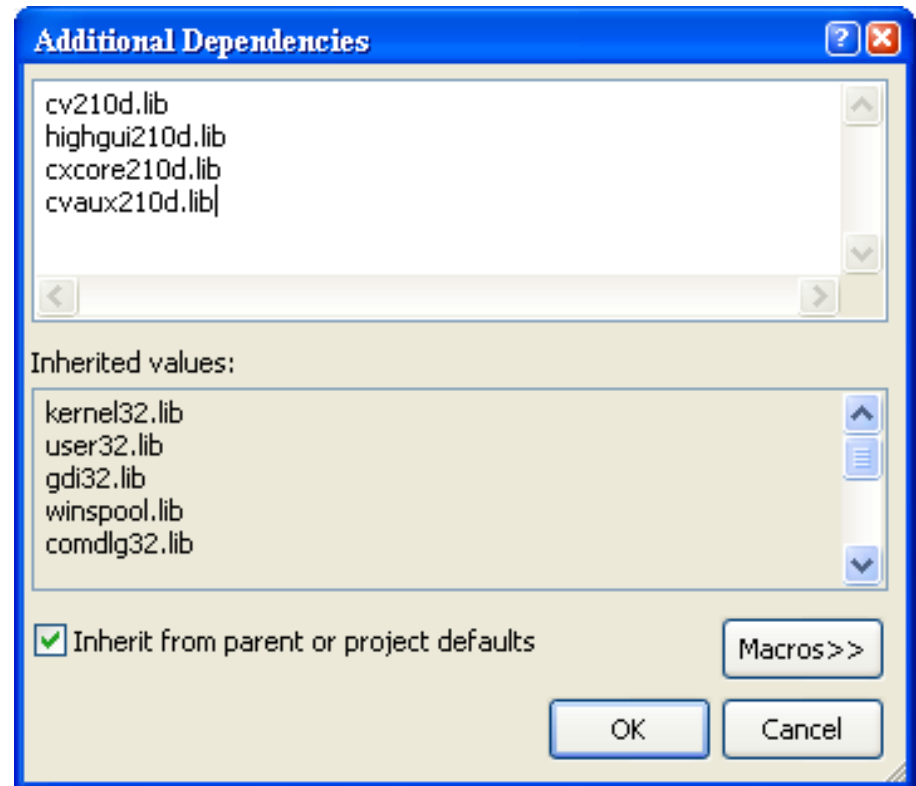
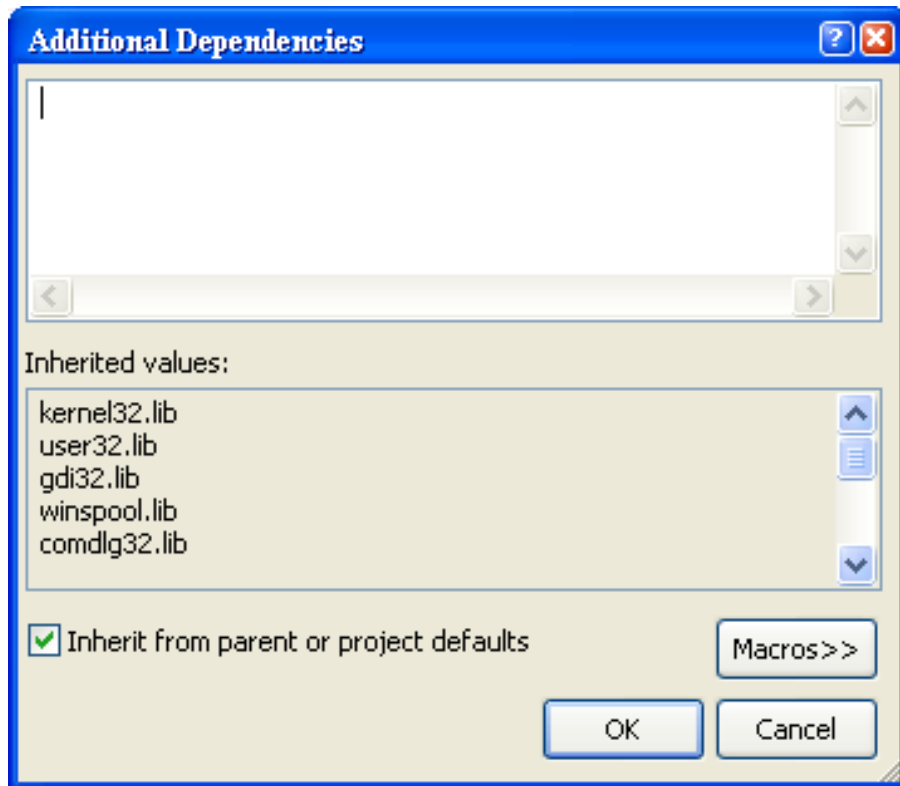


OPENCV VÀ VS2010

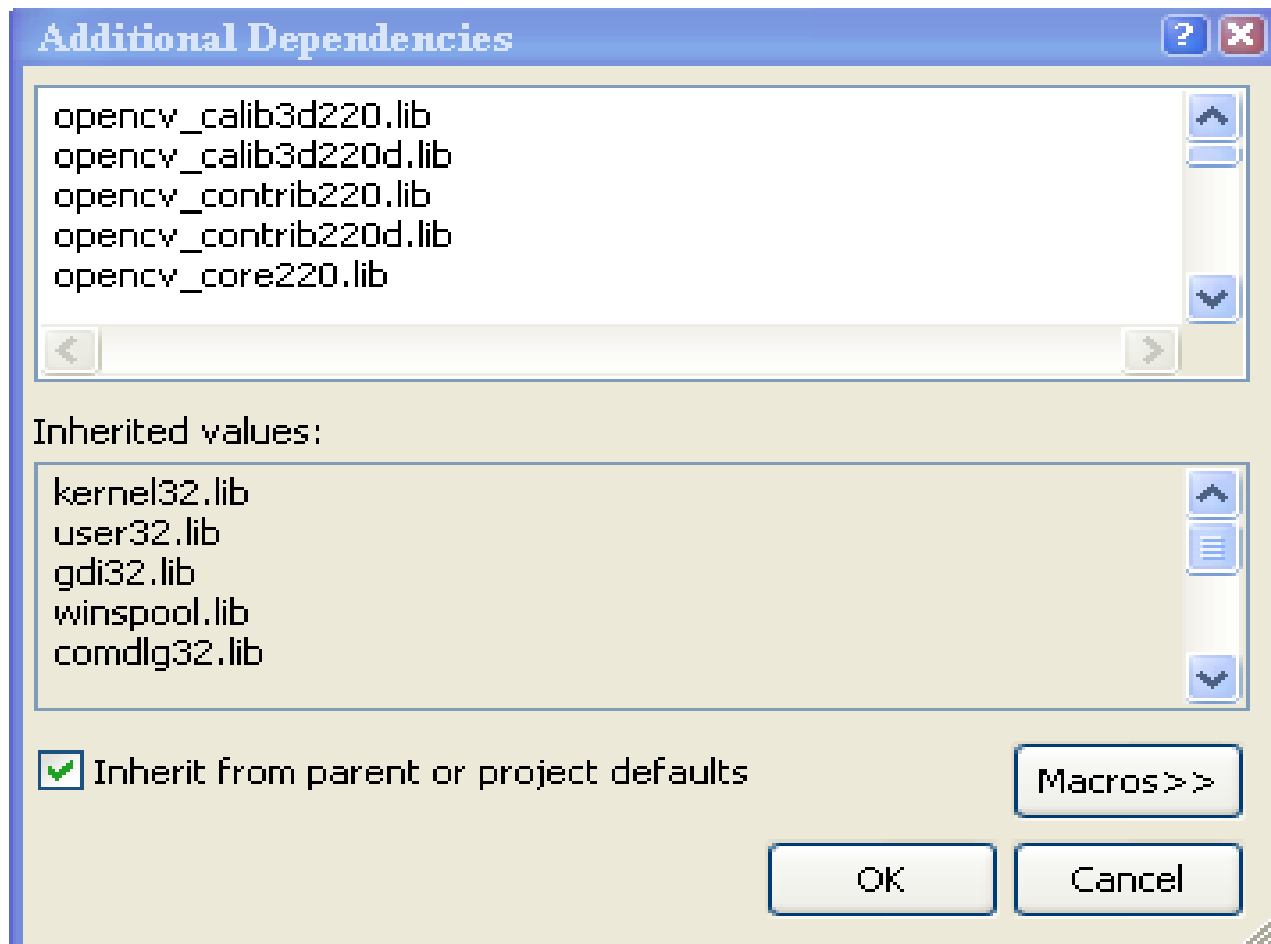
Vào Linker Input Additional Dependencies



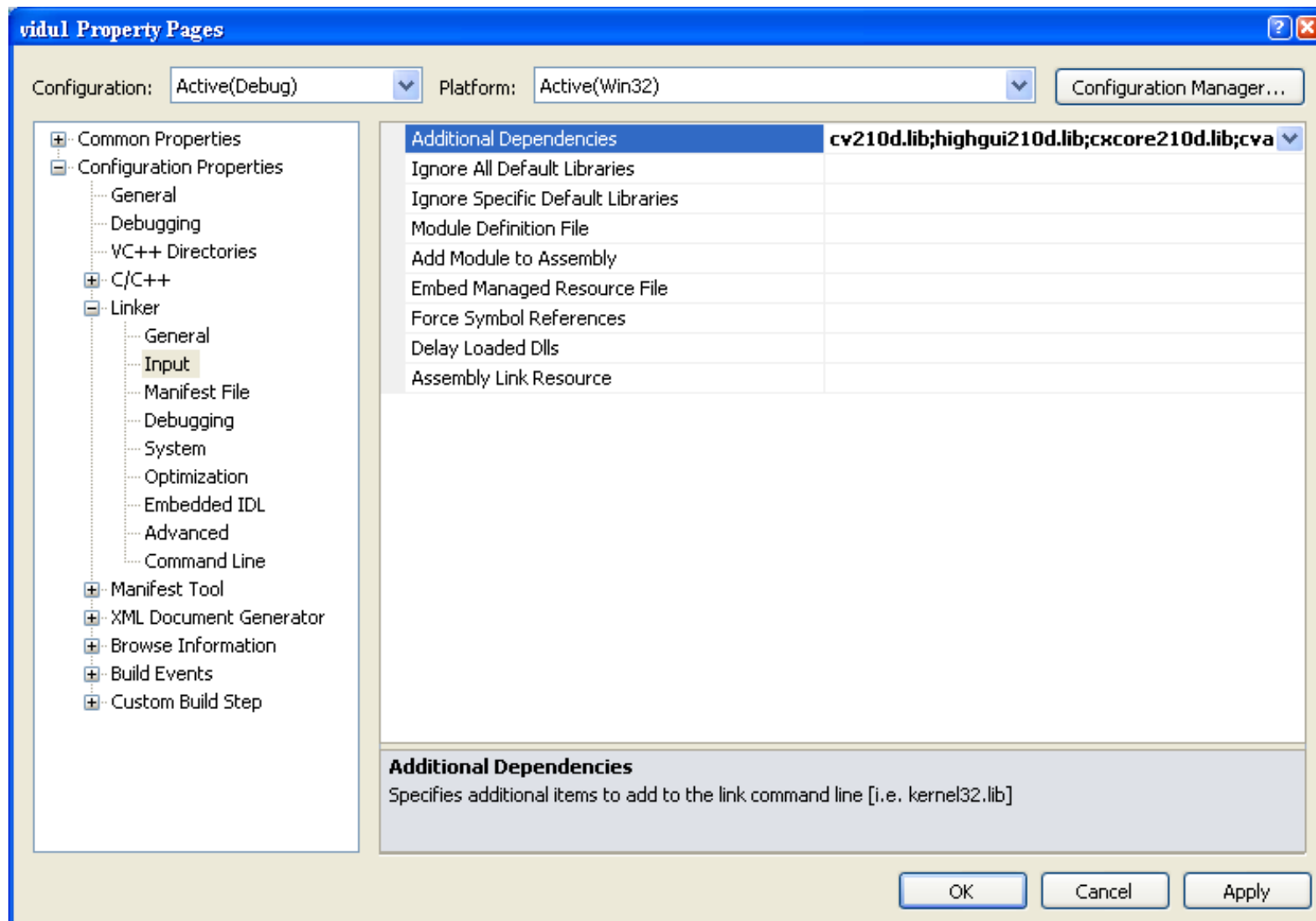
OPENCV VÀ VS2010



- Chép các tên file lib vào (dùng copyfilenames)
- opencv_calib3d220.lib opencv_calib3d220d.lib
 opencv_contrib220.lib opencv_contrib220d.lib opencv_core220.lib
 opencv_core220d.lib opencv_features2d220.lib
 opencv_features2d220d.lib opencv_ffmpeg220.lib
 opencv_ffmpeg220d.lib opencv_flann220.lib opencv_flann220d.lib
 opencv_gpu220.lib opencv_gpu220d.lib opencv_highgui220.lib
 opencv_highgui220d.lib opencv_imgproc220.lib
 opencv_imgproc220d.lib opencv_legacy220.lib
 opencv_legacy220d.lib opencv_ml220.lib opencv_ml220d.lib
 opencv_objdetect220.lib opencv_objdetect220d.lib opencv_ts220.lib
 opencv_video220.lib opencv_video220d.lib



OPENCV VÀ VS2010



- Chép tất cả file opencv có đuôi dll vào windows/system32
- Khai báo đường dẫn của bin và lib opencv trong biến môi trường path
- Ví dụ:
c:\opencv\build\sources\build\bin\debug\;c:\windows\system32\;c:\opencv\sources\build\lib\
- Vào Tools-Options-Native check Load DLL Experts

OPENCV VÀ VS2010



TS NGUYỄN ĐỨC THÀNH

OPENCV VÀ VS2010

Chép file sau vào chương trình

// vidu1.cpp :

```
#include "stdafx.h"
```

```
#include "cv.h"
```

```
#include "highgui.h"
```

```
int main( int argc, char** argv ) {
```

```
//file anh.jpg đặt ở ổ c, nếu không có khi debug sẽ báo assertion error
```

```
IplImage* img = cvLoadImage( "c:/anh.jpg" );
```

```
cvNamedWindow( "Example1", CV_WINDOW_AUTOSIZE );
```

```
cvShowImage( "Example1", img );
```

```
cvSaveImage(" c:/anh2.png",img); //cất dưới dạng khác
```

```
cvWaitKey(0);
```

```
cvReleaseImage( &img );
```

```
cvDestroyWindow( "Example1" );
```

```
}
```

```
//Build -Debug
```

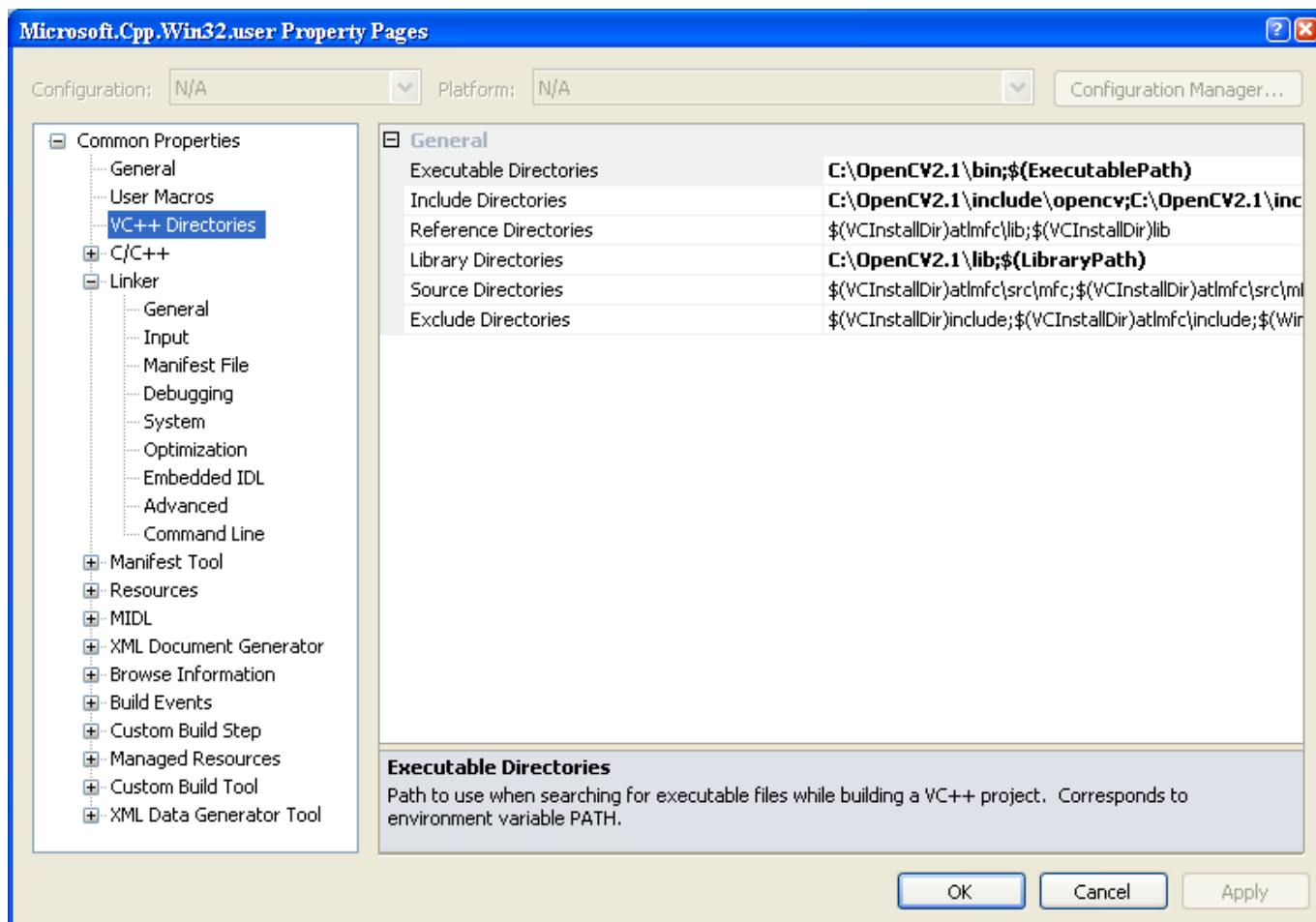
VC++

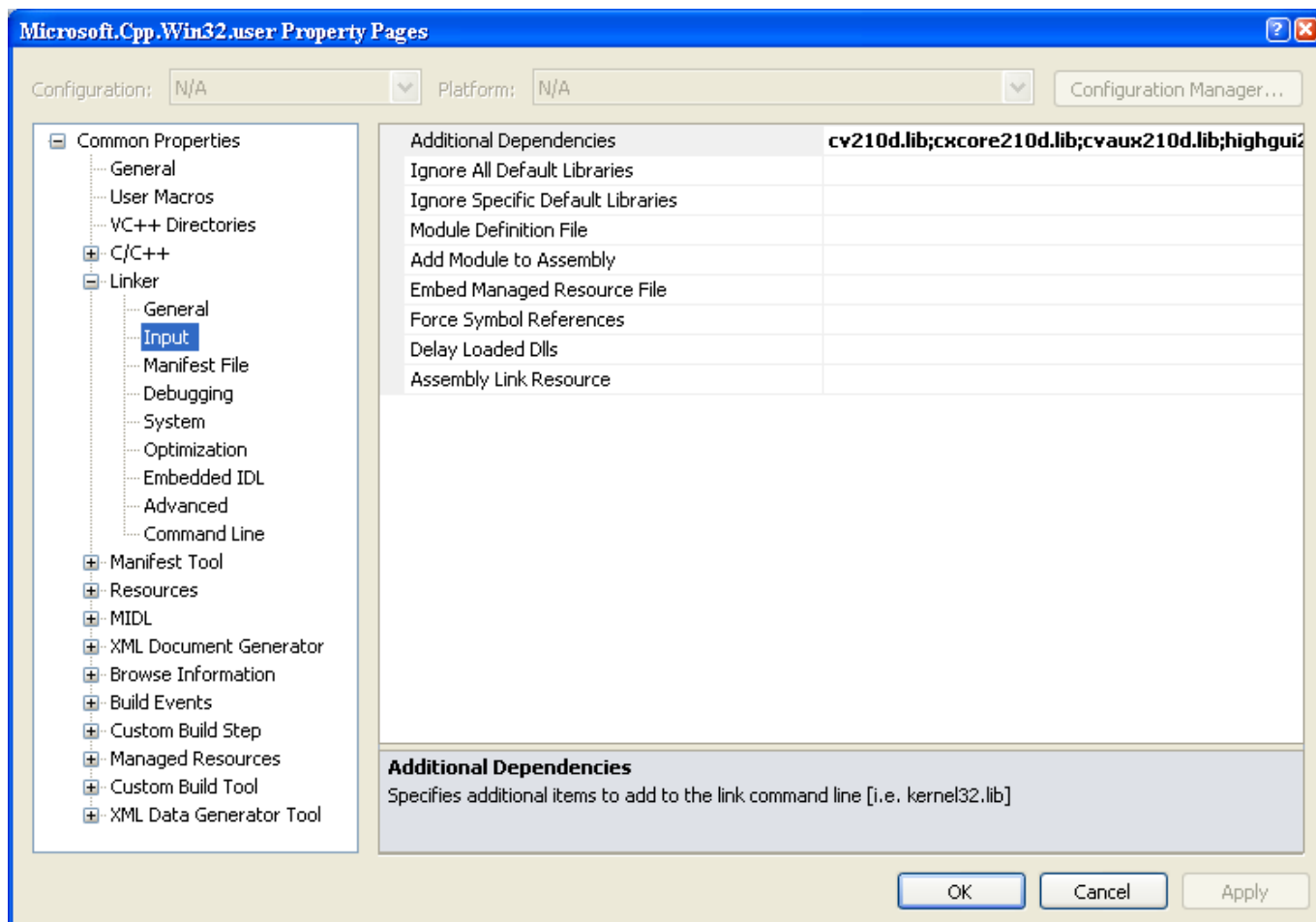
CÁCH VIẾT KHÁC

```
#include "opencv2/highgui/highgui.hpp"
#include <iostream>
using namespace cv; // để không cần gõ chữ cv:: ở các hàm liên quan
using namespace std; // để không cần gõ chữ std:: ở các hàm liên quan
int main( )
{
    Mat img = imread("c:/anh.JPG", CV_LOAD_IMAGE_UNCHANGED); //read the
    image data in the file "anh.JPG" and store it in 'img'
    if (img.empty()) //check whether the image is loaded or not
    {
        cout << "Error : Image cannot be loaded..!!" << endl;
        system("pause"); //wait for a key press
        return -1;
    }
    imshow("MyWindow", img); //display the image which is stored in the 'img' in the
    "MyWindow" window
    waitKey(0); //wait infinite time for a keypress
    destroyWindow("MyWindow"); //destroy the window with the name, "MyWindow"
    return 0;
}
```

INCLUDE

- Dòng chỉ dẫn tiền xử lý `#include` dùng để thêm các khai báo hay chương trình con đã viết ở nơi khác vào project.
- Có hai kiểu
- `#include <file>` *file* nằm trong các thư mục đã qui định sẵn
- `#include "file"` đầu tiên tìm *file* nằm trong cùng thư mục của project, sau đó tìm trong thư mục đã định
- `#include "stdafx.h"` phải là dòng đầu tiên





CÀI ĐẶT OPENCV 3.4 Prebuild VISUAL STUDIO PRO 2015 WIN 10 X64

DOWNLOAD

- Opencv 3.4.1_vc14_vc15.exe (opencv-win) chứa các lib và dll prebuild cho máy x64

https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.4.1/opencv-3.4.1-vc14_vc15.exe/download

- Tutorial

https://docs.opencv.org/3.4.1/d9/df8/tutorial_root.html

- Visual studio pro 2015

<https://tinhte.vn/threads/download-visual-studio-2015-full-huong-dan-cai-dat.2706953/>

CÀI ĐẶT VS2015

- Visual Studio 2015 sẽ cho phép lập trình viên phát triển ứng dụng đa nền tảng, từ Windows đến Linux, iOS và cả Android. Chạy tốt trên win 7, 8, 10
- Mount vs2015.iso rồi chạy vs_professional.exe
- Chọn kiểu cài custom, chọn cài đặt Visual C++
- Sau khi cài đặt thành công tạo một project trong console với chương trình sau để kiểm tra

```
#include "stdafx.h"
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{    cout << "Hello World! "; system("pause"); return 0;
```

```
}
```

CÀI ĐẶT OPENCV

- Chạy file opencv.exe vừa download, giải nén vào thư mục c:\opencv hoặc thư mục tùy ý, ghi nhớ đường dẫn
- Thư mục build\x64\vc14 chứa các file dll và lib
- Thư mục include chứa header

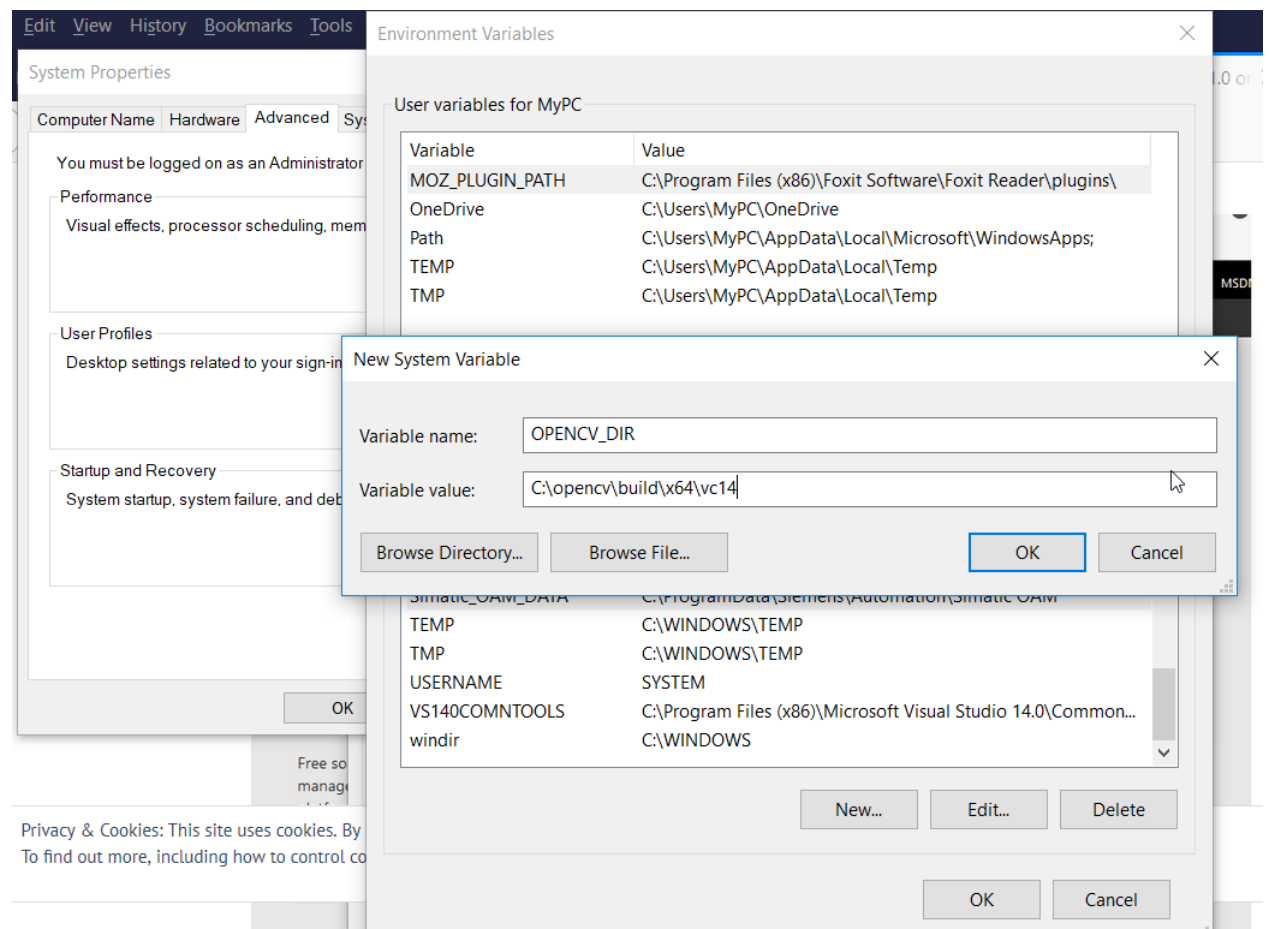
▼ c:\opencv*.*		
Name	↑Ext	Size
⬆ [..]		<DIR>
📁 [build]		<DIR>
📁 [opencv_contrib-3.4.1]		<DIR>
📁 [sources]		<DIR>
📄 LICENSE	txt	2.275
📄 LICENSE_FFMPEG	txt	27.913
📄 README.md	txt	684

▼ c:\opencv\build*.*		
Name	↑Ext	Size
⬆ [..]		<DIR>
📁 [bin]		<DIR>
📁 [etc]		<DIR>
📁 [include]		<DIR>
📁 [java]		<DIR>
📁 [python]		<DIR>
📁 [x64]		<DIR>
📄 LICENSE		2.275
📄 OpenCVConfig	cmake	5.674
📄 OpenCVConfig-version	cmake	433

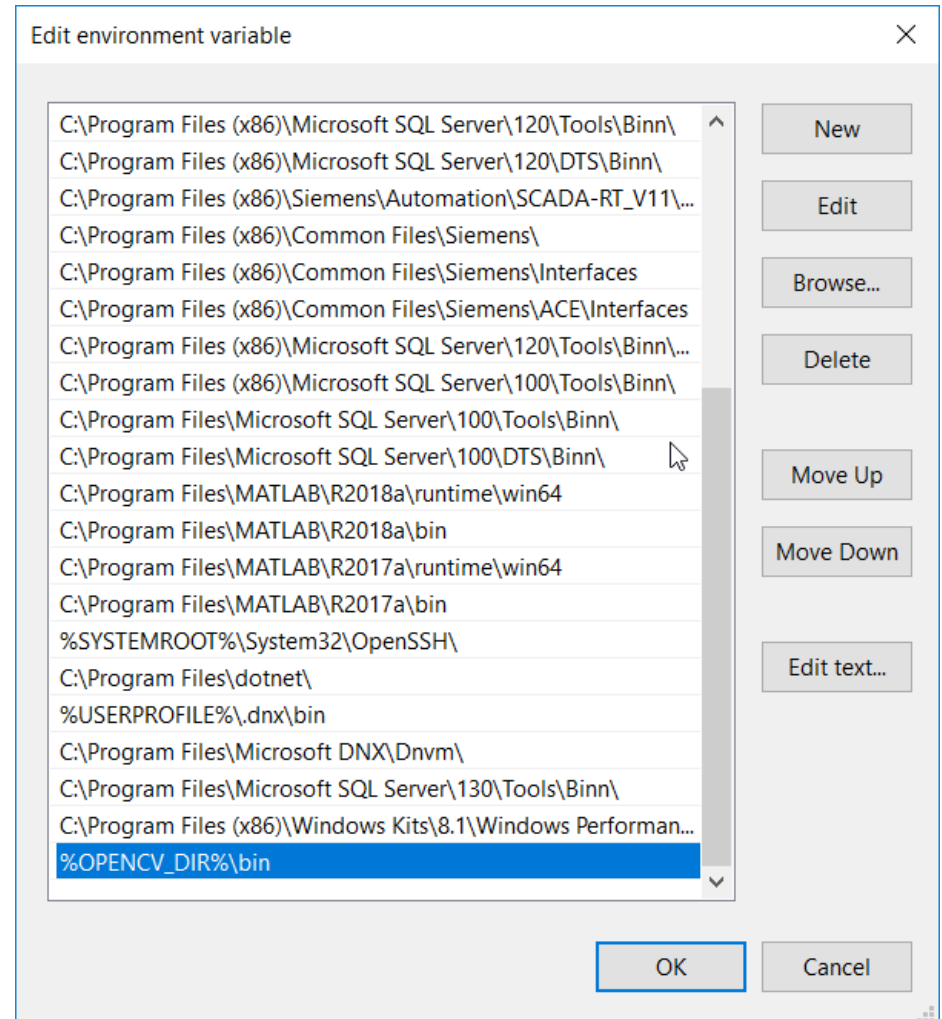
▼ c:\opencv\build\x64\vc14\bin*.*		
Name	↑Ext	Size
📁 [..]	<DIR>	
📁 opencv_ffmpeg341_64	dll	18.053.632
📁 opencv_world341	dll	65.811.968
📁 opencv_world341d	dll	104.591.872
📁 opencv_annotation	exe	53.760
📁 opencv_createsamples	exe	57.344
📁 opencv_interactive-calibra..	exe	139.776
📁 opencv_traincascade	exe	329.728
📁 opencv_version	exe	39.424
📁 opencv_visualisation	exe	61.440

▼ c:\opencv\build\x64\vc14\lib*.*		
Name	↑Ext	Size
📁 [..]	<DIR>	
📄 OpenCVConfig	cmake	13.925
📄 OpenCVConfig-version	cmake	433
📄 OpenCVMmodules	cmake	3.224
📄 OpenCVMmodules-debug	cmake	950
📄 OpenCVMmodules-release	cmake	956
📄 opencv_world341	lib	2.319.424
📄 opencv_world341d	lib	2.394.144

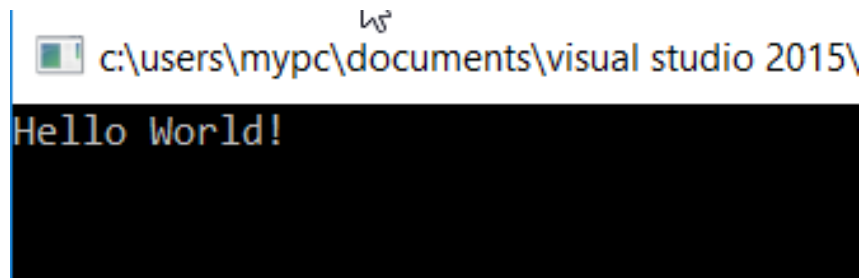
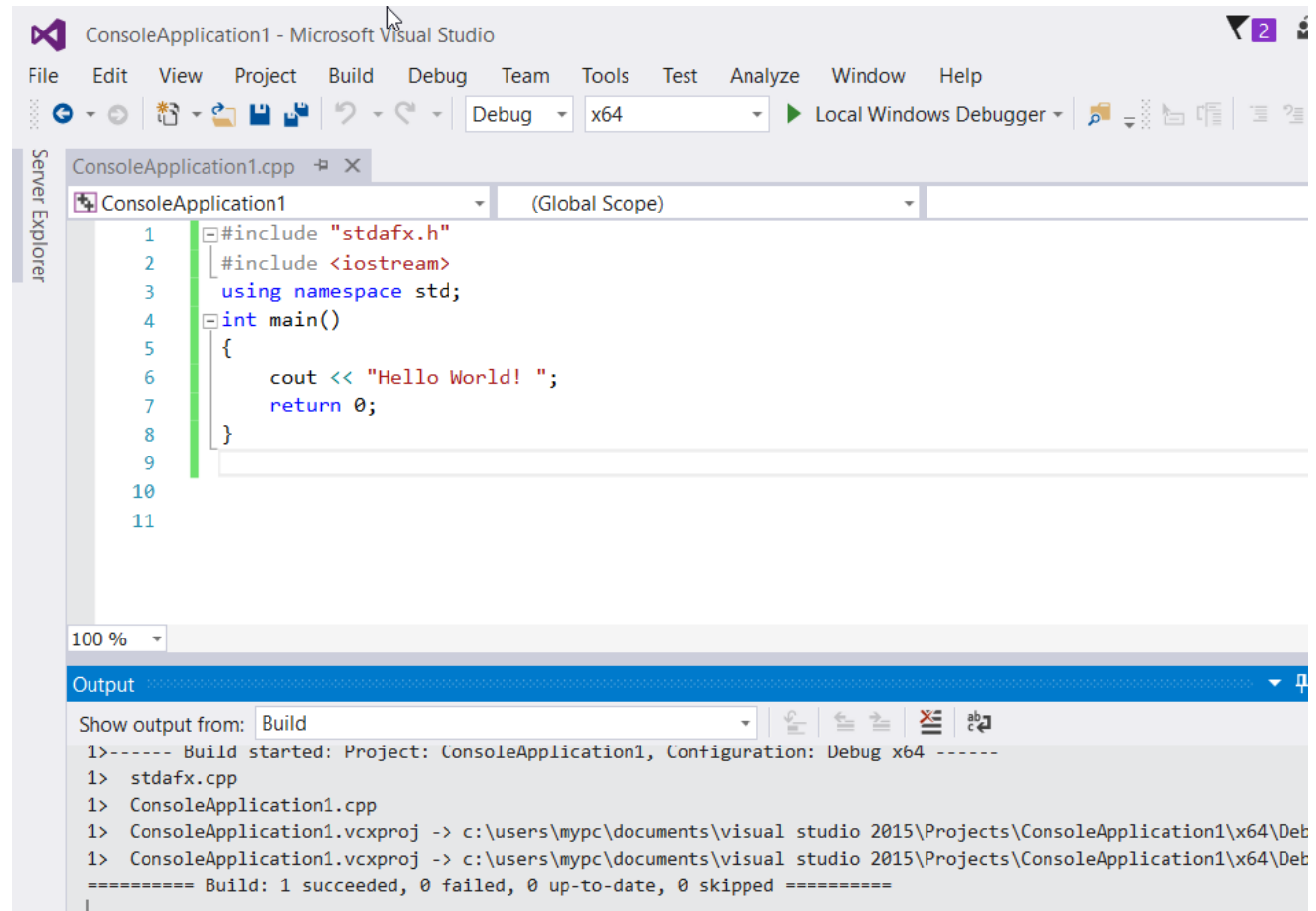
- Khai báo biến môi trường, control panel system advanced system settings Environment variables
- System Environment Variables New add OPENCV_DIR và đường dẫn C:\opencv\build\x64\vc14
- Bấm vào Path New thêm %OPENCV_DIR%\BIN để khai báo địa chỉ đặt file dll



Bấm OK OK OK để ra khỏi

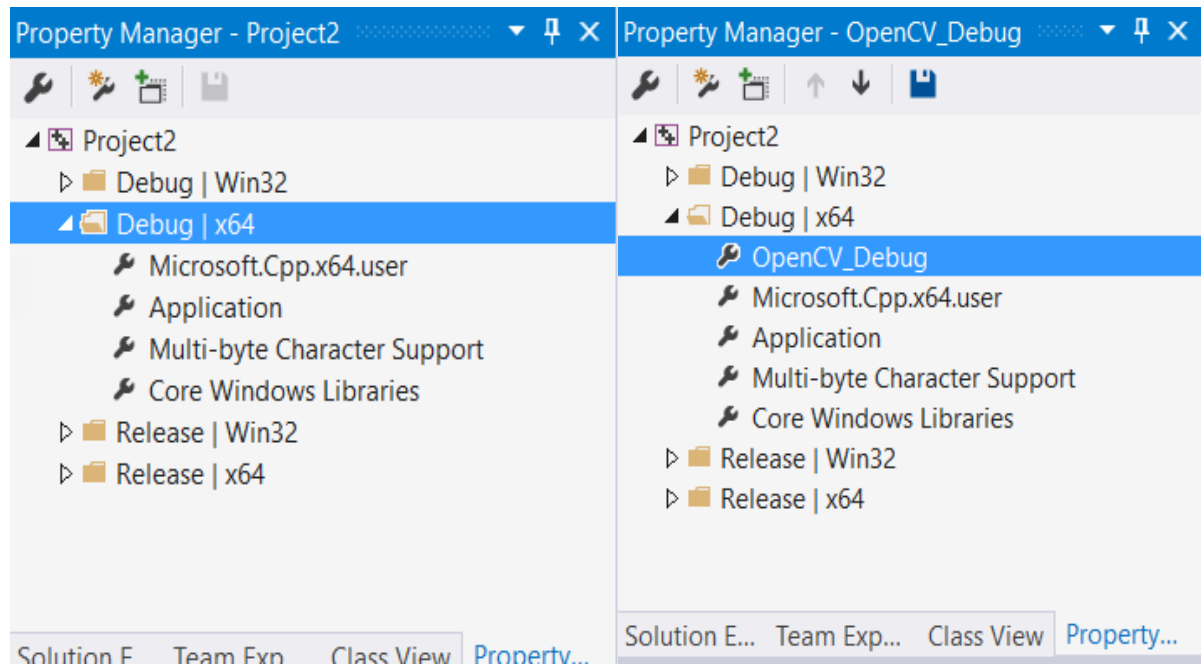


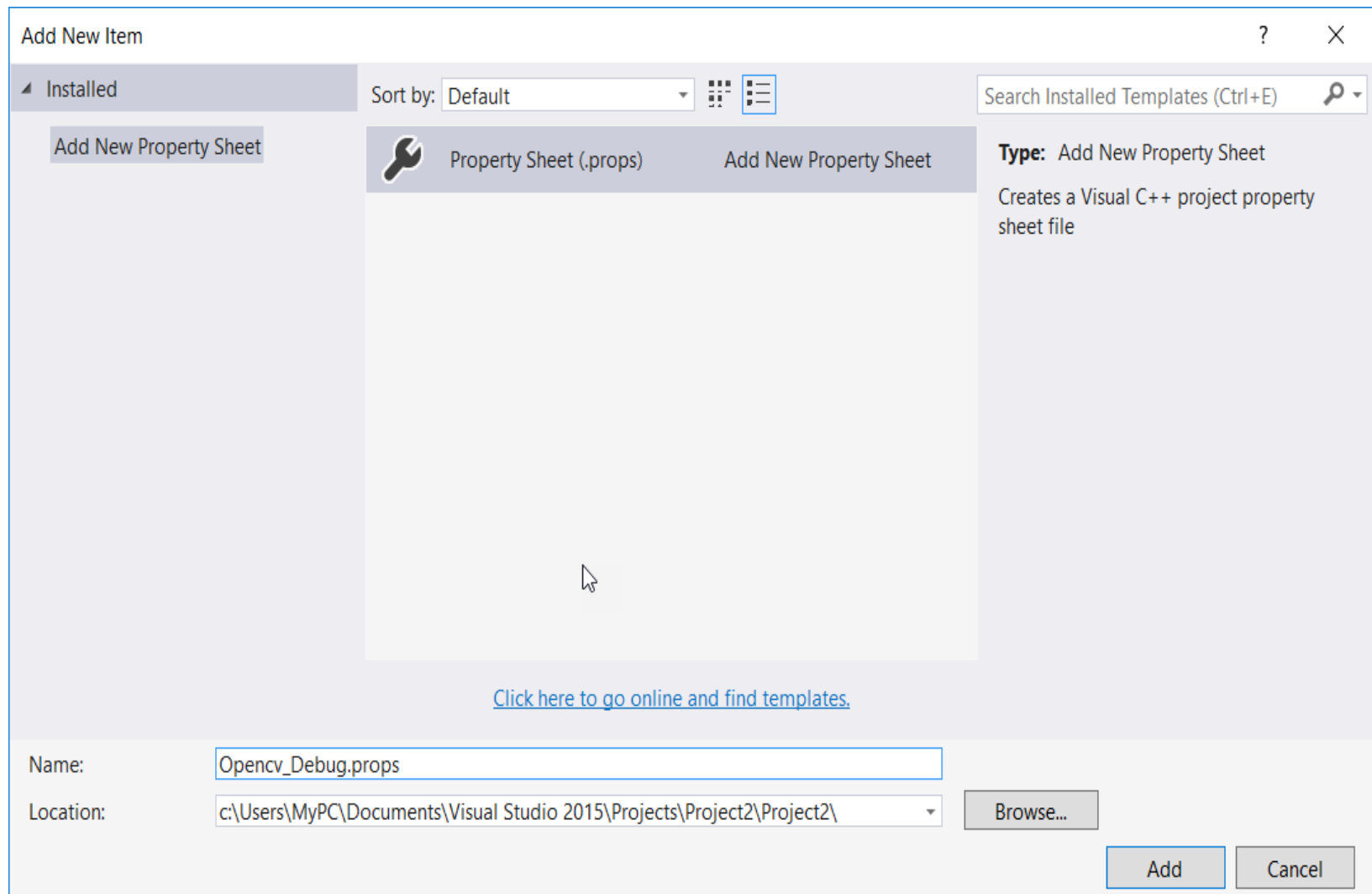
Build ở mode
Debug X64 rồi
bấm F5 hay Debug
Start Debugging,
chú ý chọn
Breakpoint ở hàng
7 để hiện màn hình
console



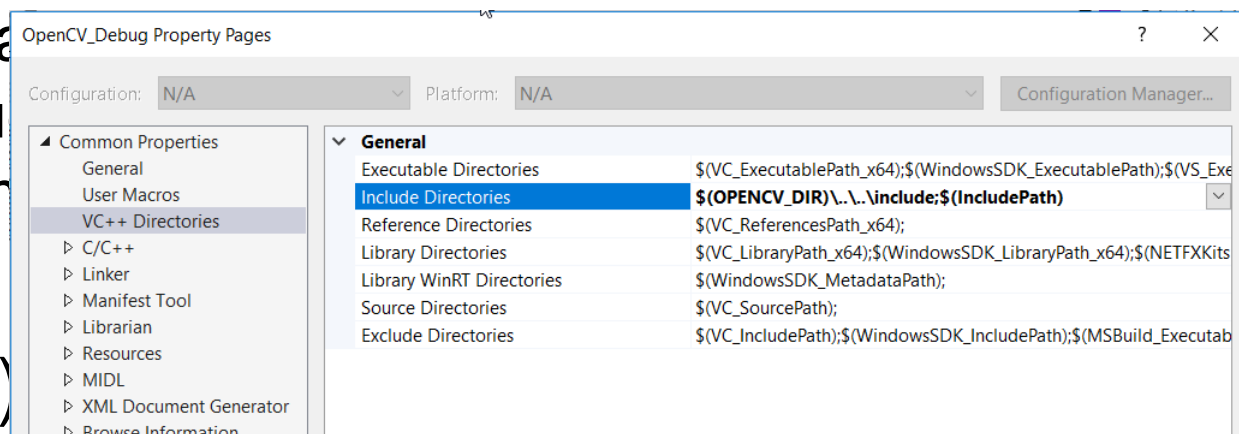
KHAI BÁO CÁC ĐƯỜNG DẪN CỦA OPENCV CHO VS2015

- Tạo Empty Project trong VC ++
- View Other Windows Property Manager
- Đặt cấu hình Debug X64
- Bấm chuột phải vào Debug (X64) chọn Add new Project Property Sheet thêm vào OpenCV_Debug.props





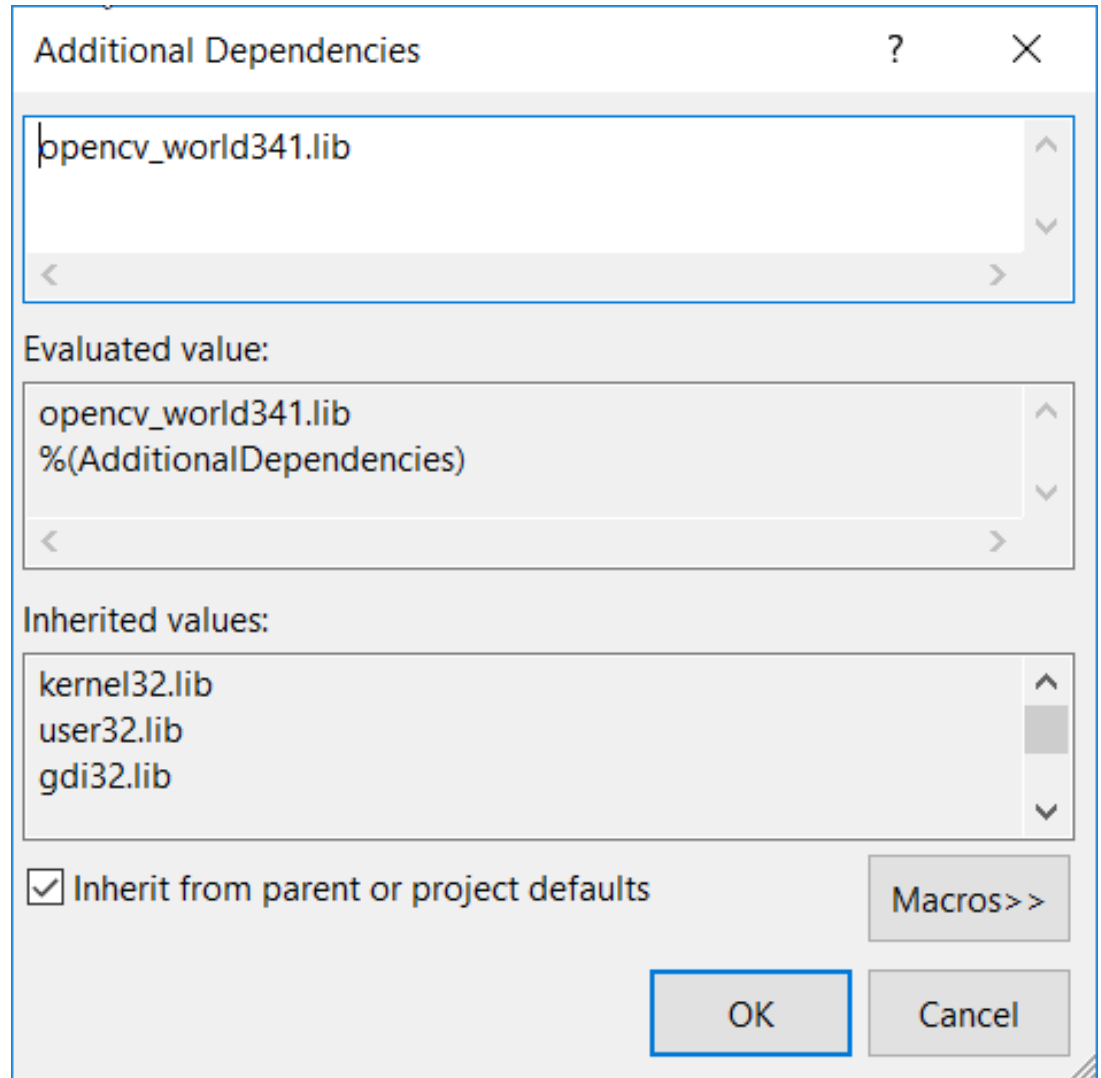
- Bấm chuột phải vào
Opencv_Debug,
chọn Properties
- C/C++->General
Additional Include
Directories thêm
đường dẫn
\$(OPENCV_DIR)
..\include



- Linker-
>General
Additional
Library
Directorie
s thêm
đường
dẫn thư
mục lib
\$(OPENCV_
DIR)\lib

<ul style="list-style-type: none"> Common Properties <ul style="list-style-type: none"> General User Macros VC++ Directories C/C++ Linker <ul style="list-style-type: none"> General Input Manifest File Debugging System Optimization Embedded IDL Windows Metadata Advanced 	<table> <tr> <td>Output File</td><td>\$(OutDir)\$(TargetName)</td></tr> <tr> <td>Show Progress</td><td>Not Set</td></tr> <tr> <td>Version</td><td></td></tr> <tr> <td>Enable Incremental Linking</td><td>Yes (/INCREMENTAL)</td></tr> <tr> <td>Suppress Startup Banner</td><td>Yes (/NOLOGO)</td></tr> <tr> <td>Ignore Import Library</td><td>No</td></tr> <tr> <td>Register Output</td><td>No</td></tr> <tr> <td>Per-user Redirection</td><td>No</td></tr> <tr> <td>Additional Library Directories</td><td>\$(OPENCV_DIR)\lib;</td></tr> <tr> <td>Link Library Dependencies</td><td>Yes</td></tr> <tr> <td>Use Library Dependency Inputs</td><td>No</td></tr> <tr> <td>Link Status</td><td></td></tr> <tr> <td>Prevent Dll Binding</td><td></td></tr> <tr> <td>Treat Linker Warning As Errors</td><td></td></tr> </table>	Output File	\$(OutDir)\$(TargetName)	Show Progress	Not Set	Version		Enable Incremental Linking	Yes (/INCREMENTAL)	Suppress Startup Banner	Yes (/NOLOGO)	Ignore Import Library	No	Register Output	No	Per-user Redirection	No	Additional Library Directories	\$(OPENCV_DIR)\lib;	Link Library Dependencies	Yes	Use Library Dependency Inputs	No	Link Status		Prevent Dll Binding		Treat Linker Warning As Errors	
Output File	\$(OutDir)\$(TargetName)																												
Show Progress	Not Set																												
Version																													
Enable Incremental Linking	Yes (/INCREMENTAL)																												
Suppress Startup Banner	Yes (/NOLOGO)																												
Ignore Import Library	No																												
Register Output	No																												
Per-user Redirection	No																												
Additional Library Directories	\$(OPENCV_DIR)\lib;																												
Link Library Dependencies	Yes																												
Use Library Dependency Inputs	No																												
Link Status																													
Prevent Dll Binding																													
Treat Linker Warning As Errors																													

- Linker->Input Additional Dependencies thêm tên file lib opencv_world341.lib
- Bấm OK OK



CHẠY CHƯƠNG TRÌNH TEST MỞ FILE ẢNH IMG.JPG

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main(int argc, char** argv)
{Mat image;
//image = imread(argv[1], IMREAD_COLOR); // Read the
file
```

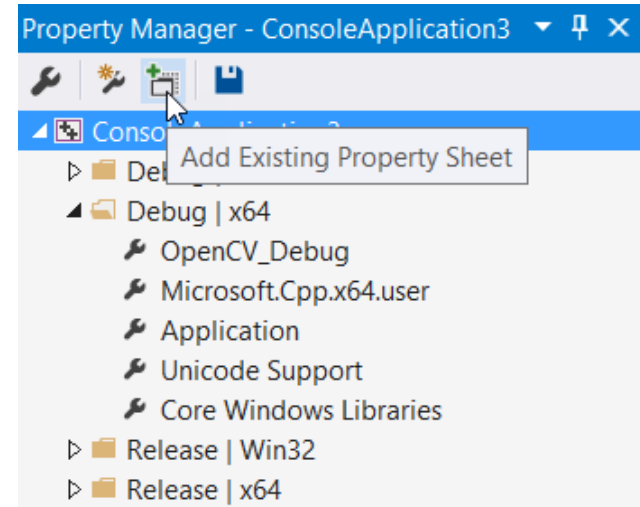
CHẠY CHƯƠNG TRÌNH TEST ĐỌC ẢNH

```
image = imread("c:\\jlo.jpg", IMREAD_COLOR); // Read the file
//hoặc imread("c:\\jlo.jpg");
//nếu muốn đổi sang ảnh xám dùng imread("c:\\jlo.jpg", 0);
if (!image.data) // Check for invalid input
{ cout << "Could not open or find the image" << std::endl;
return -1;}
namedWindow("Display window", WINDOW_AUTOSIZE);
//Create a window for display.
//hoặc namedWindow("Display window");
//namedWindow("Display window", 0); để chỉnh được kích
//thước windows
imshow("Display window", image); // Show our image inside it.
waitKey(0); // Wait for a keystroke in the window
return 0;
}
```


- Khi chương trình chạy tốt ta có thể lưu lại property sheet để dùng cho các project sau
- Bấm chuột phải vào Opencv_Debug chọn save sẽ lưu file property vào thư mục của project.
- Chép file Opencv_Debug.props vào một chỗ để nhớ
- Sau này khi tạo một project mới dùng opencv ta chỉ cần vào Property manager của project Add Existing Property Sheets
- Làm lại các bước trên ở chế độ release, lúc này dùng file opencv_world341.lib
- Chú ý Property Sheets chỉ lưu một số thông tin, khi dùng lại cho project khác ta cần kiểm tra lại

▼ \Documents\Visual Studio 2015\Projects\P

Name	Ext	Size
[.]		<DIR>
[Debug]		<DIR>
[x64]		<DIR>
Source	cpp	658
Project2.vcxproj	filters	958
Opencv_Debug	props	723
Project2.vcxproj	user	165
Project2	vcxproj	6.292



CHẠY CHƯƠNG TRÌNH TEST ĐỌC CAMERA

- Tạo project Camera Capture
- Vào Property manager Add existing property sheets chọn Opencv-Debug đã lưu
- Chọn chế độ debug x64 với máy cài Windows 64 bit
- Chép các dòng sau vào project

```
#include "stdafx.h"
#include "opencv2/opencv.hpp"
using namespace cv;
int main(int, char**)
{
    VideoCapture cap(0); // open the default camera
    if (!cap.isOpened()) // check if we succeeded
        return -1;

    Mat edges;
    Mat camera;
    namedWindow("camera", 1);
    namedWindow("edges", 1);
    for (;;)
    {
```

```

    Mat frame;
    cap >> frame; // get a new frame from camera
    imshow("camera", frame);
    cvtColor(frame, edges,
COLOR_BGR2GRAY);
    GaussianBlur(edges, edges, Size(7, 7), 1.5,
1.5);

    Canny(edges, edges, 0, 30, 3);
    imshow("edges", edges);
    if (waitKey(30) >= 0) break;
}

// the camera will be deinitialized automatically in
VideoCapture destructor
return 0;

}

```

HIỂN THỊ NHIỀU ẢNH CÙNG KÍCH THƯỚC TRÊN MỘT CỬA SỔ

- Dùng hàm `hconcat` (sắp xếp ngang) `vconcat` (sắp xếp dọc)

```
Mat im1, im2;
```

```
// source images im1 and im2
```

```
Mat newImage;
```

```
hconcat(im1, im2, newImage);
```

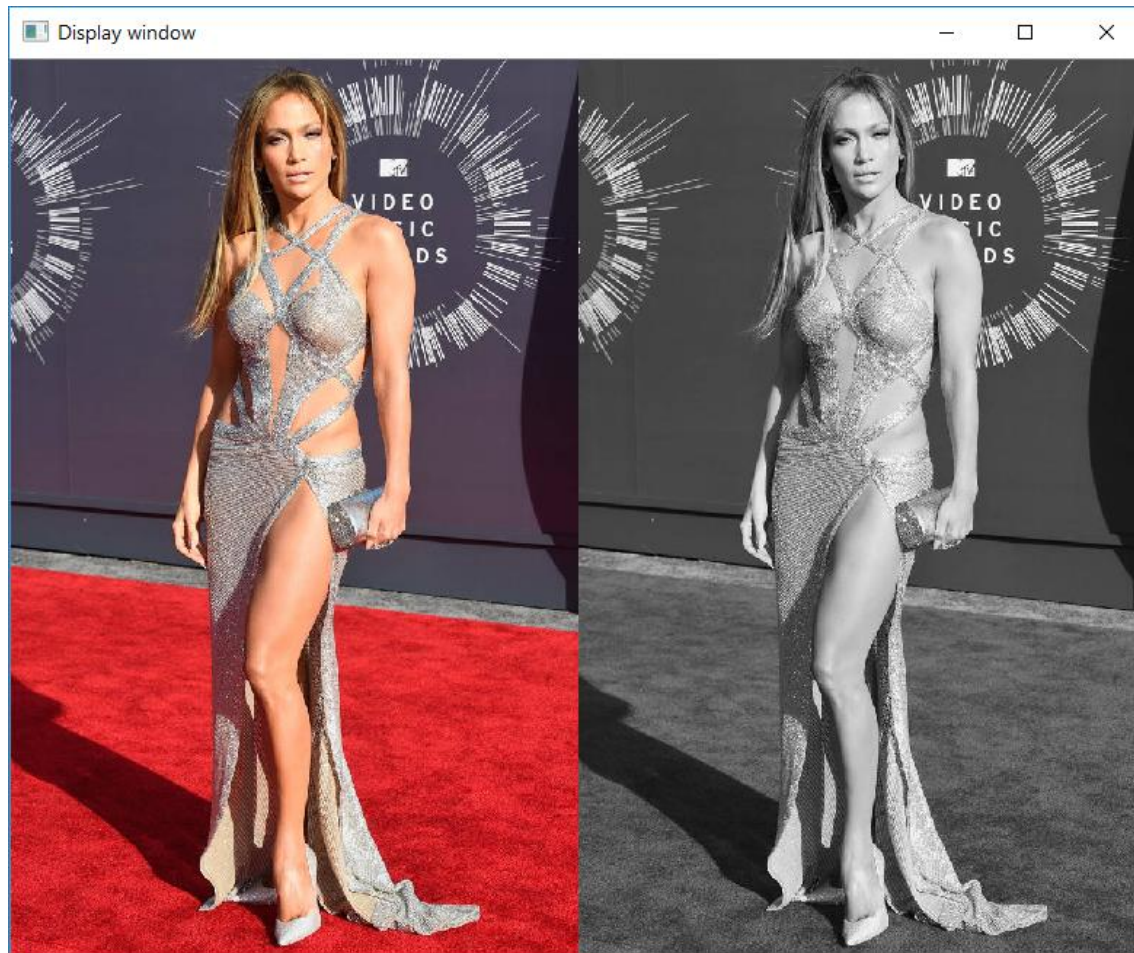
```
imshow("Display side by side", newImage);
```

```
waitKey(0);
```

HIỂN THỊ NHIỀU ẢNH CÙNG KÍCH THƯỚC TRÊN MỘT CỬA SỔ

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
using namespace cv;
int main() {
    Mat img, imgcolor, imggray;
    imgcolor = imread("c://jlo.jpg"); // no change color
    imggray = imread("c://jlo.jpg",0); //change to gray image
    cvtColor(imggray, imggray, CV_GRAY2BGR); //change to 3
    identical channel
    hconcat(imgcolor, imggray, img);
    namedWindow("Display window", 1);
    imshow("Display window", img); waitKey(0); return 0;}
```

HIỂN THỊ NHIỀU ẢNH CÙNG KÍCH THƯỚC TRÊN MỘT CỬA SỔ



Opencv 3.4.1 Python 2.7

- Python là ngôn ngữ lập trình đơn giản dễ kết hợp với opencv
- Download python 2.7.x và cài đặt vào c:\python2.7, vào Environment Variable path thêm đường dẫn c:\python2.7

<http://python.org/ftp/python/2.7.5/python-2.7.5.msi>

Cài đặt numpy 1.10.x for python 2.7

<https://sourceforge.net/projects/numpy/files/NumPy/1.10.1/>

Opencv 3.4.1 Python 2.7

- Cài đặt matplotlib

<https://sourceforge.net/projects/matplotlib/postdownload>

- Mở thư mục opencv\build\python\2.7\x86 chép cv2.pyd vào c:\python27\Lib\site-packages\
- Vào màn hình desktop chạy python 2.7 IDLE(Python GUI)
- Gõ >> import cv2, nếu không báo lỗi là thành công

Opencv 3.4.1 Python 2.7

- Viết chương trình test

http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html#display-image

- Dùng note pad edit chương trình cất dưới tên Test.py
- Mở Python IDLE File Open chọn file, sau đó bấm Run Run Module (F5) ảnh sẽ hiện ra, bấm Esc để tắt ảnh

Opencv 3.4.1 Python 2.7

```
import numpy as np
import cv2
img = cv2.imread('c:\messi5.jpg',0) # first arg is file
name,
cv2.imshow('image',img)
k = cv2.waitKey(0)
if k == 27:      # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
    cv2.imwrite('messigray.png',img)
cv2.destroyAllWindows()
```

Opencv 3.4.1 Python 2.7

74 Test.py - C:\Python27\Test.py

File Edit Format Run Options Windows Help

```
import numpy as np
import cv2

img = cv2.imread('c:\img.jpg',1)
cv2.imshow('image',img)
k = cv2.waitKey(0)
if k == 27:          # wait for ESC key to exit
    cv2.destroyAllWindows()
elif k == ord('s'): # wait for 's' key to save and exit
    cv2.imwrite('messigray.png',img)
    cv2.destroyAllWindows()
```

OPENCV PYTHON PLAY VIDEO

```
import numpy as np
import cv2
cap = cv2.VideoCapture('vtest.avi')
while(cap.isOpened()):
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Opencv Python Read camera

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Display the resulting frame
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Opencv Python Read camera

7 firstcamera.py - C:\Users\MyPC\Desktop\firstcamera.py

File Edit Format Run Options Windows Help

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame
    cv2.imshow('frame',frame)
    cv2.imshow('gray',gray)
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

OPENCV PYTHON SAVE VIDEO

```
import numpy as np
import cv2
cap = cv2.VideoCapture(0)
# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        out.write(frame)
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release();out.release();cv2.destroyAllWindows()
```


MULTIPLE IMAGE

```
import numpy as np; import cv2
imgcolor = cv2.imread('c:\dog.jpg',1) # do not change color,
imggray = cv2.imread('c:\dog.jpg',0) # convert to gray
imggray = cv2.cvtColor(imggray, cv2.COLOR_GRAY2BGR)
both = np.hstack((imgcolor,imggray))
#both = cv2.hconcat([imgcolor,imggray])
cv2.namedWindow('imgc',0)
#flag=0: user can change size of windows,flag=1:auto size of
windows
cv2.imshow('imgc',both)
k = cv2.waitKey(0)
if k == 27: cv2.destroyAllWindows()
```

THƯ VIỆN PILLOW

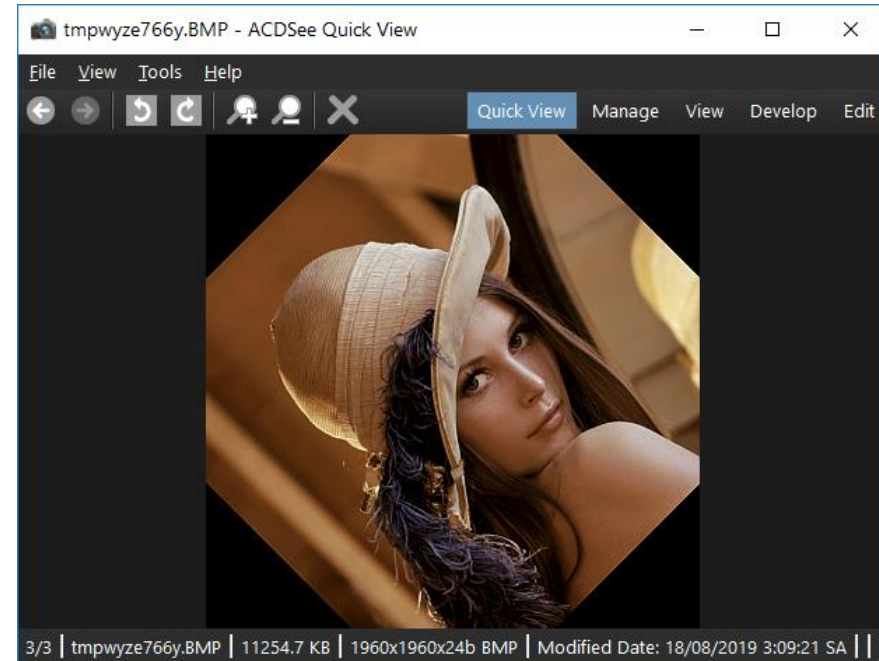
- Thư viện opencv có thể khá nặng với máy tính nhúng.
- Một cách khác đọc ảnh là dùng thư viện ảnh PIL python imaging library hay dùng Matplotlib
- Vào command windows gõ pip install Pillow

#use PIL

```
from PIL import Image
```

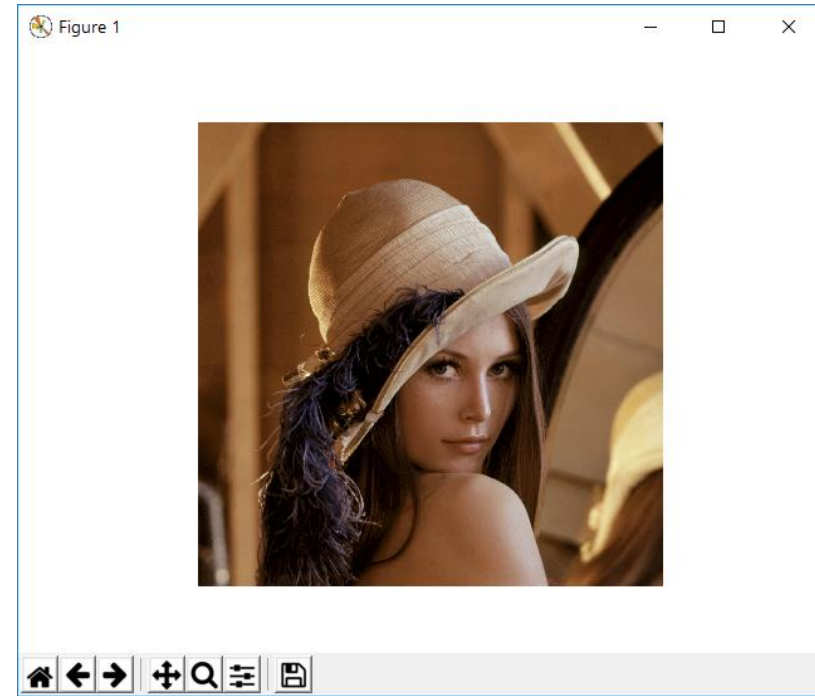
```
im = Image.open("c:/lenna.jpg")
```

```
im.rotate(45).show()
```



THƯ VIỆN MATPLOTLIB

```
#use matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
#loads image as a multi-
#dimensional NumPy array
img=mpimg.imread("c:/lenna.jpg")
plt.axis("off")
plt.imshow(img)
plt.show()
```



ĐỌC CAMERA KHÔNG DÙNG OPENCV

- Nếu muốn giao tiếp camera không dùng opencv ta có thể sử dụng thư viện pygame, matplotlib, PIL...
- Cài đặt pygame: `pip install pygame`
- Cài đặt videocapture expansion python: vào địa chỉ <https://www.lfd.uci.edu/~gohlke/pythonlibs/#videocapture> download file videocapture-0.9.5-cp37-cp37m-win_amd64.whl nếu dùng python 3.7 win 10 lưu vào ổ đĩa, ví dụ ổ C.
- Cài đặt VideoCapture: `pip install c:\videocapture-0.9.5-cp37-cp37m-win_amd64.whl`
- Vào thư mục cài python37/Libs/site-packages/VideoCapture/ mở file `_init_.py` sửa dòng `im = Image.fromstrings` thành `im = Image.frombytes`

OPENCV 3 PYTHON 3

- Nếu muốn dùng Python 3.6 với opencv 3.x có thể tham khảo tài liệu sau

<https://solarianprogrammer.com/2016/09/17/install-opencv-3-with-python-3-on-windows/>

DÙNG VIDEOCAMERA VÀ MATPLOTLIB

```
from VideoCapture import Device
from numpy import *
import matplotlib.pyplot as plt
import msvcrt
cam = Device(0, 0)
plt.ion();plt.axis("off")
while 1:
    if msvcrt.kbhit():
        msvcrt.getch()
        break
    image=cam.getImage();imgarray = asarray(image)
    img = imgarray.astype("uint8")
    plt.imshow(img)
    plt.pause(0.001)
    plt.show()
plt.close('all');plt.ioff()
```

CÀI ĐẶT PYTHON WIN 64 BIT

Python 3.6.7

<https://www.python.org/downloads/>

[Windows x86-64 executable installer](#)

Vào trang web <https://www.lfd.uci.edu/~gohlke/pythonlibs/>
download numpy-1.14.6+mkl-cp36-cp36m-win_amd64.whl
opencv_python-3.4.3+contrib-cp36-cp36m-win_amd64.whl
matplotlib-2.2.3-cp36-cp36m-win_amd64.whl

Cài đặt:

-Numpy: pip install "numpy-1.14.6+mkl-cp36-cp36m-win_amd64.whl"

-Opencv: pip install "opencv_python-3.4.3+contrib-cp36-cp36m-win_amd64.whl"

-Matplotlib pip install "matplotlib-2.2.3-cp36-cp36m-win_amd64.whl"

OPENCV PYTHON UBUNTU

- Ubuntu (tiếng zulu có nghĩa là tình người) là hệ điều hành mở dựa trên Linux
- Ubuntu không tốn nhiều dung lượng đĩa cứng, chừng 10GB là đủ
- Có thể cài đặt Ubuntu song song windows hay dùng máy ảo Vmware
- <https://anonyviet.com/huong-dan-cai-ubuntu-song-song-voi-windows-moi-nhat/>

- Sau khi cài xong ubuntu ta khởi động máy và vào ubuntu
- Cài đặt python
- Cài đặt opencv

Android OpenCV

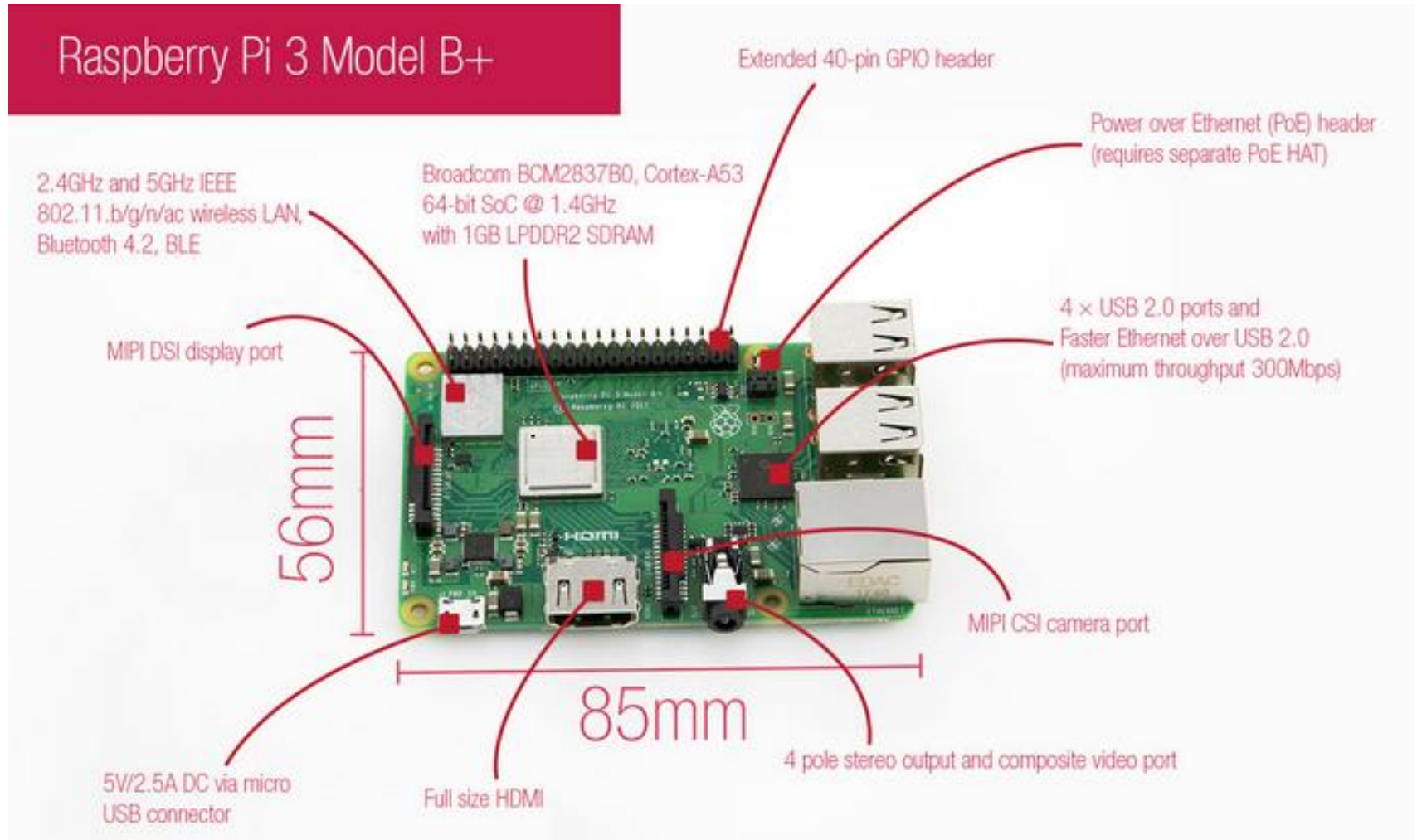
- Nếu muốn tạo ứng dụng thị giác máy trên thiết bị di động Android ta cần cài đặt Android Studio, Opencv4Android trên máy tính
- Tham khảo link <https://o7planning.org/vi/10405/huong-dan-cai-dat-va-cau-hinh-android-studio>
- <https://o7planning.org/vi/10415/huong-dan-lap-trinh-android-cho-nguoi-moi-bat-dau-hello-android>
- <https://viblo.asia/p/cai-dat-va-vi-du-minh-hoa-su-dung-opencv-trong-android-DKBedwNjGdX>
- <https://blog.codeonion.com/tutorials/opencv-for-android-tutorials/>

Opencv Máy tính nhúng

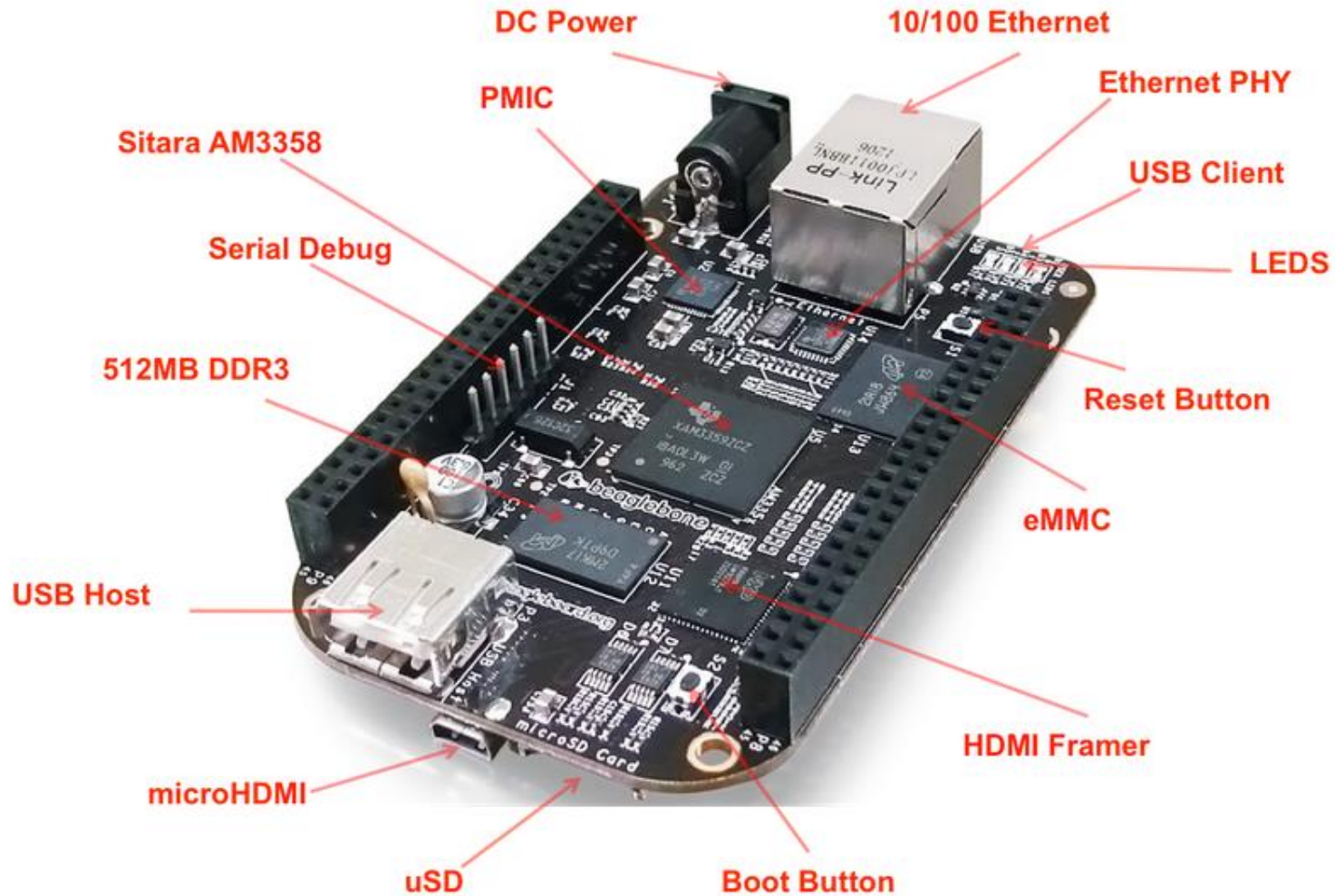
- Máy tính nhúng thường dùng là Raspberry Pi hay Beagle Bone hoặc có trang bị card đồ họa như NVIDIA Jetson TK1 developer kit .
- Các hệ điều hành cho máy tính nhúng đều dựa trên Linux với các biến thể khác nhau như Raspbian, Ubuntu, Android.
- Tham khảo trên mạng cách cài đặt hệ điều hành, Python và OpenCV

Raspberry Pi 3

Raspberry Pi 3 Model B+



BeagleBone Black

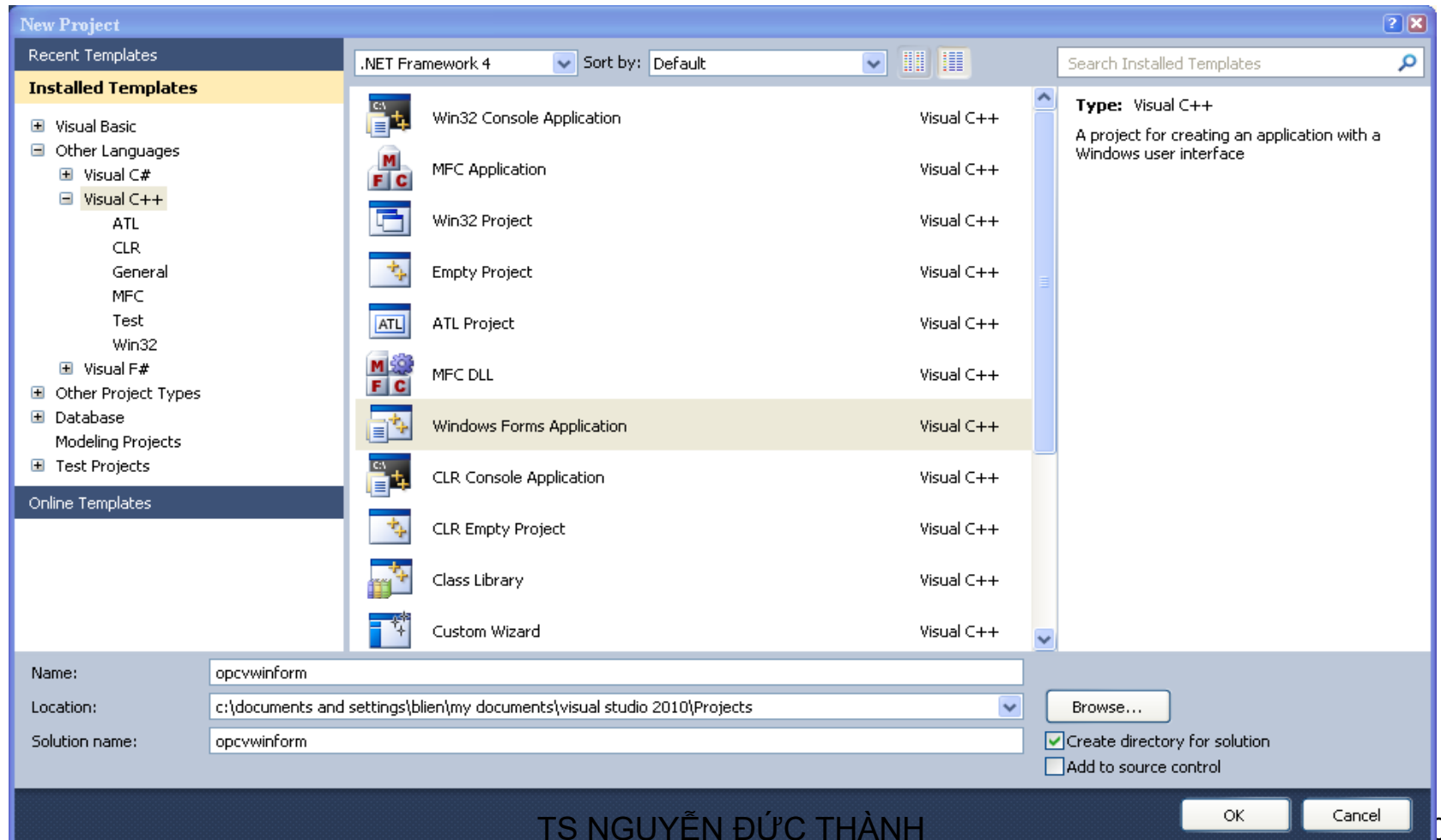


NVIDIA Jetson TK1 developer kit



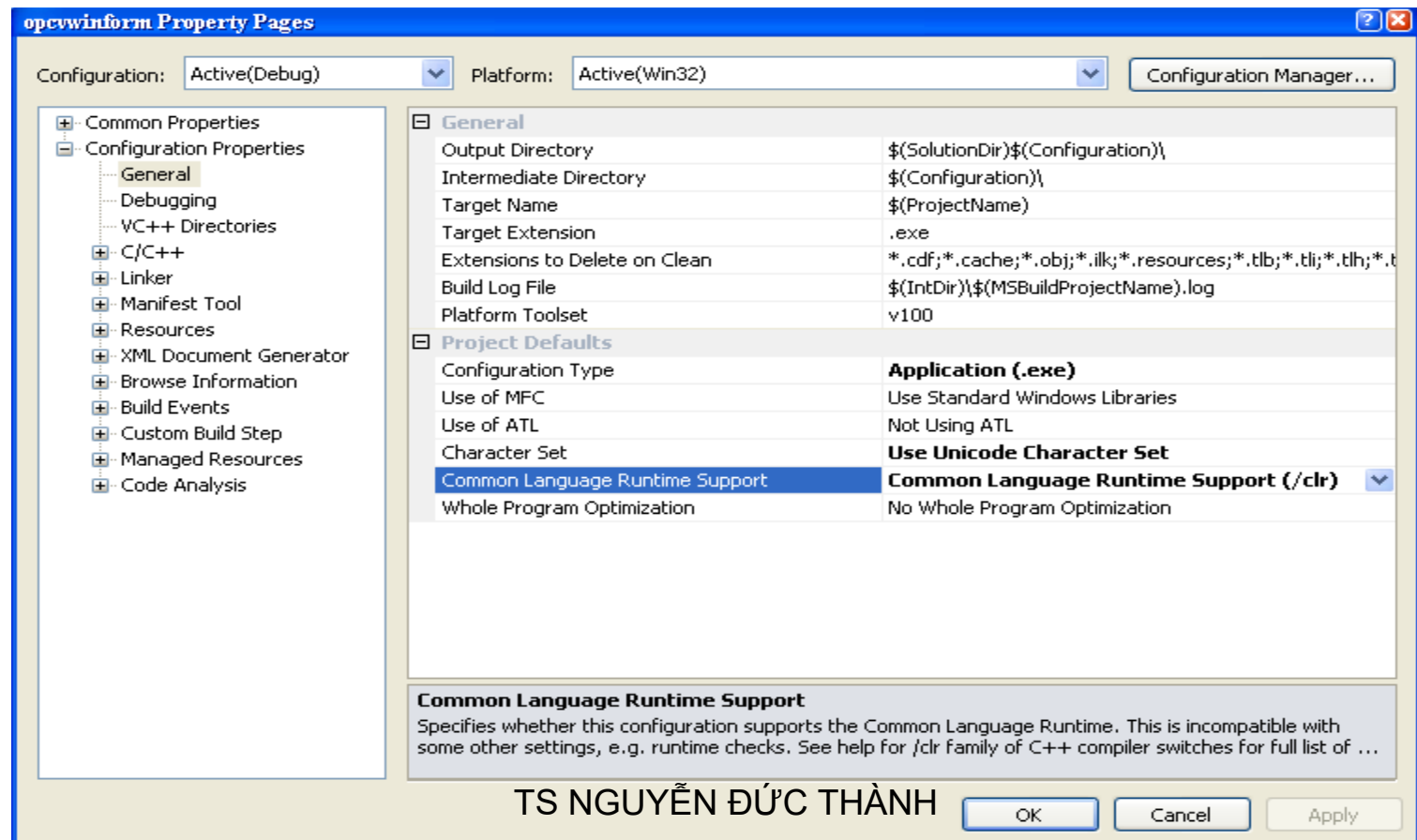
SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

- Tạo Project mang tên , ví dụ “opcvwinform” kiểu windows forms application



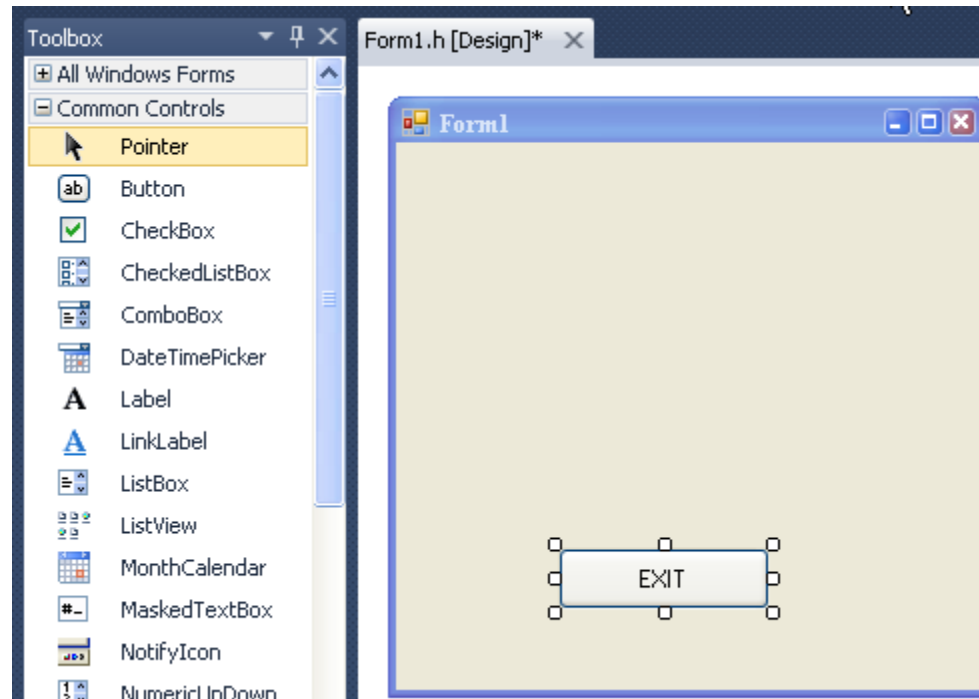
SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

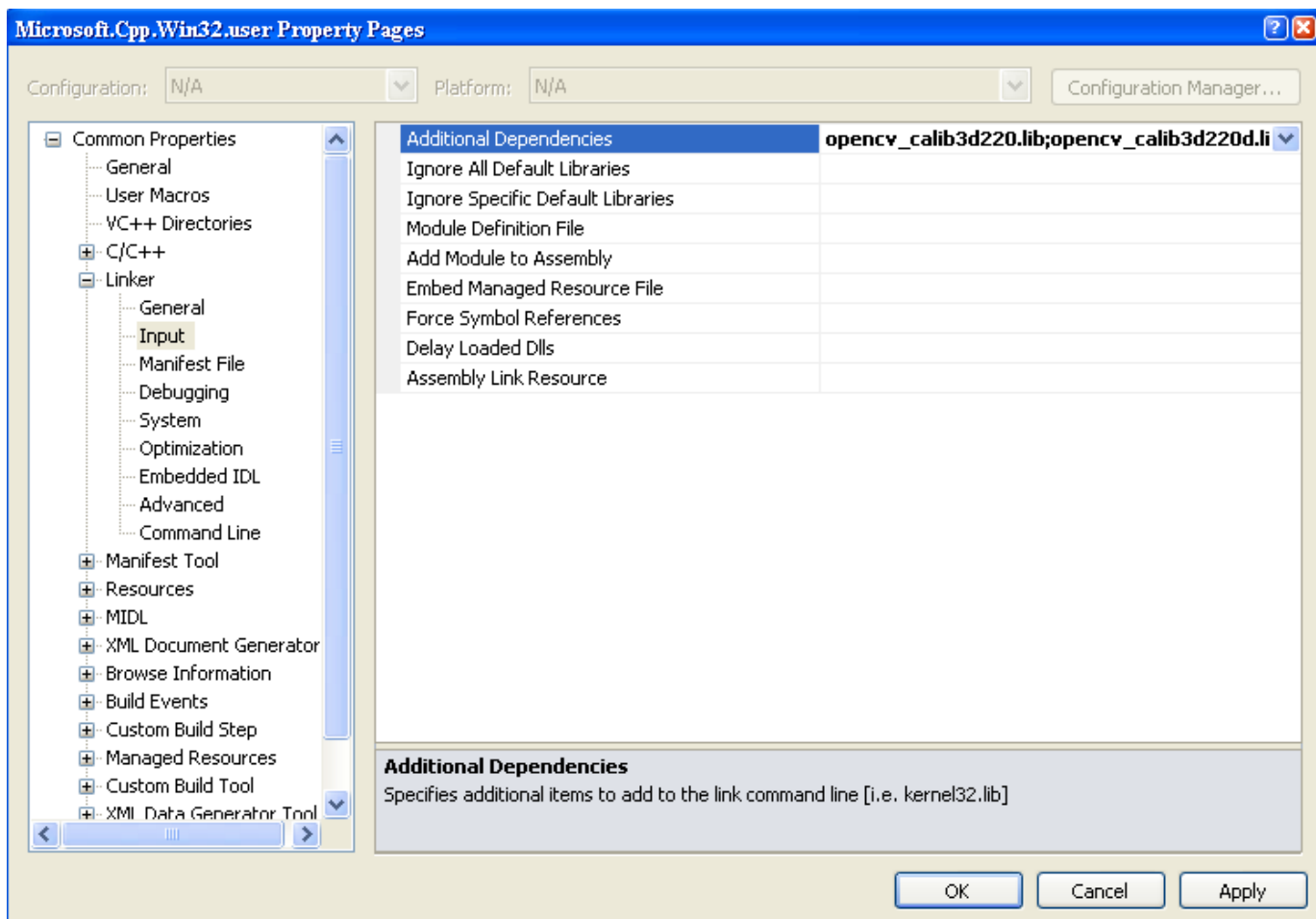
- Vào Properties , dòng General- Common Language Runtime Support chọn Common Language Runtime Support (/clr)



SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

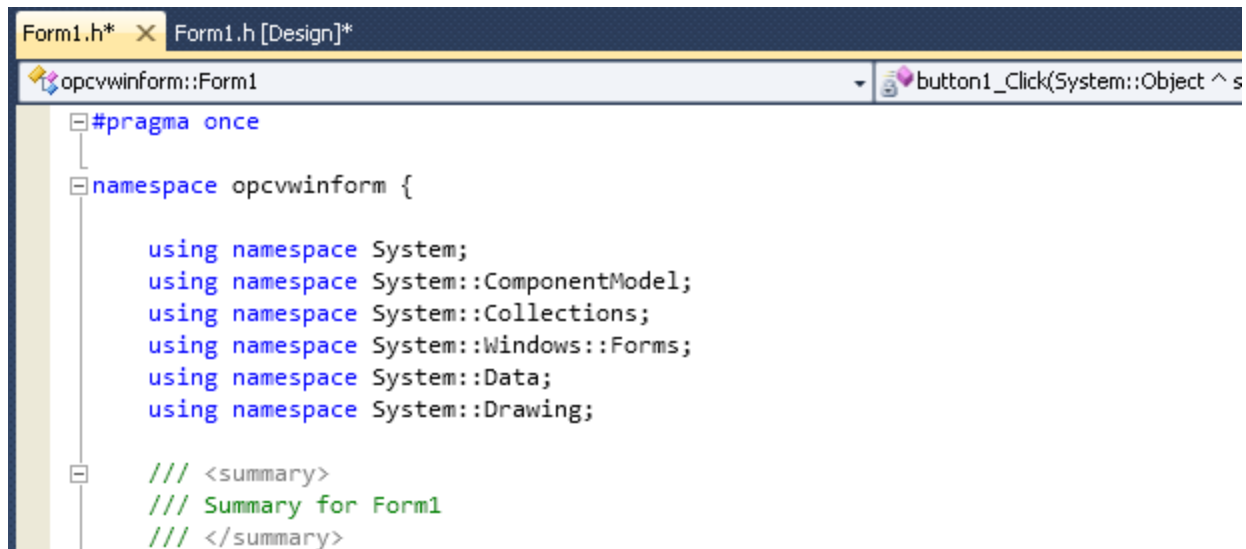
- Khai báo các đường dẫn cho include, lib... như trong win32 console
- Tạo button EXIT





SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

- Bấm chuột kép vào nút EXIT, mở ra code



```
Form1.h* x Form1.h [Design]*
opcvwinform::Form1 button1_Click(System::Object ^ s
#pragma once
namespace opcvwinform {
    using namespace System;
    using namespace System::ComponentModel;
    using namespace System::Collections;
    using namespace System::Windows::Forms;
    using namespace System::Data;
    using namespace System::Drawing;

    /// <summary>
    /// Summary for Form1
    /// </summary>
```

- Dưới dòng #pragma once thêm đoạn code sau

SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

```
#include <cv.h>
#include <highgui.h>
#ifdef _DEBUG
    //Debug
    #pragma comment(lib,"cv210d.lib")
    #pragma comment(lib,"cxcore210d.lib")
    #pragma comment(lib,"cvaux210d.lib")
    #pragma comment(lib,"highgui210d.lib")
#else
    //Release
    #pragma comment(lib,"cv210.lib")
    #pragma comment(lib,"cxcore210.lib")
    #pragma comment(lib,"cvaux210.lib")
    #pragma comment(lib,"highgui210.lib")
#endif
```

- Nếu dùng opencv2.2xx trở lên (ví dụ 2.4.8) ta dùng

```
#include <cxcore.h>
```

```
#include <highgui.h>
```

```
#ifdef _DEBUG
```

```
    //Debug
```

```
        #pragma comment(lib, "opencv_core248d.lib")
```

```
        #pragma comment(lib, "opencv_highgui248d.lib")
```

```
#else
```

```
    //Release
```

```
        #pragma comment(lib, "opencv_core248.lib")
```

```
        #pragma comment(lib, "opencv_highgui248.lib")
```

```
#endif
```

- Trong VC++ Dir khai báo include
 - Đường dẫn\build\include\opencv2
 - Đường dẫn\build\include\opencv
 - Đường dẫn \build\include
- Khai báo lib
 - Đường dẫn \build\x86\vc10\lib, nếu dùng WIN 64 thay x86 bằng x64, nếu dùng VS2012 thay vc10 bằng vc12
- Khai báo exe cho dll
 - Đường dẫn \build\x86\vc10\bin
- Linker-Input- Additional dependencies khai báo các thư viện

opencv_core248.lib

opencv_highgui248.lib

....

- Bấm inherit from parent để dùng cho các lần sau
- Vào C++ -General -Common Language Run Time Support Chọn Common Language Run Time Support (/clr)

SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

- Chỗ dòng lệnh
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {

}
- Thêm code vào giữa: this->Close();
- Bấm chuột kép vào form

```
#pragma endregion  
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)  
    this->Close();  
}  
private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {  
}  
};
```

- Thêm code vào chỗ form_load

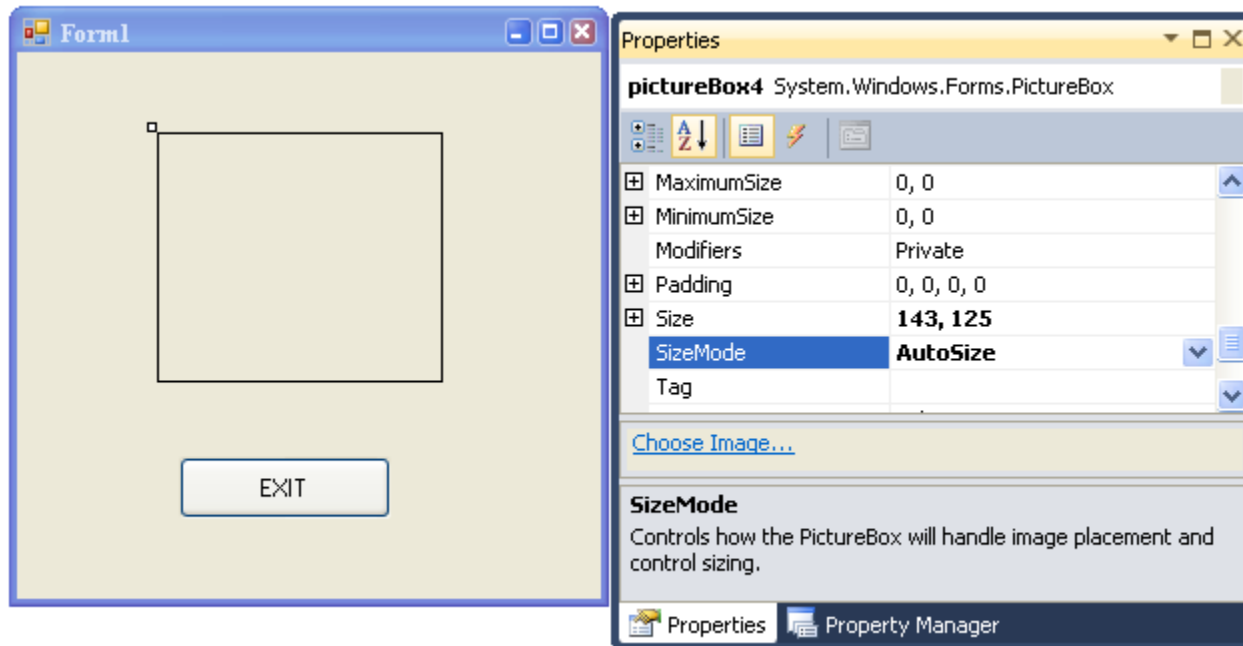
SỬ DỤNG OPENCV 2.1 VÀ VC 2008-2010 WINFORM APPLICATION

```
IpplImage* img = cvLoadImage("c:/anh.jpg");  
cvNamedWindow("WINFORM", CV_WINDOW_AUTOSIZE);  
cvShowImage("WINFORM", img);  
cvWaitKey(0);  
cvReleaseImage(&img);  
cvDestroyAllWindows();
```

Chỗ cvLoadImage() ta thêm đường dẫn và tên file ảnh

PICTURE BOX

- Tạo picture box trên form



PICTURE BOX

- Vào Properties của Picture Box1, chọn StretchMode- StretchImage, BorderStyle- FixedSingle
- Dưới #pragma once ta thêm các dòng khai báo như trước
- Thêm dòng khai báo biến chung `IplImage* img= NULL;`
- Đoạn mã sau được sử dụng để hiện ảnh lên PictureBox1 và thoát chương trình khi bấm EXIT

PICTURE BOX

```
private: System::Void Form1_Load(System::Object^ sender,
    System::EventArgs^ e) {
    Image* img = cvLoadImage( "c:/anh.jpg" );
    pictureBox1->Image = gcnew    //replacement of cvShowImage
    System::Drawing::Bitmap(img->width,img->height,img->widthStep,
    System::Drawing::Imaging::PixelFormat::Format24bppRgb,(System::Int
        Ptr) img->imageData);
    pictureBox1->Refresh();
}

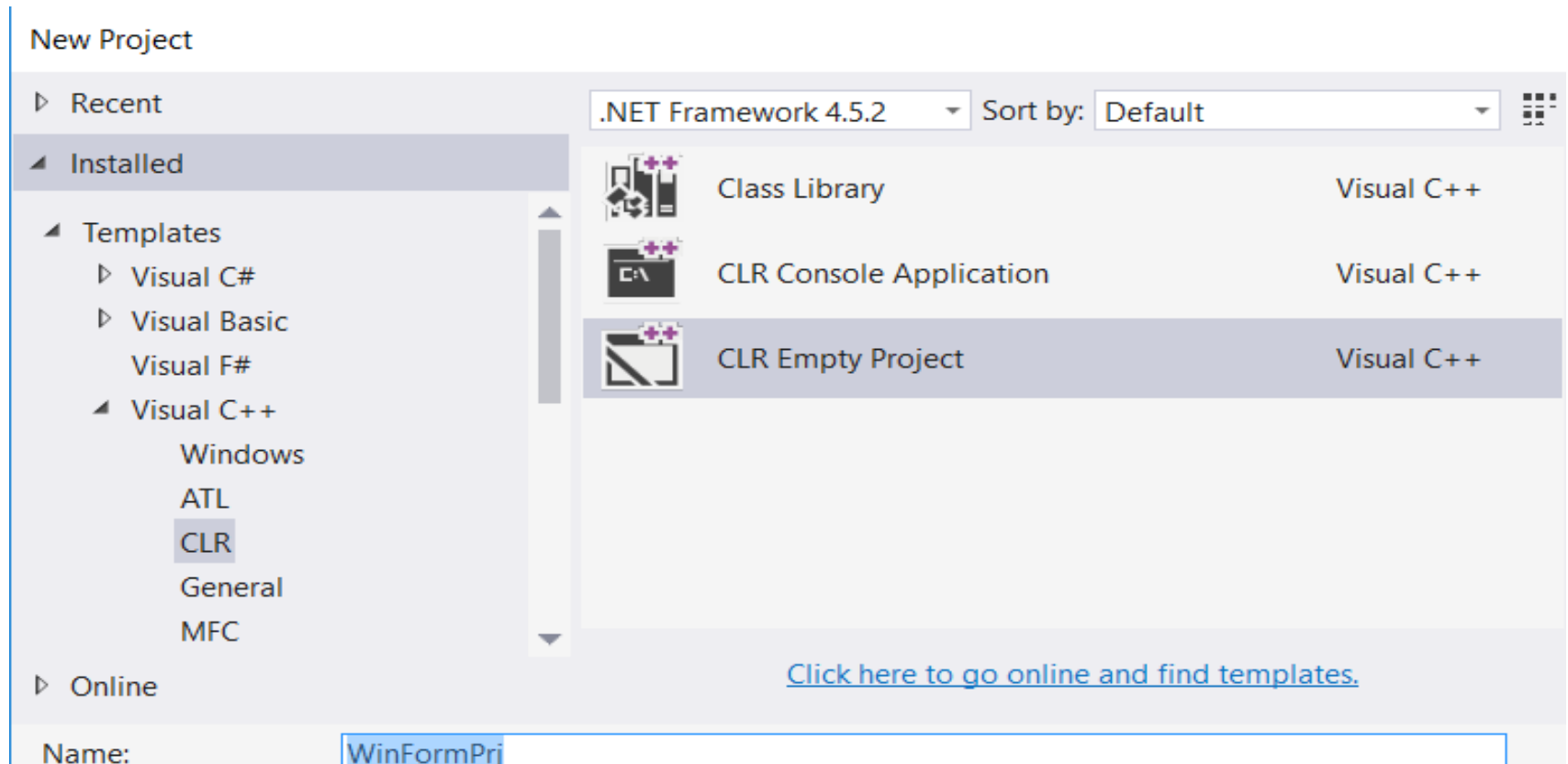
private: System::Void button1_Click(System::Object^ sender,
    System::EventArgs^ e) {
    cvReleaseImage(&img);
    this->Close();
}
```

PICTURE BOX



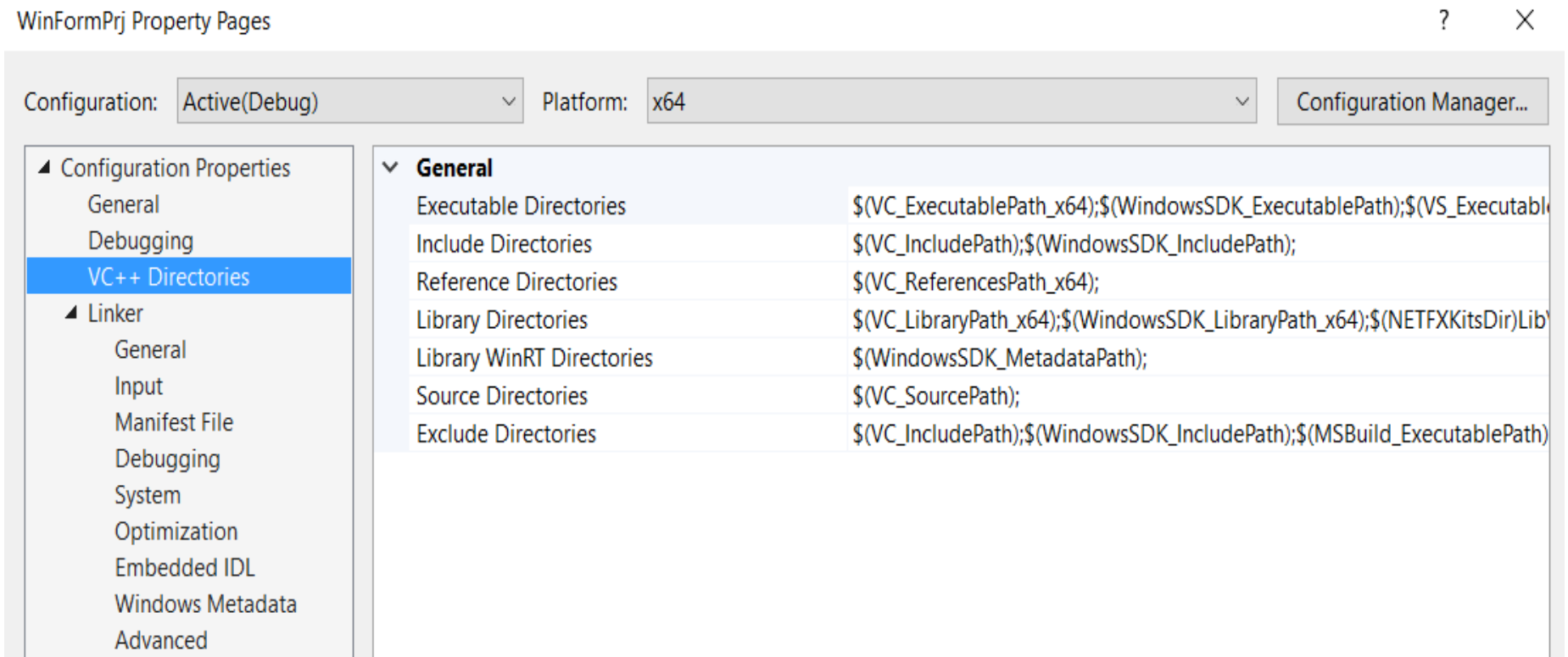
Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Tạo project dạng CLR (Common Language Runtime) Empty Project, đặt tên ví dụ WinFormPrj



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Thêm đường dẫn, vào Project Properties ,
mở cửa sổ Property Manager Debug X64



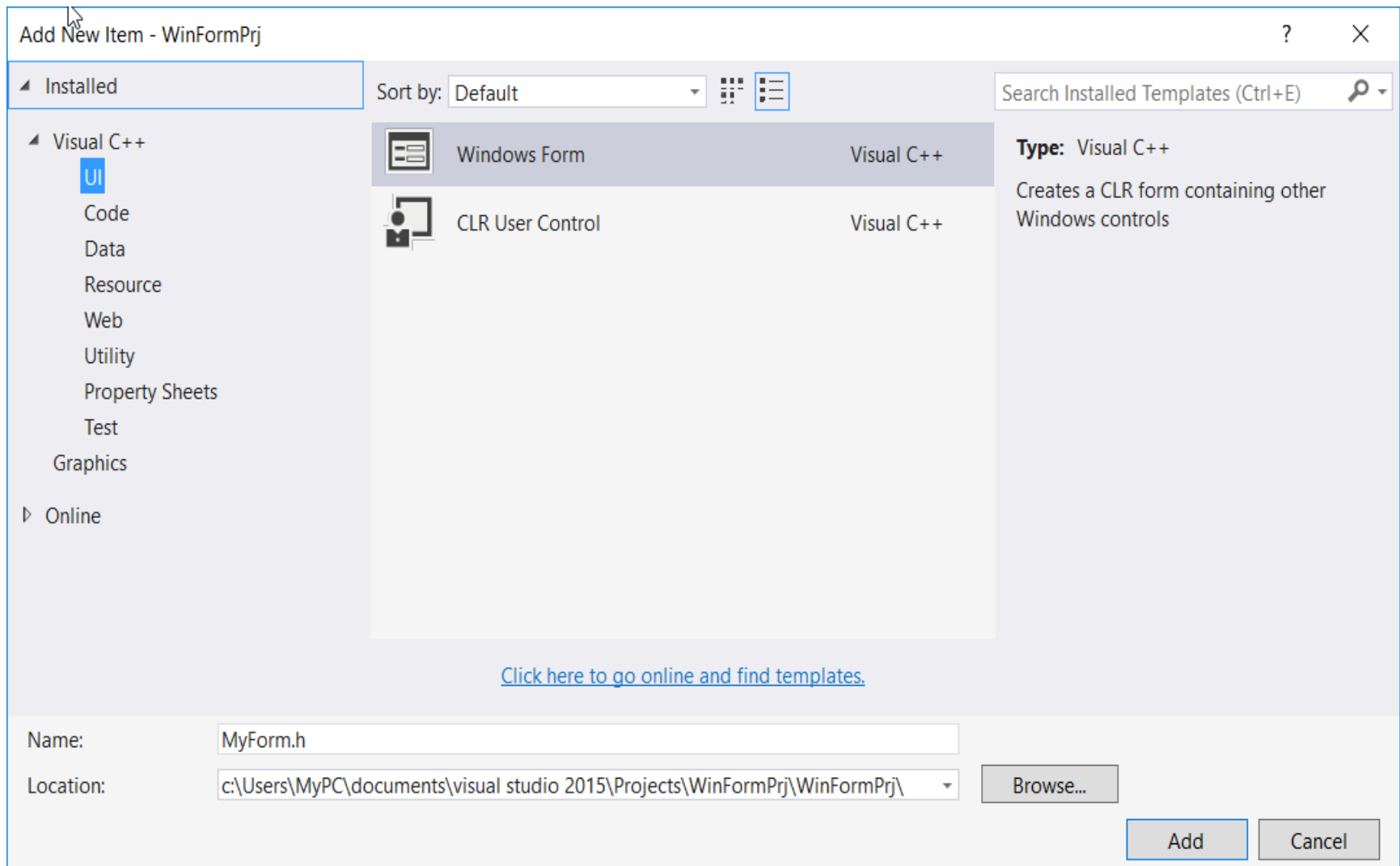
Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- VC++ Directories Executable Directories thêm đường dẫn C:\opencv\build\include, giả sử opencv trong thư mục trên
- VC++ Directories Include Directories thêm đường dẫn C:\opencv\build\include
- VC++ Directories Library Directories thêm đường dẫn C:\opencv\build\x64\vc14\lib
- Bấm Apply

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

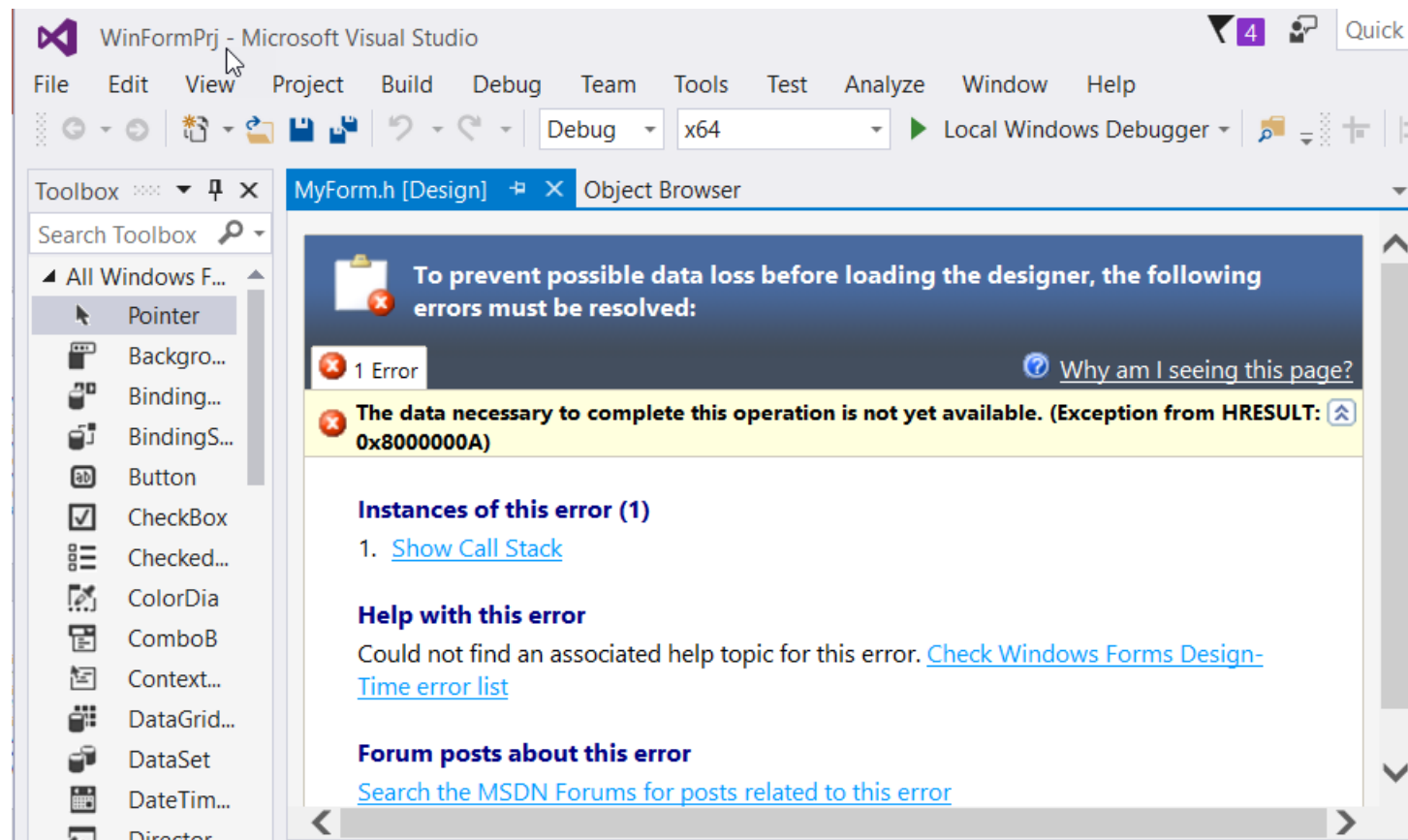
- Linker Input Additional Dependencies thêm opencv_world341d.lib
- Linker System Subsystem thêm Windows (/SUBSYSTEM:WINDOWS)
- Linker Advanced Entry Point Thêm main
- Apply OK
- Project Add New Item VisualC++ UI chọn Window Form Add

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015



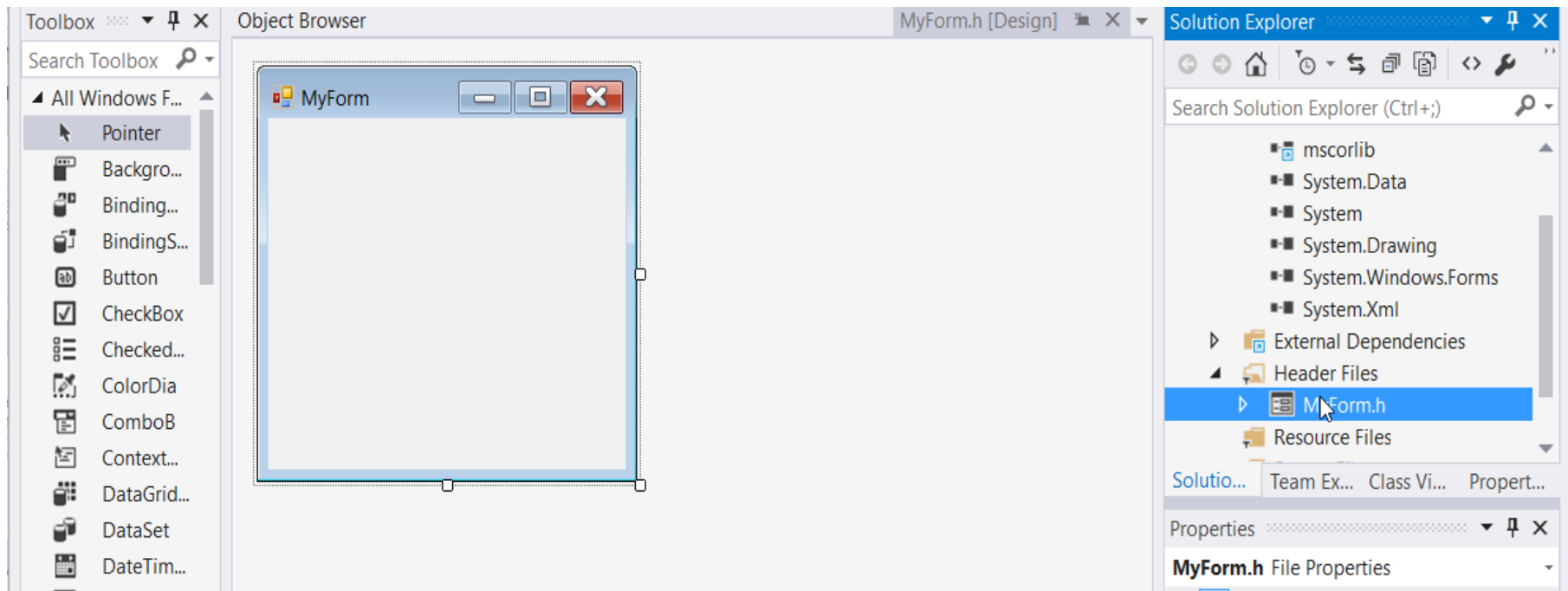
Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

Tắt MyForm.h (Design)



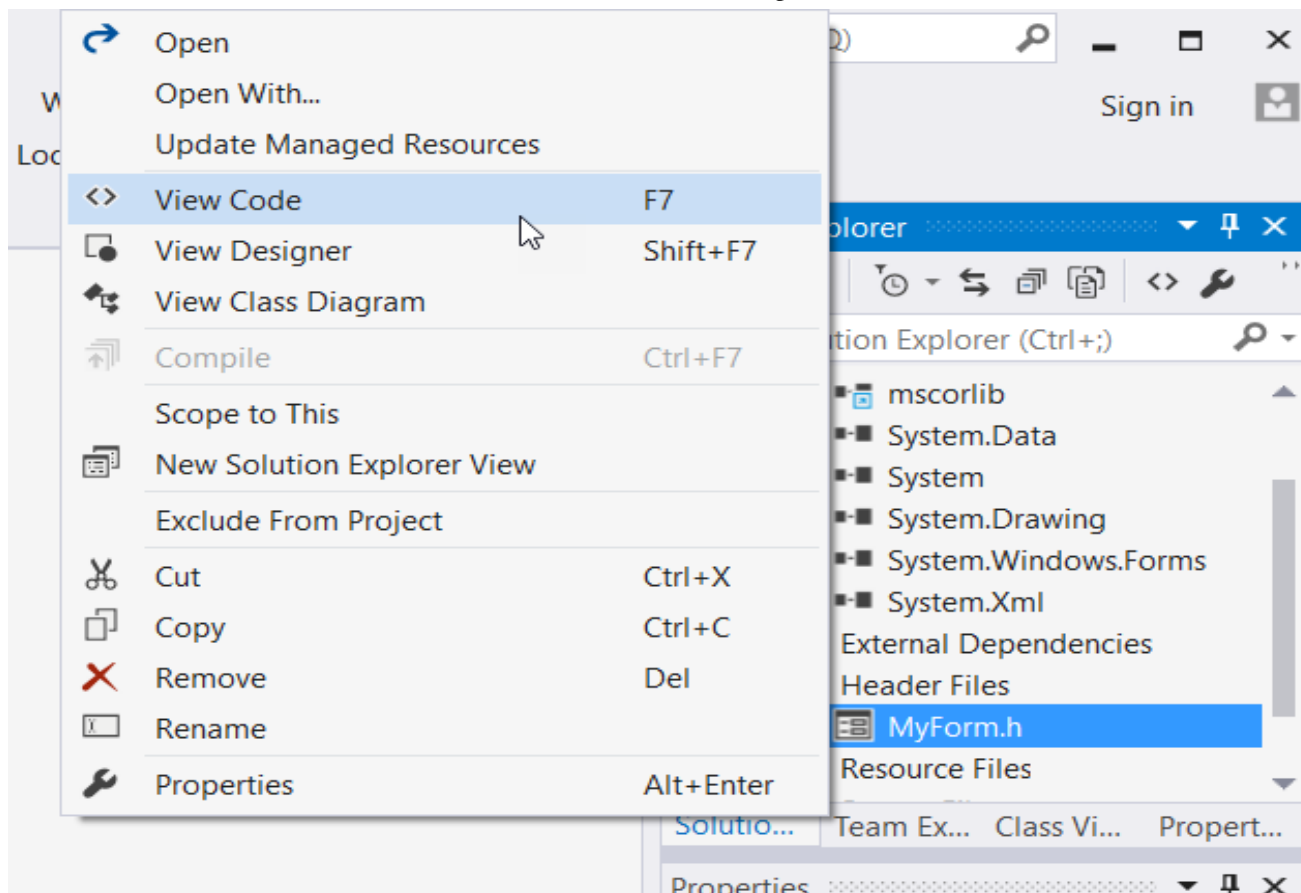
Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Vào Solution Explorer Header Files bấm vào MyForm.h

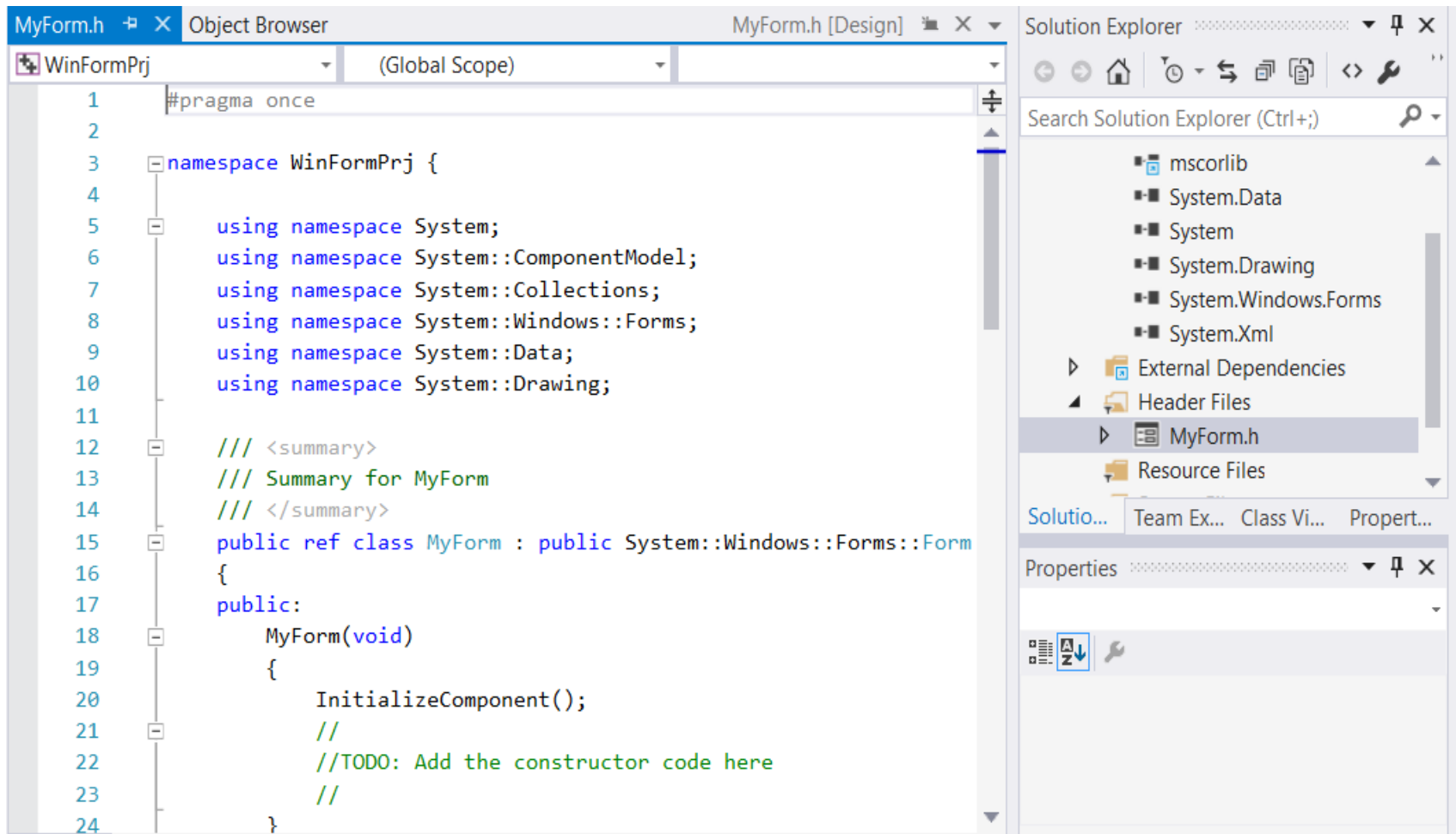


Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Bấm chuột phải vào MyForm.h View Code



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Trong MyForm.h, thêm các dòng sau dưới #pragma once, chú ý chọn Debug x64

```
#include <opencv2/core/core.hpp>
```

```
#include <opencv2/highgui/highgui.hpp>
```

```
#include <opencv2/imgcodecs/imgcodecs.hpp>
```

(tùy theo ta muốn dùng opencv vào việc gì ,hoặc chỉ cần dùng dòng dưới đây là đủ), dòng pragma để tắt warning

```
#pragma managed(push,off)
```

```
#include <opencv2/opencv.hpp>
```

```
#pragma managed(pop)
```

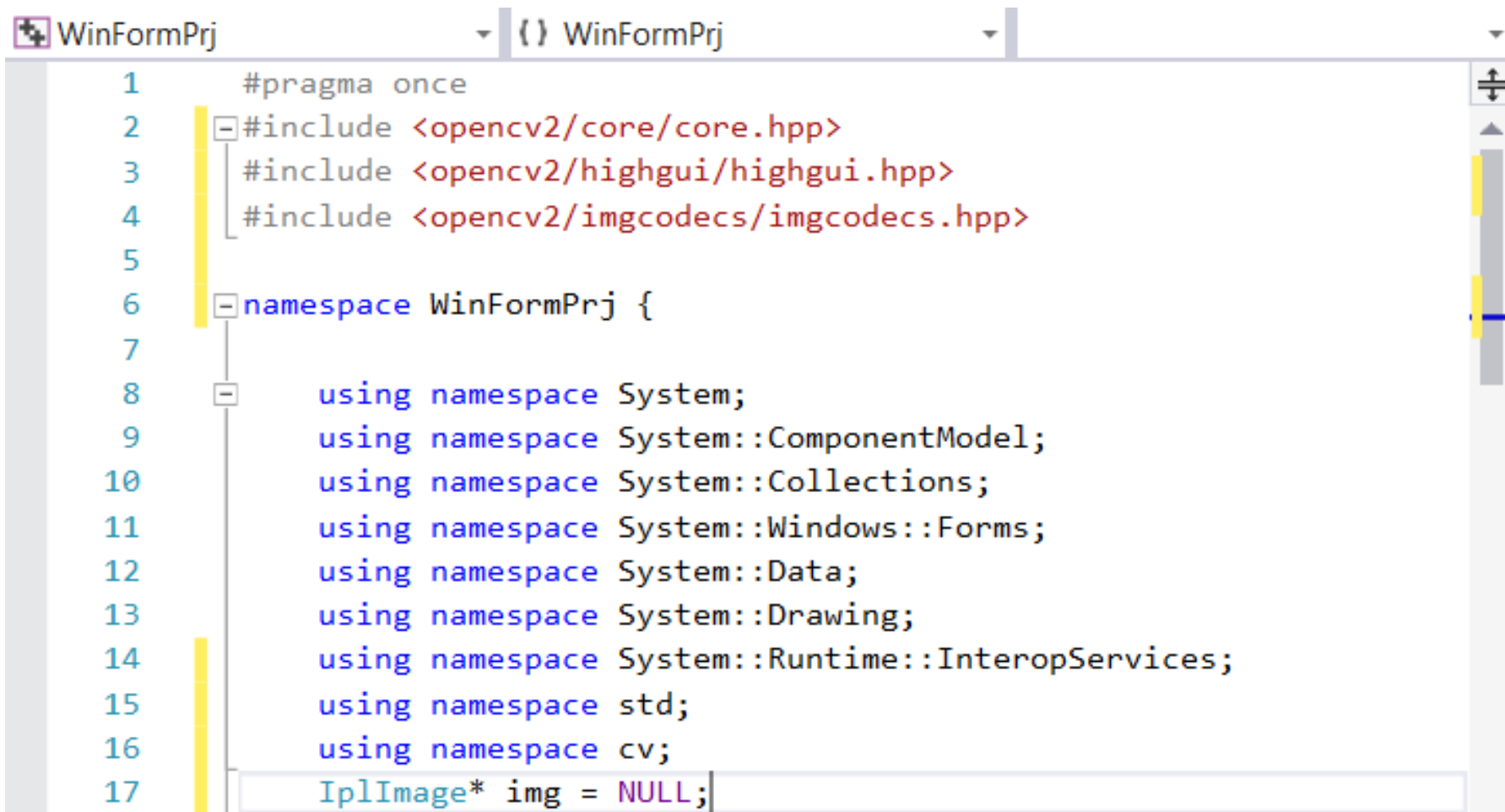
- Dưới các dòng using ta thêm

```
using namespace System::Runtime::InteropServices;
```

```
using namespace std; using namespace cv;
```

```
ImgImage* img = NULL;
```

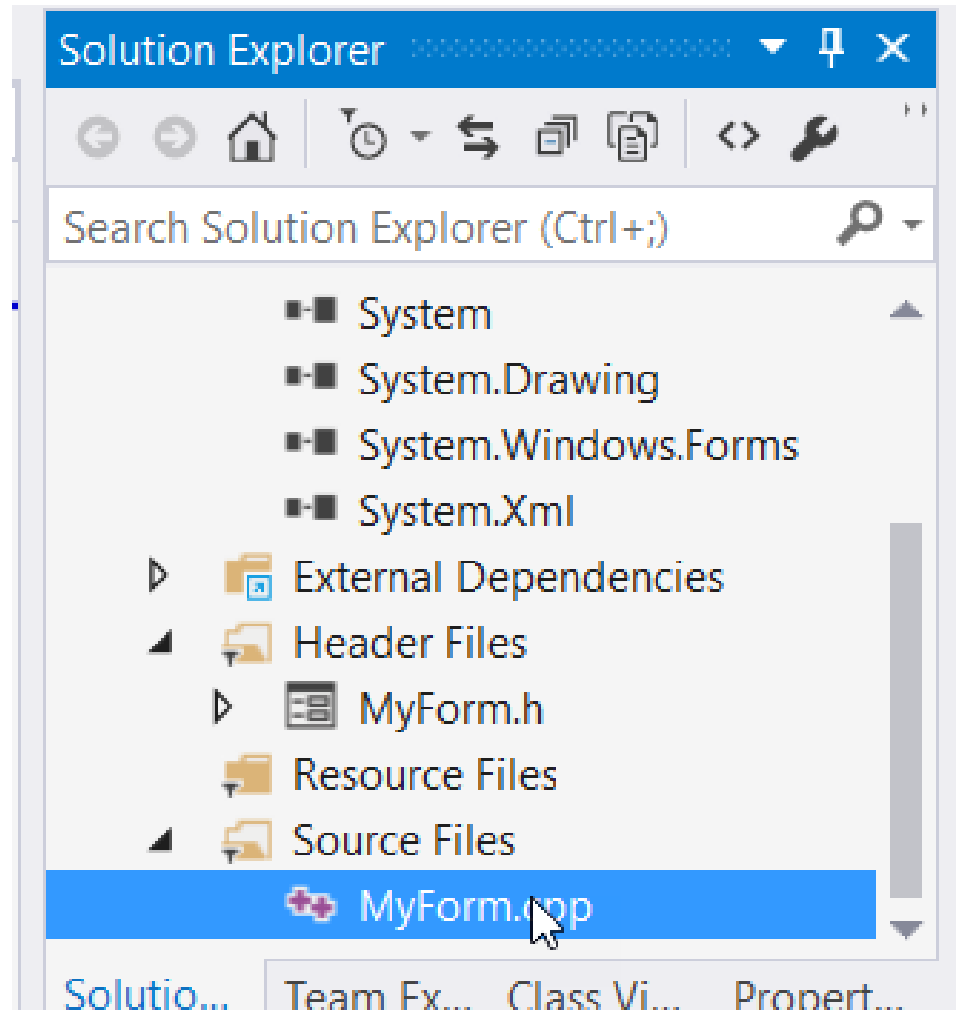

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015



```
1  #pragma once
2  #include <opencv2/core/core.hpp>
3  #include <opencv2/highgui/highgui.hpp>
4  #include <opencv2/imgcodecs/imgcodecs.hpp>
5
6  namespace WinFormPrj {
7
8      using namespace System;
9      using namespace System::ComponentModel;
10     using namespace System::Collections;
11     using namespace System::Windows::Forms;
12     using namespace System::Data;
13     using namespace System::Drawing;
14     using namespace System::Runtime::InteropServices;
15     using namespace std;
16     using namespace cv;
17     IplImage* img = NULL;
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Solution Explorer Source Files bấm mở MyForm.cpp



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Thêm các dòng sau vào MyForm.cpp dưới using MyForm.h (WinFormPrj là tên project trong ví dụ này, còn MyForm.h là mặc định, nếu đã thay đổi thì chỉnh sửa cho phù hợp)

```
using namespace System;
```

```
using namespace System::Windows::Forms;
```

```
[STAThread]
```

```
int main(array<String^>^ args){
```

```
Application::EnableVisualStyles();
```

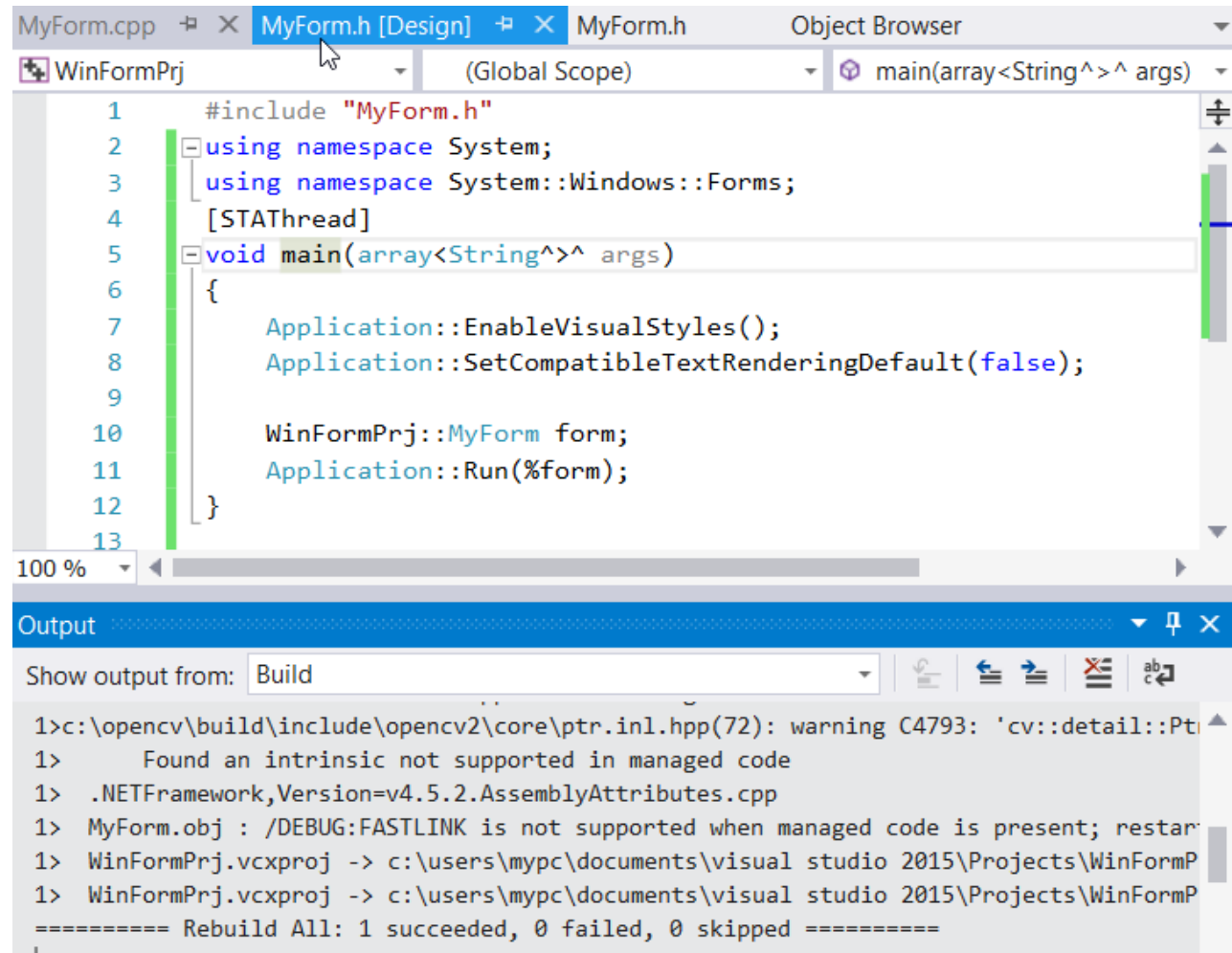
```
Application::SetCompatibleTextRenderingDefault(false);
```

```
WinFormPrj::MyForm form;
```

```
Application::Run(%form);}
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Build Build Solution
- Nếu không có error bạn đã thành công bước đầu!!!



The screenshot displays the Visual Studio 2015 IDE. The top pane shows the code for `MyForm.h` in Design view. The code includes the `MyForm.h` header, uses the `System` and `System::Windows::Forms` namespaces, and defines a `main` function within a `[STAThread]` block. The `main` function calls `Application::EnableVisualStyles()`, `Application::SetCompatibleTextRenderingDefault(false)`, creates a `WinFormPrj::MyForm` instance, and calls `Application::Run(%form)`.

The bottom pane shows the Output window with the following text:

```
Show output from: Build

1>c:\opencv\build\include\opencv2\core\ptr.inl.hpp(72): warning C4793: 'cv::detail::Ptr<cv::Mat_>'
1> Found an intrinsic not supported in managed code
1> .NETFramework,Version=v4.5.2.AssemblyAttributes.cpp
1> MyForm.obj : /DEBUG:FASTLINK is not supported when managed code is present; restart
1> WinFormPrj.vcxproj -> c:\users\mypc\documents\visual studio 2015\Projects\WinFormP
1> WinFormPrj.vcxproj -> c:\users\mypc\documents\visual studio 2015\Projects\WinFormP
===== Rebuild All: 1 succeeded, 0 failed, 0 skipped =====
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015

- Bấm vào MyForm.h(Design) thêm vào 2 Button và một Picture Box, chỉnh vị trí và kích thước phù hợp.
- Button1 Name: OpenImage, Text: Open Image
- Button2 Name, Text: Exit
- PictureBox1 SizeMode: Zoom
- Bấm chuột vào các button để tạo sự kiện
- Vào MyForm.h thêm code

```
#pragma endregion
private: System::Void OpenImage_Click(System::Object^ sender, System::EventArgs^ e)
{
}
private: System::Void Exit_Click(System::Object^ sender, System::EventArgs^ e)
{
};
}
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

- Thêm các dòng sau vào Open_Image_Click
img = cvLoadImage("c:/anh.jpg");//đường dẫn ảnh
pictureBox1->Image = gnew
System::Drawing::Bitmap(img->width,img->height,img->widthStep,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(System::IntPtr) img->imageData);
pictureBox1->Refresh();
- Thêm các dòng sau vào Exit_Click
cvReleaseImage(&img);
this->Close();

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++

- Nếu dùng ảnh dạng Mat thì khi đưa ảnh vào pictureBox ta phải điều chỉnh code. Thêm các dòng sau vào open_Image_Click

//Cách 1: đọc ảnh Mat xử lý.. Rồi lưu vào đĩa, sau đó cho pictureBox đọc file

```
img = imread("c://jlo.jpg");
```

```
imwrite("d://im.jpg", img);
```

```
pictureBox1->Image = Image::FromFile("d://im.jpg");
```

//Cách 2: Cách 1 bất tiện nếu ta dùng video hay camera

```
img = imread("c://jlo.jpg");
```

Trên phần include ta thêm dòng `#include <windows.h>`

Phần code:

```
cvtColor(img, img, CV_BGR2BGRA);
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

//Tạo bitmap từ ảnh Mat

```
HBITMAP hBit = CreateBitmap(img.cols, img.rows, 1, 32, img.data);
```

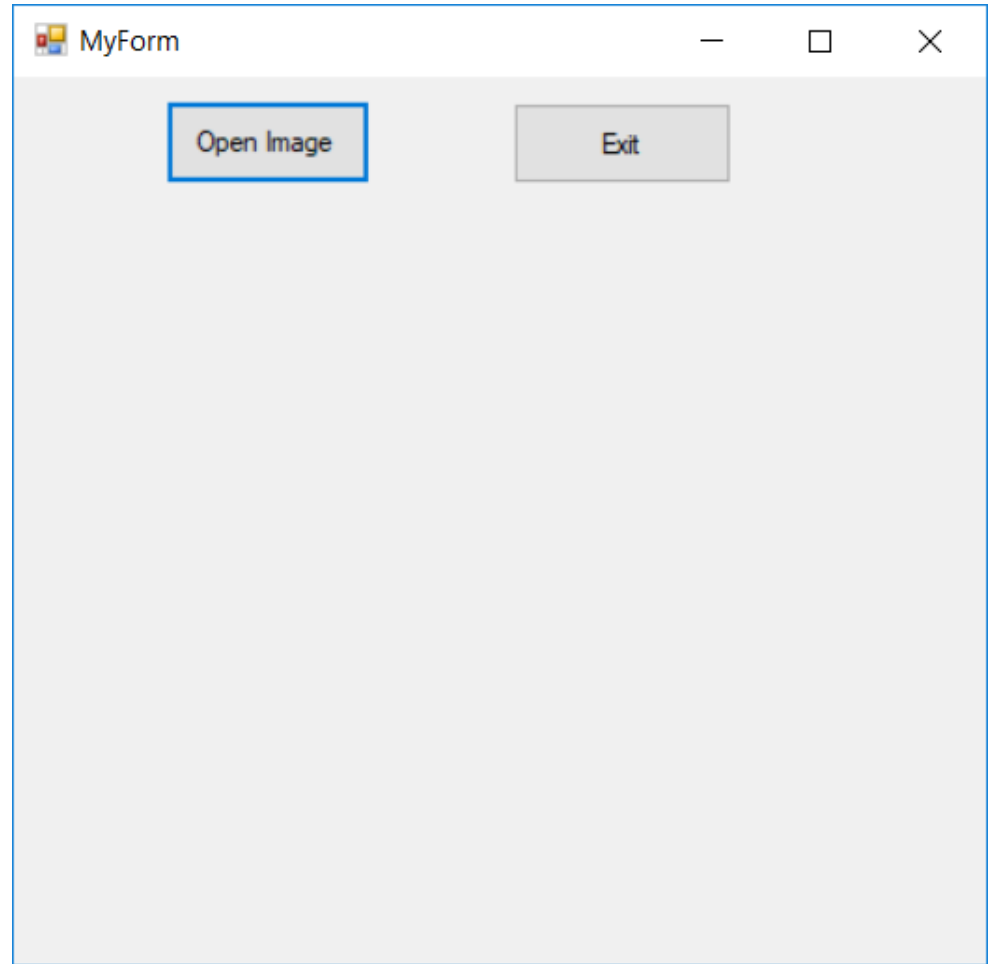
```
Bitmap^ bmp = Bitmap::FromHbitmap((IntPtr)hBit);
```

```
pictureBox1->Image = bmp;
```

Dùng các lệnh C++ của opencv như cv:: thuận tiện hơn dùng lệnh C

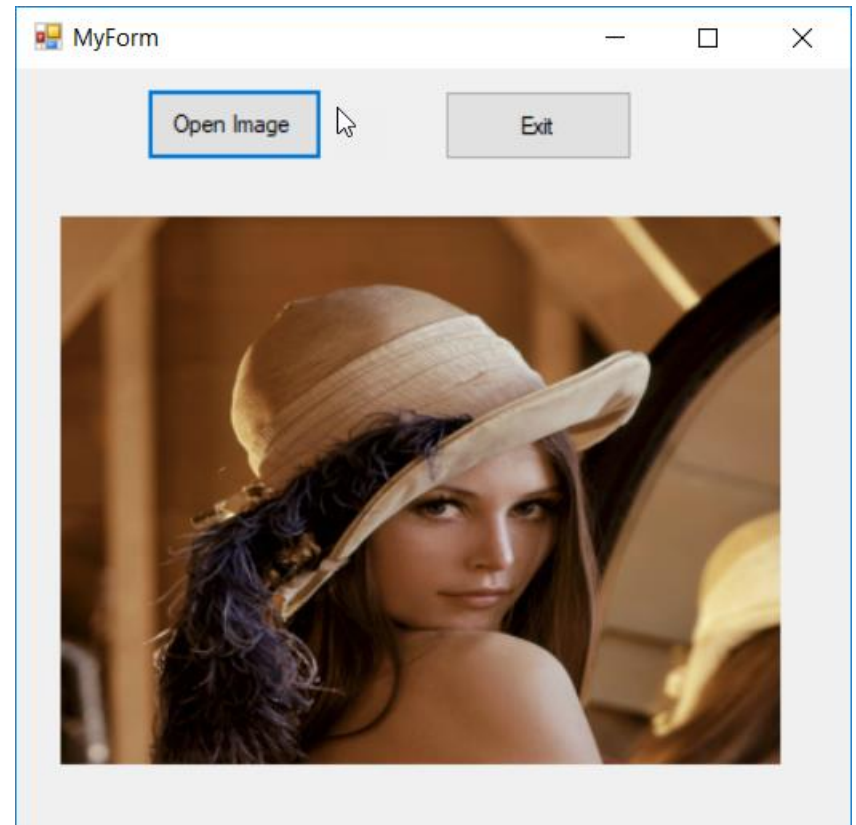
Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

- Build Nếu không có error bạn sắp thành công
- Debug Start Debugging



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

- Bấm vào nút Open Image
- Bấm vào nút Exit thoát chương trình
- Bạn đã tạo một project opencv winform thành công



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

- Viết chương trình gồm hai pictureBox và ba nút nhấn LOAD, GRAY EXIT
- Nhấn LOAD nạp ảnh màu và pictureBox1,
- Nhấn GRAY đổi ảnh màu ra ảnh xám nạp vào pictureBox2,
- Nhấn EXIT thoát chương trình
- Dùng ảnh Mat

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

Color2GrayWinForm_2_2019 Property Pages

Configuration: Debug Platform: x64 Configurat

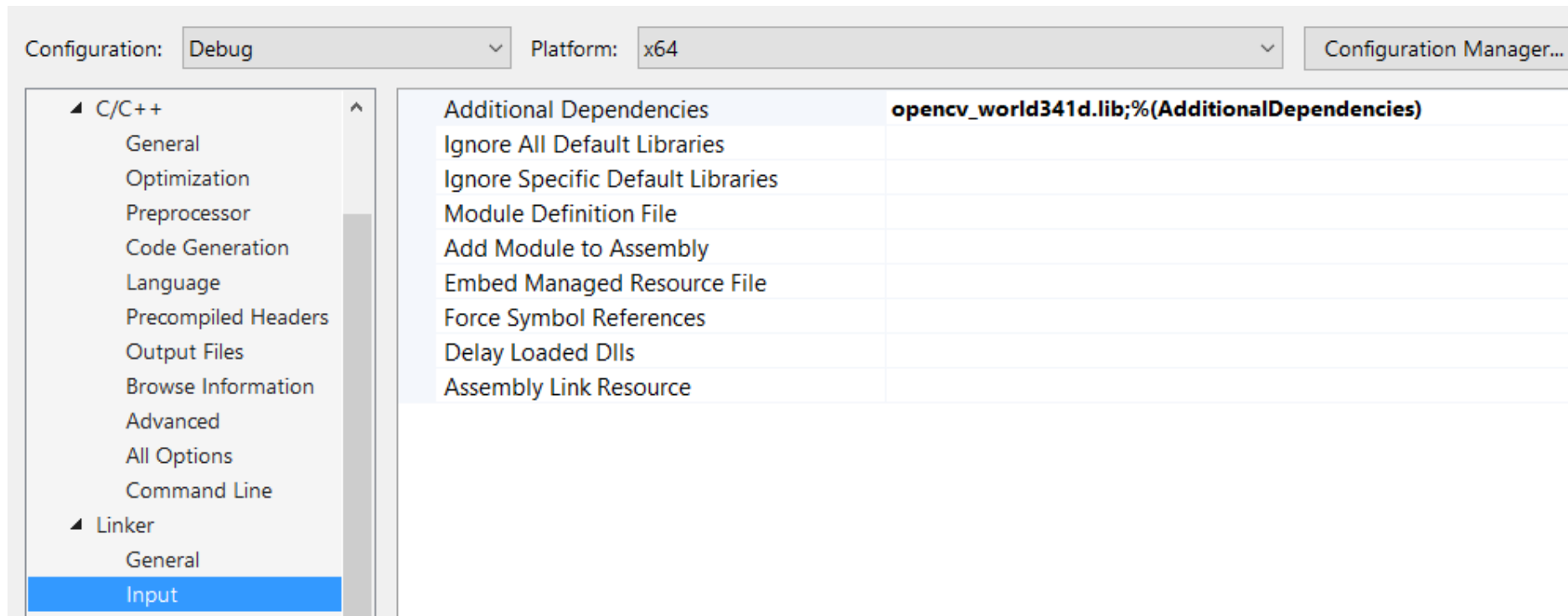
<ul style="list-style-type: none"> Configuration Properties <ul style="list-style-type: none"> General Debugging VC++ Directories C/C++ 	General <table> <tr> <td>Executable Directories</td> <td>C:\opencv\build\x64\vc14\bin;\$(ExecutablePath)</td> </tr> <tr> <td>Include Directories</td> <td>C:\opencv\build\include;\$(IncludePath)</td> </tr> <tr> <td>Reference Directories</td> <td>\$(VC_ReferencesPath_x64);</td> </tr> <tr> <td>Library Directories</td> <td>C:\opencv\build\x64\vc14\lib;\$(LibraryPath)</td> </tr> </table>	Executable Directories	C:\opencv\build\x64\vc14\bin;\$(ExecutablePath)	Include Directories	C:\opencv\build\include;\$(IncludePath)	Reference Directories	\$(VC_ReferencesPath_x64);	Library Directories	C:\opencv\build\x64\vc14\lib;\$(LibraryPath)
Executable Directories	C:\opencv\build\x64\vc14\bin;\$(ExecutablePath)								
Include Directories	C:\opencv\build\include;\$(IncludePath)								
Reference Directories	\$(VC_ReferencesPath_x64);								
Library Directories	C:\opencv\build\x64\vc14\lib;\$(LibraryPath)								

Configuration: Debug Platform: x64 Configuration Ma

<ul style="list-style-type: none"> Configuration Properties <ul style="list-style-type: none"> General Debugging VC++ Directories C/C++ General 	<table> <tr> <td>Additional Include Directories</td> <td>C:\opencv\build\include;%(AdditionalIncludeDirectories)</td> </tr> <tr> <td>Additional #using Directories</td> <td></td> </tr> <tr> <td>Debug Information Format</td> <td>Program Database (/Zi)</td> </tr> <tr> <td>Common Language RunTime Support</td> <td>Common Language RunTime Support (/clr)</td> </tr> <tr> <td>Consume Windows Runtime Extension</td> <td></td> </tr> <tr> <td>Suppress Startup Banner</td> <td>Yes (/nologo)</td> </tr> </table>	Additional Include Directories	C:\opencv\build\include;%(AdditionalIncludeDirectories)	Additional #using Directories		Debug Information Format	Program Database (/Zi)	Common Language RunTime Support	Common Language RunTime Support (/clr)	Consume Windows Runtime Extension		Suppress Startup Banner	Yes (/nologo)
Additional Include Directories	C:\opencv\build\include;%(AdditionalIncludeDirectories)												
Additional #using Directories													
Debug Information Format	Program Database (/Zi)												
Common Language RunTime Support	Common Language RunTime Support (/clr)												
Consume Windows Runtime Extension													
Suppress Startup Banner	Yes (/nologo)												

<ul style="list-style-type: none"> Browse Information Advanced All Options Command Line Linker 	<table> <tr> <td>Additional Library Directories</td> <td>C:\opencv\build\x64\vc14\lib;%(AdditionalLibraryDirectories)</td> </tr> <tr> <td>Link Library Dependencies</td> <td>Yes</td> </tr> <tr> <td>Use Library Dependency Inputs</td> <td>No</td> </tr> <tr> <td>Link Status</td> <td></td> </tr> <tr> <td>Prevent Dll Binding</td> <td></td> </tr> </table>	Additional Library Directories	C:\opencv\build\x64\vc14\lib;%(AdditionalLibraryDirectories)	Link Library Dependencies	Yes	Use Library Dependency Inputs	No	Link Status		Prevent Dll Binding	
Additional Library Directories	C:\opencv\build\x64\vc14\lib;%(AdditionalLibraryDirectories)										
Link Library Dependencies	Yes										
Use Library Dependency Inputs	No										
Link Status											
Prevent Dll Binding											

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

Color2GrayWinForm_2_2019 Properties

Configuration:

Debug

Platform:

x64

▲ C/C++

- General
- Optimization
- Preprocessor
- Code Generation
- Language
- Precompiled Headers
- Output Files
- Browse Information
- Advanced
- All Options
- Command Line

▲ Linker

- General
- Input
- Manifest File
- Debugging

System

SubSystem

Windows (/SUBSYSTEM:WINDOWS)

Minimum Required Version

Heap Reserve Size

Heap Commit Size

Stack Reserve Size

Stack Commit Size

Enable Large Addresses

Terminal Server

Swap Run From CD

No

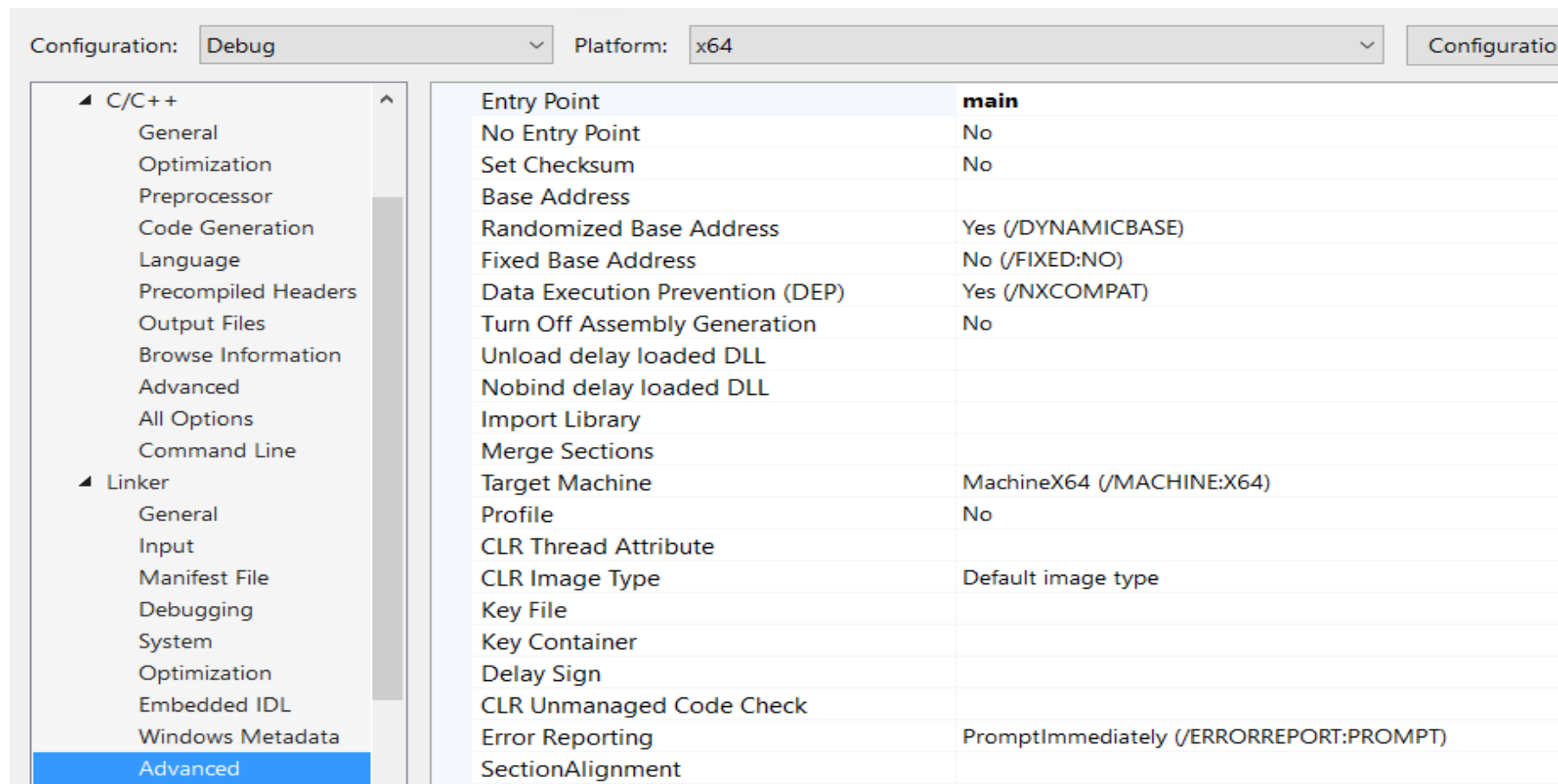
Swap Run From Network

No

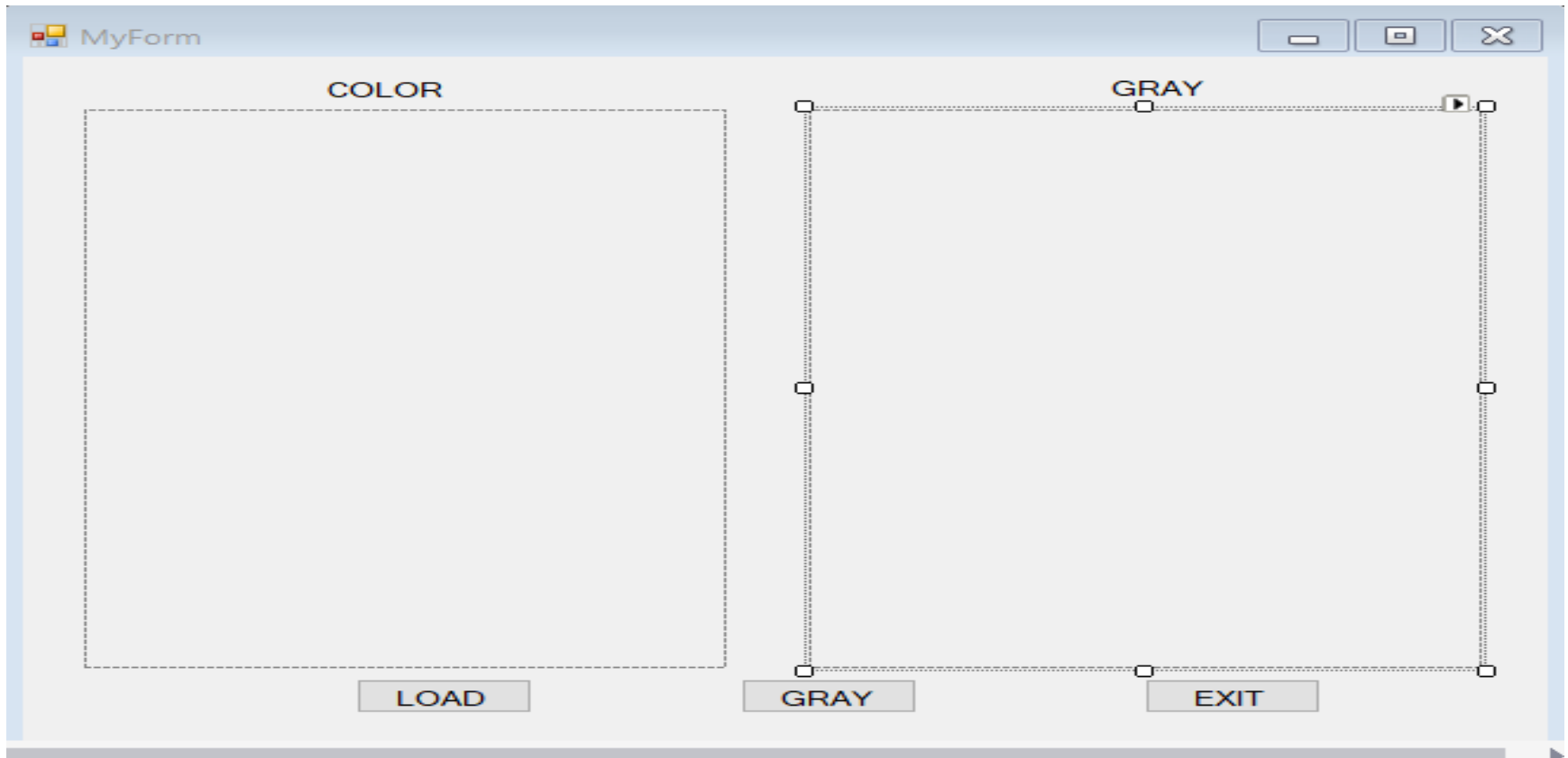
Driver

Not Set

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C



pictureBox:SizeMode: Zoom
Gray Button: Enabled: False

TS NGUYỄN ĐỨC THÀNH

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

```
#pragma once
#pragma managed(push,off)
#include <opencv2/opencv.hpp>
#include <windows.h>
#pragma managed(pop)
namespace Color2GrayWinForm_2_2019 {
using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Runtime::InteropServices;
using namespace std; using namespace cv;
Mat img;
```

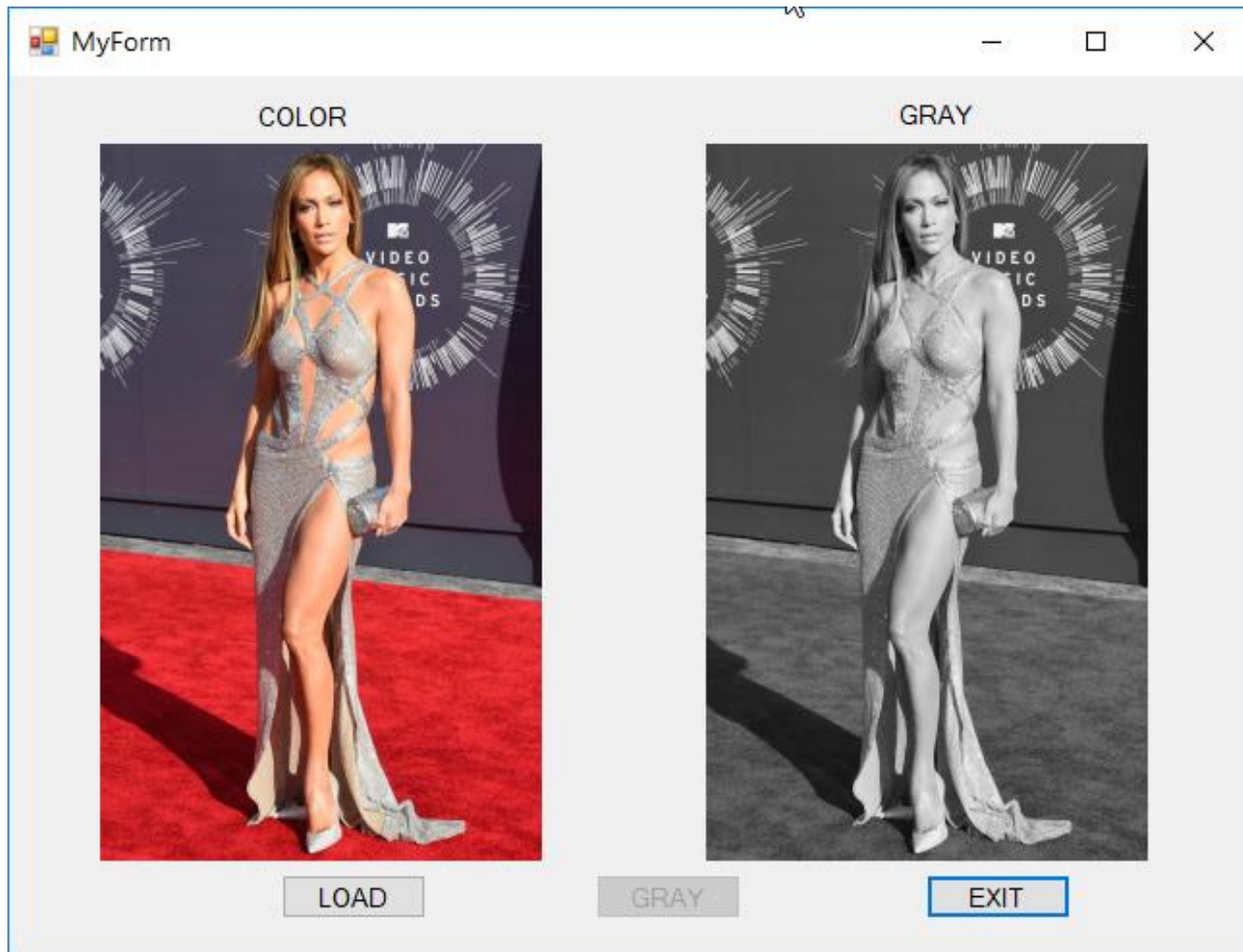
Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

```
private: System::Void LOAD_Click(System::Object^ sender,
System::EventArgs^ e) {
img = imread("c:/jlo.jpg");
cvtColor(img, img, CV_BGR2BGRA);
HBITMAP hBit = CreateBitmap(img.cols, img.rows, 1, 32, img.data);
Bitmap^ bmp = Bitmap::FromHbitmap((IntPtr)hBit);
pictureBox1->Image = bmp;
GRAY->Enabled = 1;
}
private: System::Void EXIT_Click(System::Object^ sender,
System::EventArgs^ e) {
this->Close();
}
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C

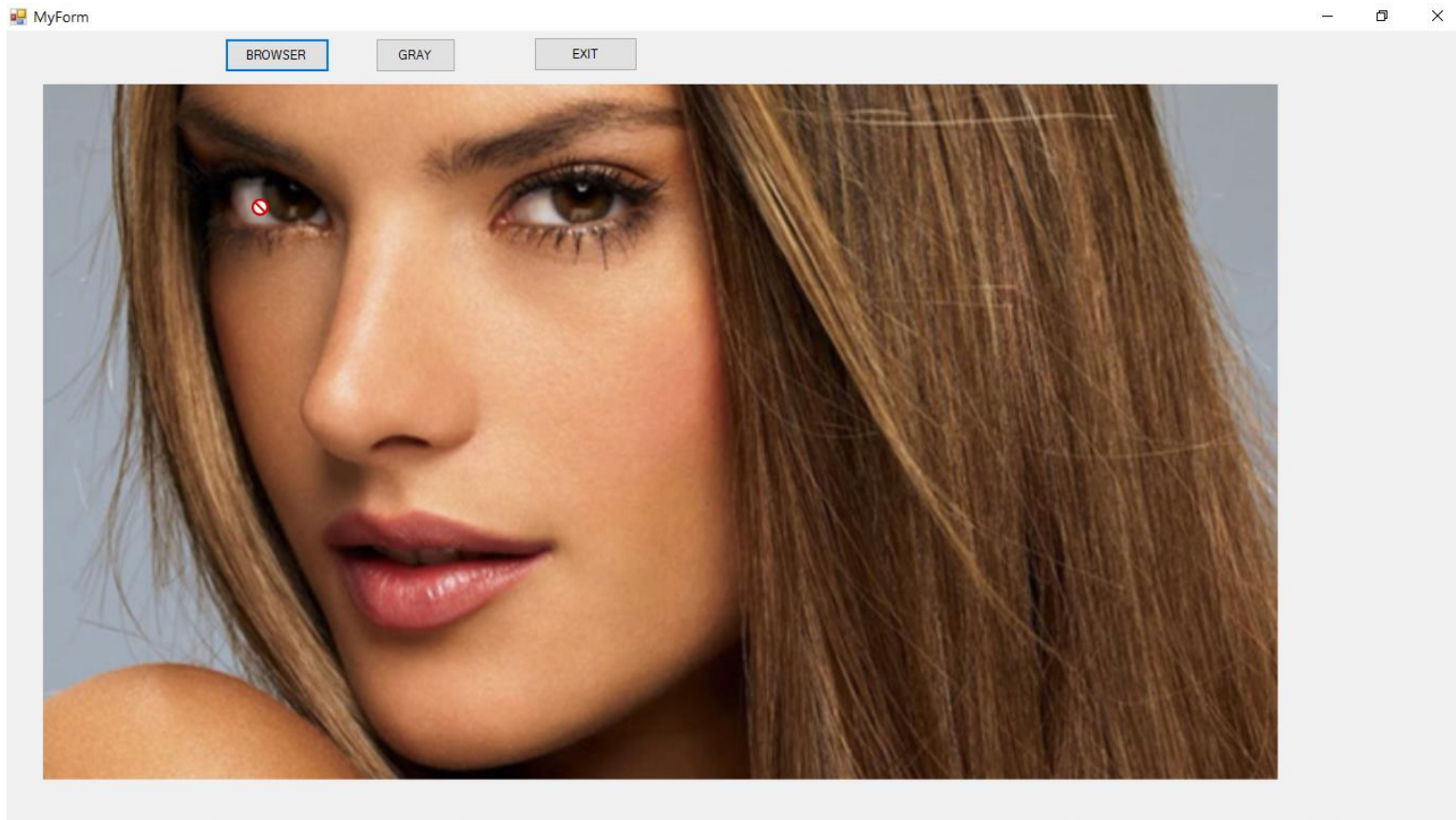
```
private: System::Void GRAY_Click(System::Object^ sender,  
System::EventArgs^ e) {  
    Mat gray;  
    cvtColor(img, gray, CV_BGR2GRAY);  
    cvtColor(gray, gray, CV_GRAY2BGR);  
    cvtColor(gray, gray, CV_BGR2BGRA);  
    HBITMAP hBit = CreateBitmap(gray.cols, gray.rows, 1, 32, gray.data);  
    Bitmap^ bmp = Bitmap::FromHbitmap((IntPtr)hBit);  
    pictureBox2->Image = bmp;  
    GRAY->Enabled = 0;  
}
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Image -> PictureBox

- Viết chương trình gồm 3 nút nhấn và một picturebox, hình sẽ đưa vào PB và chuyển sang gray khi bấm nút



Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Image ->PictureBox

- PB chọn size mode là zoom
- Trong MyForm.h

```
#pragma managed(push,off)
#include <opencv2/opencv.hpp>
#include <windows.h>
#pragma managed(pop)
using namespace std;
using namespace cv;
Mat img; Mat gray;
IplImage* iplgray;
IplImage* iplimg; IplImage* iplimggr; IplImage* iplimgt;
IplImage* iplimgt1;
```

Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Image -> PictureBox

- Khi nhấn nút Browser chọn file ảnh và đưa vào PB
- ```
private: System::Void btnBrowser_Click(System::Object^
sender, System::EventArgs^ e) {
 OpenFileDialog^ dgopen = gcnew OpenFileDialog();
 dgopen->Filter = "Image (*.bmp;*.jpg|*.bmp;*.jpg|All Files
(*.*|*.*|))";
 if (dgopen->ShowDialog() ==
 System::Windows::Forms::DialogResult::Cancel)
 {return;}
 img = imread(convertstring2char(dgopen-
```

# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++

## Image -> PictureBox

```
>FileName));
if (img.empty()) exit;
iplimg = cvCloneImage(&(IplImage)img); //Mat to IplImage
iplimgt = cvCreateImage(cvSize((int)(iplimg-> width*0.5),
(int)(iplimg->height*0.5)),iplimg->depth, iplimg->nChannels); //Resize image
cvResize(iplimg, iplimgt);
pbsrc->Image = gcnew System::Drawing:: Bitmap(iplimgt->width, iplimgt->height, iplimgt->widthStep,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(System::IntPtr) iplimgt->imageData);
pbsrc->Refresh();
```



# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Image -> PictureBox

- Nhấn nút Gray chuyển sang ảnh xám đưa vào PB

```
private: System::Void btnGray_Click(System::Object^
sender, System::EventArgs^ e) {
```

```
Mat gray;
```

```
cvtColor(img, gray, CV_BGR2GRAY);
```

```
cvtColor(gray, gray, CV_GRAY2BGR);
```

```
iplimggr = cvCloneImage(&(IplImage)gray);
```

```
iplimgt = cvCreateImage(cvSize((int)(iplimggr->width*0.5),
(int)(iplimggr->height*0.5)), iplimggr->depth, iplimggr->
>nChannels);
```

# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Image -> PictureBox

```
cvResize(iplimggr, iplimgt);
pbsrc->Image = gcnew
System::Drawing::Bitmap(iplimgt->width, iplimgt->height,
iplimgt->widthStep,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(System::IntPtr) iplimgt->imageData);
pbsrc->Refresh();
}
```

# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Image -> PictureBox

- Nhấn nút Exit thoát chương trình

```
private: System::Void btnProcess_Click(System::Object^
sender, System::EventArgs^ e) {
```

```
cvReleaseImage(&iplimg); cvReleaseImage(&iplimggr);
```

```
destroyAllWindows();
```

```
this->Close();}
```

```
private: char* convertstring2char(System::String^ str)
```

```
{
```

```
char* str2 =
```

```
(char*)(void*)Marshal::StringToHGlobalAnsi(str);
```

```
return str2;}
```

# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Camera->PictureBox

- Tạo form có nút nhấn btStart, btExit, PictureBox1 và Timer1

```
//dưới #pragma once
```

```
//Tắt warning khi dùng opencv
```

```
#pragma managed(push,off)
```

```
#include <opencv2/opencv.hpp>
```

```
#pragma managed(pop)//dưới using namespace
```

```
System::Drawing;
```

```
using namespace cv;
```

```
VideoCapture cap; Mat frame;
```

```
IplImage* img;
```

# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Camera->PictureBox

```
private: System::Void timer1_Tick(System::Object^ sender,
System::EventArgs^ e) { cap >> frame;
img = cvCloneImage(&(IplImage)frame);
pictureBox1->Image = gcnew System::Drawing::Bitmap
(img->width, img->height, img->widthStep,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
(System::IntPtr) img->imageData);
pictureBox1->Refresh();
}
```

# Tạo ứng dụng OpenCV 341 trên WinForm VS 2015 C++ Camera->PictureBox

```
private: System::Void MyForm_Load(System::Object^
sender, System::EventArgs^ e) {
 cap.open(0); //mở camera
}
```

```
private: System::Void btStart_Click(System::Object^
sender, System::EventArgs^ e) {
 timer1->Start();} //Timer chạy với chu kỳ 30ms
```

```
private: System::Void btExit_Click(System::Object^
sender, System::EventArgs^ e) {
 this->Close();// Thoát
}
```

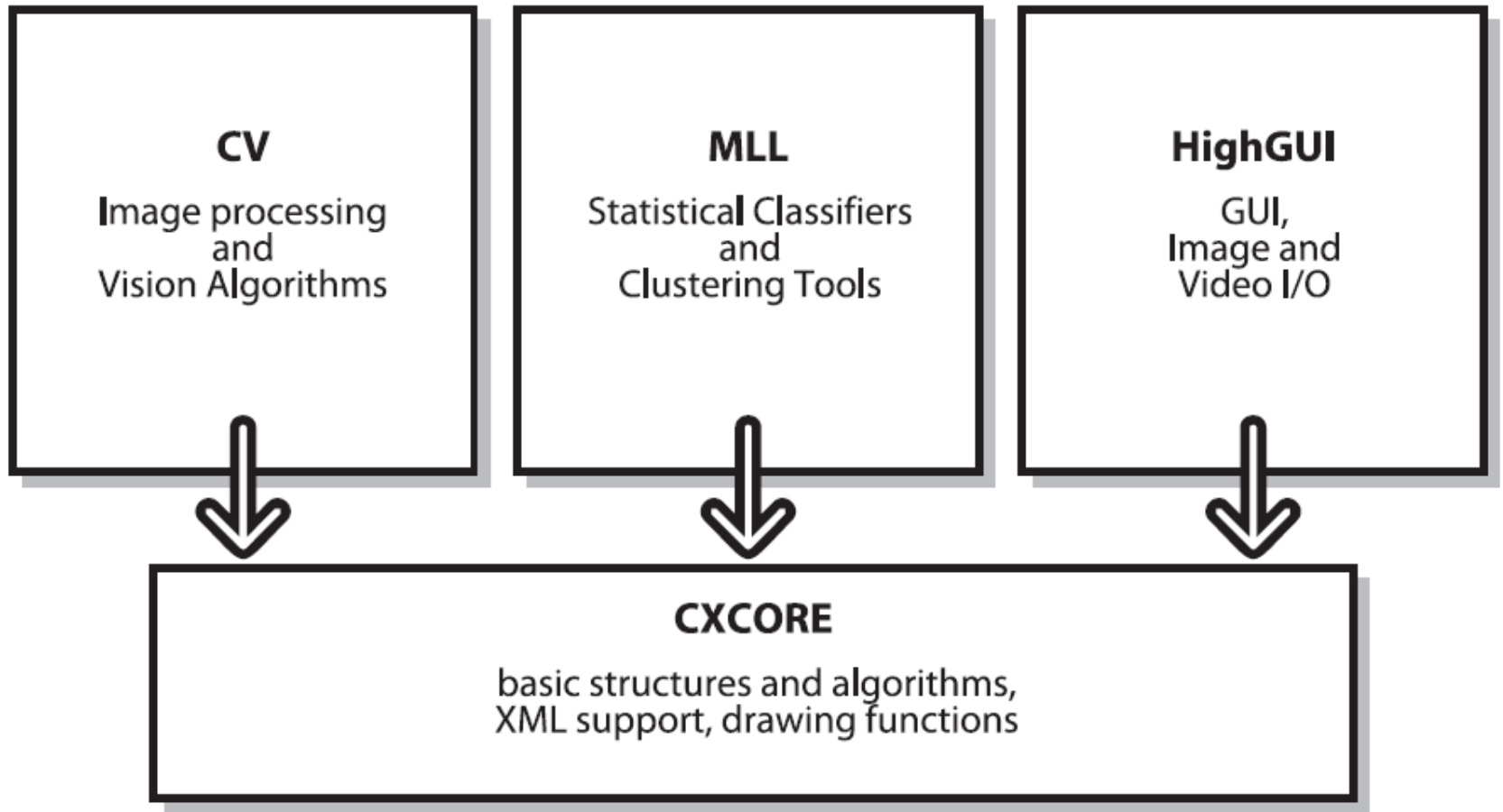
# Cấu trúc của OpenCV

**CVAUX:** Được mô tả trong văn bản của OpenCV như là module cũ và chỉ dùng để thí nghiệm. Tuy nhiên, giao diện đơn giản nhất cho nhận dạng mặt được nằm trong module này. Những mã nguồn nằm trong module này rất phù hợp cho việc nhận dạng mặt và chúng được sử dụng rộng rãi cho mục đích này.

**HIGHGUI:** Chứa các giao diện vào ra cơ bản, nó cũng chứa các khả năng cửa sổ mở rộng và vào ra video.

**CVCAM:** Chứa các giao diện cho video truy cập qua DirectX trên nền Windows 32 bits.

- **MACHINE LEARNING:** các thuật toán học





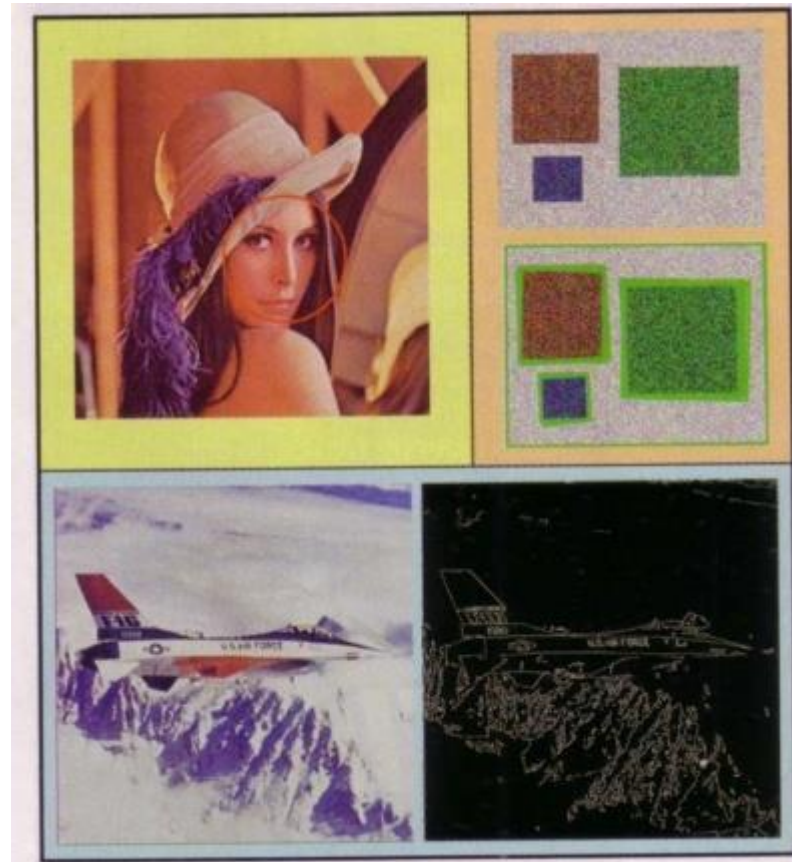
# CVAUX

- Eigen objects, a computationally efficient recognition technique that is, in essence, a template matching procedure
- 1D and 2D hidden Markov models, a statistical recognition technique solved by dynamic programming
- Embedded HMMs (the observations of a parent HMM are themselves HMMs)
- Gesture recognition from stereo vision support
- Extensions to Delaunay triangulation, sequences, and so forth
- Stereo vision
- Shape matching with region contours
- Texture descriptors
- Eye and mouth tracking

# CVAUX

- 3D tracking
- Finding skeletons (central lines) of objects in a scene
- Warping intermediate views between two camera views
- Background-foreground segmentation
- Video surveillance (see Wiki FAQ for more documentation)
- Calibration C++ classes (the C functions and engine are in CV)

Face  
Detection



Contour  
Detection

Edge  
Detection

# MỘT SỐ HÀM BẮT ẢNH VÀ CAMERA

- Cấu trúc ảnh: `IplImage` (`Ipl` là viết tắt của Image processing Library)

```
typedef struct _IplImage
{
int nSize;
int ID; // 0
int nChannels;
int alphaChannel; // không dùng
int depth;
char colorModel[4]; // OpenCV không sử dụng
char channelSeq[4]; // OpenCV không sử dụng
int dataOrder;
int origin;
int align; // OpenCV không sử dụng
int width;
int height;
struct _IplROI *roi;
struct _IplImage *maskROI; // bằng NULL trong OpenCV
```

```

void *imgeld; // bằng NULL trong OpenCV
struct _IplTileInfo *tileInfo; // bằng NULL trong OpenCV
int imageSize;
char *imageData;
int widthStep;
int BorderMode[4]; // OpenCV không sử dụng
int BorderConst[4]; // OpenCV không sử dụng
char *imageDataOrigin;
}
IplImage;

```

nSize : kích thước của ảnh

ID : chỉ số version, luôn bằng 0

nChannels : số kênh, các hàm của OpenCV đều hỗ trợ từ 1-4 kênh

alphaChannel : OpenCV không sử dụng

depth : chiều sâu của pixel theo bit, có các dạng sau :

IPL\_DEPTH\_8U : unsigned 8-bit integer

IPL\_DEPTH\_8S : signed 8-bit integer

IPL\_DEPTH\_16U : unsigned 16-bit integer

dataOrder : 0 = IPL\_DATA\_ORDER\_PIXEL các kênh màu đan xen nhau, 1 = các kênh màu tách rời

origin : gốc tọa độ 0 = top-left origin, 1 = bottom-left origin (kiểu Window bitmap)

width : bề rộng ảnh theo pixel

height : chiều cao ảnh theo pixel

roi Region of Interest (ROI) nếu bằng NULL thì chỉ có vùng ROI được xử lý

imageSize : kích thước dữ liệu ảnh theo byte

imageData : một char pointer trở tới vùng địa chỉ ảnh

widthStep : the size of an aligned image row, in bytes

# Create An Image

To draw a red square we'll need to start off by creating an image.

```
IplImage *img = cvCreateImage(cvSize(100, 100), IPL_DEPTH_8U, 3);
```

This creates an image of width/height 100/100, using 8-bit unsigned integers to represent the colour values, and with 3 colour channels.

However, 8-bit unsigned values are not the only type available; values can also be held as 32-bit floating point numbers (IPL\_DEPTH\_32F) and a variety of other ways. In each case the depth is represented as IPL\_DEPTH\_<bits>{U|S|F} where U, S and F stand for unsigned, signed and floating point. i.e.

- IPL\_DEPTH\_8U, IPL\_DEPTH\_8S
- IPL\_DEPTH\_16U, IPL\_DEPTH\_16S
- IPL\_DEPTH\_32S, IPL\_DEPTH\_32F
- IPL\_DEPTH\_64F
- Also notice that it's a pointer to an image - all images should be created in this way when using OpenCV as most (if not all) of its methods take image pointers as parameters in order to modify images directly.

# Image Data

- Images are not stored by pixel. Instead they are stored as arrays of colour levels which means less processor overhead as you're not constantly dereferencing pointers. These arrays of colour are stored in **BGR** order.
- e.g. `IplImage`'s `imageData` field looks like this...

| <code>imageData[0]</code> | <code>imageData[1]</code> | <code>imageData[2]</code> | <code>imageData[3]</code> | <code>imageData[4]</code> | <code>imageData[5]</code> |
|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|---------------------------|
| B                         | G                         | R                         | B                         | G                         | R                         |
| colour                    | values                    | go                        | in                        | these                     | elements                  |

- Finally, images in OpenCV are padded. Most image formats available today such as JPEG, PNG, TIFF and the like are padded out so that the number of columns in an image is divisible by 4 - with the exception of BMPs.



## Direct Pixel Access

So to get our red square going we'll just have to edit every third channel. Direct access of the pixels is possible using the `imageData` attribute and the number of bytes in the image (or `img->imageSize`) can be used as a quick way of bounding the for loop.

```
img->imageData[i] = value;
```

so we get:

```
int i;
```

```
for (i = 2; i < img->imageSize; i+=3)
```

```
 img->imageData[i] = 255;
```

It is worth noting that while most images and methods in OpenCV use or return 8-bit unsigned data (e.g. `cvLoadImage` always returns an `IPL_DEPTH_8U` image), this is not how OpenCV is written. `imageData` isn't *int* or *float*, it's actually a *char* pointer to data within `IplImage`.

## FLOAT DATA

- 32F images can only hold values between 0 and 1, so we have to adjust values accordingly. We also have to change the way for loops are defined - *imageSize* is measured in bytes and as there are now four bytes per colour value (floats are **four** bytes each), Instead we can use the image's *width* and *height* attributes, multiplying by 3 so that all channels are filled. Finally, the values themselves need to be converted to float pointers so that the data is stored in the correct format. The following code should clarify things.

```
int i;
for (i = 0; i < img->width*img->height*3; i+=3)
{
 ((float*)img->imageData)[i] = 64/256.0;
 ((float*)img->imageData)[i+1] = 196/256.0;
 ((float*)img->imageData)[i+2] = 256/256.0;
}
```

## Gía trị điểm ảnh

Cho ảnh RGB, ảnh chứa trong bộ nhớ theo thứ tự BGR

```
IplImage* img = cvLoadImage ("C:/anh.JPG");
```

Img là pointer trỏ đến địa chỉ ô nhớ chứa pixel góc trên trái của ảnh

Cho ảnh img, một điểm ảnh có tọa độ i hàng, j cột, chứa trong bộ nhớ ở địa chỉ  $u = i * \text{widthStep} + j * \text{nChannels} + k$

k=0: màu xanh b

k=1: màu xanh lá g

k=2: màu đỏ r

Gía trị một điểm màu 3 kênh là `imageData[i]` có thể đọc hay ghi

VÍ DỤ cho thành phần thứ nhất của điểm ảnh tọa độ (i,j) là 0

```
Img->imageData[i* img->widthStep+j* Img->nChannels]=0
```

# TÁC ĐỘNG ĐẾN PIXEL

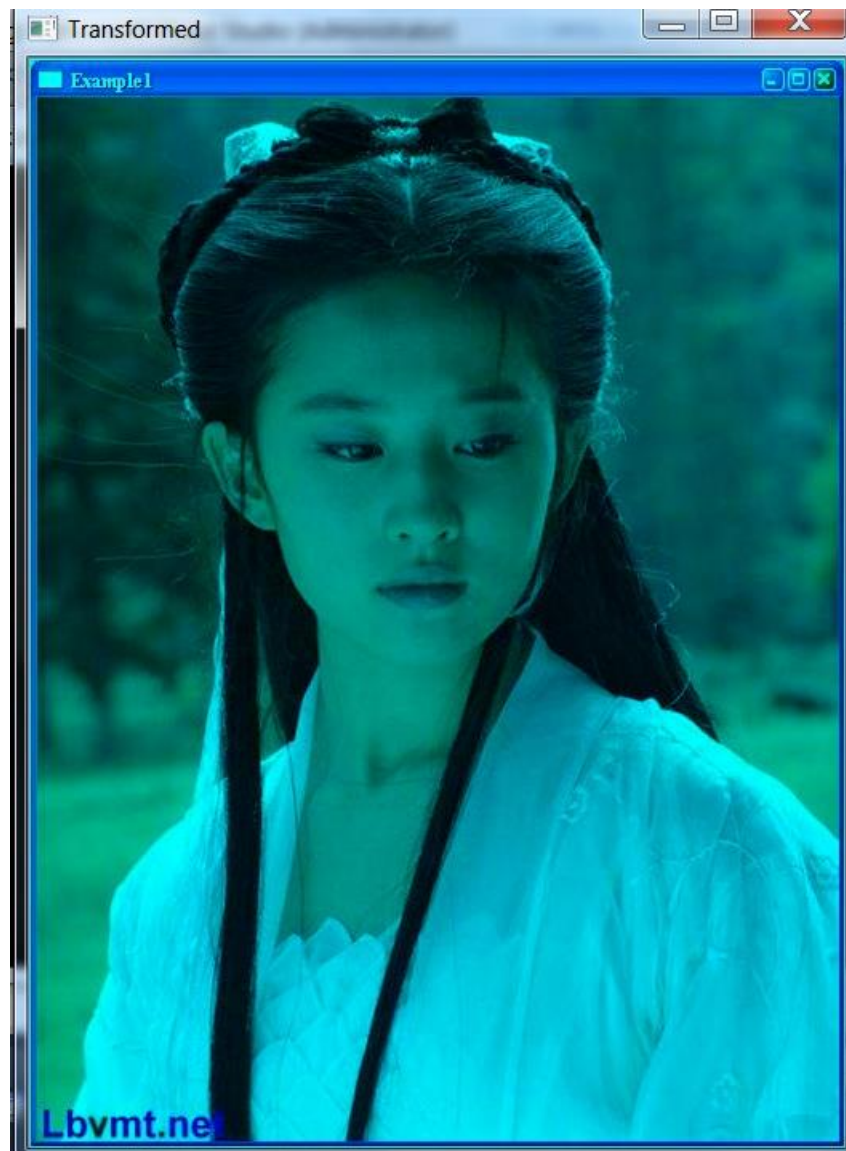
```
// ACCESS PIXEL.cpp : Defines the entry point for the console application.
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
using namespace std;
int main() {
 IplImage* img =cvLoadImage ("C:/tieulongnu.JPG");
 IplImage* img1=cvCloneImage(img);
 for(int y=0; y<img->height; y++) {
 uchar* ptr = (uchar*) (
 img->imageData + y * img->widthStep);
 for(int x=0; x<img->width; x++) {
 ptr[3*x+1] = 255;
 ptr[3*x+2] = 255;
 }
 }
```

## Hàm cvFlip

```
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
using namespace std;
int main() {
 IplImage* img =cvLoadImage ("C:/tieulongnu.JPG");
 IplImage* img1=cvCloneImage(img);
 cvFlip(img,img1,0);
 cvShowImage("ORG",img); //show the original image
 cvShowImage("FLIP",img1);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvReleaseImage(&img1);
 cvDestroyAllWindows();
}
```

# BIẾN ĐỔI MỘT KÊNH MÀU

```
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
using namespace std;
int main() {
 IplImage* img =cvLoadImage ("C:/tieulongnu.JPG");
 for(int y=0; y<img->height; y++)
 {
 for(int x=0; x<img->width; x++)
 {
 img->imageData[y * img->widthStep+x*img->nChannels+2]=0;
 }
 }
 cvShowImage("Transformed",img);
 cvWaitKey();
 cvDestroyAllWindows();
```



TS NGUYỄN ĐỨC THÀNH

# ĐỌC VÀ GHI ẢNH

- Việc đọc một file ảnh, ta gọi hàm *cvLoadImage()*, *OpenCV* hỗ trợ hầu hết các định dạng phổ biến như JPEG, PNG và BMP.

*IplImage\* cvLoadImage(const char\* filename, int iscolor = CV\_LOAD\_IMAGE\_COLOR);*

- Việc ghi một ảnh vào file ta gọi hàm *cvSaveImage()*, hàm này quyết định định dạng file ta sử dụng từ file ban đầu.

*Cả cvLoadImage() và cvSaveImage() đều nằm trong modul HighGUI.*

*Int cvSaveImage(const char\* filename, const CvArr\* image);*

- Khi chúng ta đã kết thúc quá trình nhập ảnh và sử dụng xong hàm *cvLoadImage()*, chúng ta cần gọi hàm *cvReleaseImage()* để giải phóng vùng nhớ cho ảnh.

– *Void cvReleaseImage( IplImage\*\* image);*

Để tạo một hình ảnh ta dùng hàm *cvCreateImage()*, hàm trả về một pointer trỏ đến hình ảnh có cấu trúc *IplImage*

*IplImage\* cvCreateImage(CvSize size, int depth, int channels);*



# ĐỌC VÀ HIỆN ẢNH

- Để Copy ảnh ta dùng hàm *CvCopy()*

*void cvCopy( const CvArr\* src, CvArr\* dst, const CvArr\* mask = NULL);*

Trong đó :

src : ảnh nguồn

dst : ảnh đích

mask : mặt nạ phủ lên ảnh nguồn nếu ta chỉ muốn copy một phần ảnh.

- Ví dụ:

```
#include "highgui.h"

int main(int argc, char** argv) {
 IplImage* img = cvLoadImage(argv[1]);
 cvNamedWindow("Example1", CV_WINDOW_AUTOSIZE);
 cvShowImage("Example1", img);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvDestroyWindow("Example1");
}
```

# ĐỌC VÀ HIỆN ẢNH

```
#include "stdafx.h"
#include <highgui.h>
#include <cv.h>
int main()
{
 IplImage* img =cvLoadImage("C:/SDC16361.JPG");
 cvNamedWindow("Example1", CV_WINDOW_AUTOSIZE);
 cvShowImage("Example1", img);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvDestroyWindow("Example1");
}
```

## ĐỌC ẢNH DÙNG Mat

- **C++:** Mat **imread**( const string& **filename**, int **flags**=1 )  
**filename** – Name of file to be loaded.  
**flags** – Flags specifying the color type of a loaded image:
  - CV\_LOAD\_IMAGE\_ANYDEPTH - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.
  - CV\_LOAD\_IMAGE\_COLOR - If set, always convert image to the color one
  - CV\_LOAD\_IMAGE\_GRAYSCALE - If set, always convert image to the grayscale one
  - **>0** Return a 3-channel color image.
  - **=0** Return a grayscale image.
- **Note:** In the current implementation the alpha channel, if any, is stripped from the output image. Use negative value if you need the alpha channel.
  - **<0** Return the loaded image as is (with alpha channel).

## Mat structure

```
class CV_EXPORTS Mat
{
public:
 // ... a lot of methods ...
 /*! includes several bit-fields:
 - the magic signature
 - continuity flag
 - depth
 - number of channels
 */
 int flags;
 /*! the array dimensionality, ≥ 2
 int dims;
 /*! the number of rows and columns or (-1, -1) when the array has more than 2
 dimensions
 int rows, cols;
 /*! pointer to the data
 uchar* data;
 /*! pointer to the reference counter;
 // when array points to user-allocated data, the pointer is NULL
 int* refcount;
 // other members };
```

# namedWindow

Creates a window.

C++: `void namedWindow(const string& winname, int flags=WINDOW_AUTOSIZE )`

Python: `cv2.namedWindow(winname[, flags]) → None`

C: `int cvNamedWindow(const char* name, int flags=CV_WINDOW_AUTOSIZE )`

Python: `cv.NamedWindow(name, flags=CV_WINDOW_AUTOSIZE) → None`

# namedWindow

## Parameters:

`name` – Name of the window in the window caption that may be used as a window identifier.

`flags` –

`WINDOW_NORMAL` If this is set, the user can resize the window (no constraint).

`WINDOW_AUTOSIZE` If this is set, the window size is automatically adjusted to fit the displayed image (see `imshow()` ), and you cannot change the window size manually.

`WINDOW_OPENGL` If this is set, the window will be created with OpenGL support.

# imshow

Displays an image in the specified window.

C++: `void imshow(const string& winname, InputArray mat)`

Python: `cv2.imshow(winname, mat) → None`

C: `void cvShowImage(const char* name, const CvArr* image)`

Python: `cv.ShowImage(name, image) → None`

Parameters:

`winname` – Name of the window.

`image` – Image to be shown.

# Đọc và hiện ảnh dạng MAT

```
#include "stdafx.h "
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main(int argc, char** argv)
{ if(argc != 2)
 { cout <<" Usage: display_image ImageToLoadAndDisplay" <<
endl;
 return -1;
 }
 Mat image;
 image = imread(argv[1], CV_LOAD_IMAGE_COLOR); // Read the
file
```



# Đọc và hiện ảnh dạng MAT

```
if(! image.data) // Check for invalid input
{
 cout << "Could not open or find the image" <<
std::endl ;
 return -1;
}
namedWindow("Display window",
WINDOW_AUTOSIZE);// Create a window for display.
imshow("Display window", image); // Show our image
inside it.
waitKey(0); // Wait for a keystroke in the window
return 0;
}
```

- Mat To Mat

Mat a;

Mat b = a; //“copy” a to b

Mat c(a); //“copy” a to c

- Mat To IplImage

Mat M; Mat gray;

IplImage img = M;

IplImage img = IplImage(M);

IplImage\* gray2 = cvCloneImage(&(IplImage)gray);

# ĐỔI RA ẢNH XÁM (IPL)

```
#include "stdafx.h"
#include "cv.h"
#include "highgui.h"
int main() {
 IplImage* img = cvLoadImage("C:/tieulongnu.jpg");
 IplImage* grayimg =cvCreateImage(cvGetSize(img), 8, 1);
 cvCvtColor(img, grayimg, CV_BGR2GRAY);
 cvShowImage("ANH MAU", img);
 cvShowImage("ANH XAM", grayimg);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvReleaseImage(&grayimg);
}
```

# ĐỔI RA ẢNH XÁM (MAT)

```
#include "stdafx.h"
#include "opencv2/core/core.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "iostream"
using namespace cv;
using namespace std;
int main()
{ Mat image;
image = imread("c:/jlo.jpg", CV_LOAD_IMAGE_COLOR);
if (!image.data)
{ cout << "Could not open or find the image" << std::endl;
```

# ĐỔI RA ẢNH XÁM (MAT)

```
return -1;}
```

```
// Create a new matrix to hold the gray image
```

```
Mat gray;
```

```
// convert RGB image to gray
```

```
cvtColor(image, gray, CV_BGR2GRAY);
```

```
namedWindow("Display window",
CV_WINDOW_AUTOSIZE);
```

```
imshow("Display window", image);
```

```
imwrite("d:/Gray_Image.jpg", gray);//save to disk
```

```
namedWindow("Result window",
CV_WINDOW_AUTOSIZE);
```

```
imshow("Result window", gray);
```

```
waitKey(0);return 0;
```

```
}
```

# Đổi ra ảnh xám không dùng lệnh cvtColor

```
img = cv::imread("jlo.jpg");
Mat img = imread("c:/jlo.jpg");
 Mat gr(img.rows, img.cols, CV_8U, Scalar(0));
 for (int row = 0; row < img.rows; row++)
 {for (int col = 0; col < img.cols; col++)
 {Vec3b p = img.at<Vec3b>(row, col);
 int gr_int = 0.1140*p[0] + 0.5870* p[1] +
0.2989*p[2];
 gr.at<uchar>(row, col) = gr_int;
 }
 }
 imshow("Result window", gr);
 waitKey(0); return 0;
```

# THAY KÍCH THƯỚC ẢNH

```
#include "stdafx.h"
#include "cv.h"
#include "highgui.h"
int main() {
 IplImage* img = cvLoadImage("C:/tieulongnu.jpg");
 IplImage* resizeimg =cvCreateImage(cvSize(img->width/2,img-
 >height/2), 8, 3);
 cvResize(img,resizeimg,1);
 cvShowImage("ANH GOC", img);
 cvShowImage("ANH RESIZE", resizeimg);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvReleaseImage(&resizeimg);
}
```

# FLIP

```
// flipvert2.cpp : Defines the entry point for the console application.
#include "cv.h"
#include "highgui.h"
#include "stdafx.h"
void Flip(IplImage* img1, IplImage* img2)
{
 int height = img1->height;//số dòng
 int width = img1->width; //số cột
 int channels = img1->nChannels;//số kênh màu
 int step = img1->widthStep / sizeof(uchar); //số byte mỗi hàng
 for (int i = 0; i < width; i++)
 {
 for (int j = 0; j < height; j++)//hàng j cột i
 {
```



# FLIP

```
for (int k = 0; k < channels; k++)
{
 img2->imageData[j*step + i*channels + k] =
 img1->imageData[(height - j - 1)*step +
 i*channels + k];
}
```

```
}
```

```
}
```

```
}
```

```
void main()
```

```
{
```

```
int i, j, k;
```

```
const char* window1 = "Picture Color 1";
```

```
const char* window2 = "Picture Color 2";
```

```
IplImage* img1 = cvLoadImage("c:/tieu long nu.jpg");
```

## FLIP

```
IpImage* img2 = cvCloneImage(img1);
Flip(img1, img2); //cvFlip(img_Color, img_Color_2, flip_mode);
cvShowImage("window1", img1);
cvShowImage("window2", img2);
cvWaitKey(0);
cvDestroyAllWindows();
}
```

# ĐỌC FILE VIDEO

```
#include "highgui.h"
int main()
{
 cvNamedWindow("Example2", CV_WINDOW_AUTOSIZE);
 CvCapture* capture = cvCreateFileCapture("Doraemon.avi");
 IplImage* frame;
 while(1)
 {
 frame = cvQueryFrame(capture);
 if(!frame) break;
 cvShowImage("Example2", frame);
 char c = cvWaitKey(33);
 if(c == 27) break; }
 cvReleaseCapture(&capture);
 cvDestroyWindow("Example2");
}
```

# WEBCAM CAPTURE 1 C

```
#include "stdafx.h"
#include <highgui.h>
int main() {
 CvCapture* capture = cvCaptureFromCAM(0); //trong highgui
 IplImage* src = NULL;
 cvNamedWindow("Webcam", CV_WINDOW_AUTOSIZE);
 while(1)
 {
 src = cvQueryFrame(capture);
 if(!src) break;
 cvShowImage("Webcam", src);
 char c = cvWaitKey(33);
 if (c == 27)
 {
 break;
 }
 }
}
```

# WEBCAM CAPTURE 1

```
}
 // cvReleaseImage(&src);
 //cvDestroyWindow("Webcam");
cvReleaseCapture(&capture);
 return 0;
}
```

Hàm cvCreateCameraCapture() , cvCaptureFromCAM( 0) hoạt động tương tự hàm cvCreateFileCapture()  
Tham số là ID của camera, thường là 0

# WEBCAM CAPTURE 2

```
#include <iostream>
//#include "stdio.h"
#include <cv.h>
#include <highgui.h>
using namespace std;
int main() {
 CvCapture* capture = cvCaptureFromCAM(0); //trong highgui
 if (!capture){
 cout << "Cannot Capture" << endl;
 return -1;
 }
 IplImage* src = NULL;
 cvNamedWindow("Webcam", CV_WINDOW_AUTOSIZE);
 while(1)
 {
 src = cvQueryFrame(capture);
 TS NGUYỄN ĐỨC THÀNH
```

# WEBCAM CAPTURE 2

```
// continue if not frame, wait for camera ready
if(!src){
cout << "Can't Capture Frame, Please Wait "<< endl; continue; }
cvShowImage("Webcam", src);
char c = cvWaitKey(33);
 if (c == 27)
 { cout << "Pressed Escape "<< endl;
 break; }
 }
cvReleaseCapture(&capture);
return 0;
}
```

## WEBCAM CAPTURE 3 C++

```
#include "opencv2/opencv.hpp"
#include <iostream> // std::cout
using namespace cv;
using namespace std;
int main()
{
 VideoCapture cap(0); // open the default camera
 // check if we succeeded capture
 if(! cap.isOpened()){
 cout << "failed to capure" << std::endl;return - 1;}
 namedWindow(" frame" , 1);
 for(;;)
 {
 Mat frame;
 bool bSuccess = cap.read(frame); // read a new frame from video
```



## WEBCAM CAPTURE 3 C++

```
if (!bSuccess) //if not success, continue loop, wait for camera ready
{
 cout << "Cannot read a frame from video file" << endl;
 continue;
}
imshow("frame" , frame);
if(waitKey(30) >= 0) break;
}
// the camera will be deinitialized automatically in VideoCapture
destructor
return 0;
}
```

# WEBCAM CAPTURE 4 CV3.4

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <iostream>
int main() {
cv::namedWindow("Example", cv::WINDOW_AUTOSIZE);
cv::VideoCapture cap;
cap.open(1); // open the second camera
if(!cap.isOpened()) { // check if we succeeded
std::cerr << "Couldn't open capture." << std::endl;return -1;
}
cv::Mat frame;
for(;;) {
cap >> frame; if(frame.empty()) break; // Ran out of film
cv::imshow("Example", frame);if(cv::waitKey(20) >= 0) break;
} return 0;}
```

# WEBCAM CAPTURE 4 CV3.4

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main() {
 namedWindow("Example", cv::WINDOW_AUTOSIZE);
 VideoCapture cap; cap.open(1); // open the first camera
 if(!cap.isOpened()) { // check if we succeeded
 cerr << "Couldn't open capture." << endl; return -1;
 }
 Mat frame; for(;;) { cap >> frame;
 if(frame.empty()) break; // Ran out of film
 imshow("Example", frame);
 if(waitKey(20) >= 0) break; } return 0;}
```

# DỮ LIỆU SƠ KHAI CỦA OPENCV2 C

- `CvPoint(x,y)`: xác định một điểm , x cột, y hàng là số nguyên
- `CvPoint2D32f(x,y)`: xy là số thực
- `CvPoint3D32f(x,y,z)`: xyz là số thực
- `CvSize (width, height)`: xác định kích thước hình
- `CvSize2D32f(width, height)`: tham số là số thực
- `CvRect(x,y,w,h)`: xác định hình chữ nhật có góc trên trái ở tọa độ x hàng y cột, rộng w và cao h
- `CvScalar(double val[4])` chứa 4 số thực trong val là các thành phần RGB A, với ảnh xám cường độ sáng chứa trong val(0), với ảnh màu B trong val[0], G trong val[1], R trong val[2],

## Hàm tạo ma trận C

- `cvMat* cvCreateMat ( int rows, int cols, int type );` type cho biết loại dữ liệu của phần tử ma trận
- `CV_<bit_depth>(S|U|F)C<number_of_channels>` Thus, the matrix could consist of 32-bit floats (`CV_32FC1`), of unsigned integer 8-bit triplets (`CV_8UC3`)  
`cvGetSize(CvMat*)`: cho kích thước ma trận
- `cvCloneMat(CvMat*)`: tạo ma trận giống ma trận có sẵn
- `cvReleaseMat(CvMat**)`: xóa bộ nhớ
- `cvInitMatHeader`: tạo ma trận với dữ liệu cho trước
- `// Create an OpenCV Matrix containing some fixed data.`  
`//bốn phần tử`

```
float vals[] = { 0.866025, -0.500000, 0.500000, 0.866025 };
```

```
CvMat rotmat;
```

```
cvInitMatHeader(
```

# MA TRẬN

```
&rotmat,
2,
2,
CV_32FC1,
vals
);
```

- `cvGetElemType( const CvArr* arr )`: loại dữ liệu trong ma trận
- `cvGetDims( const CvArr* arr, int* sizes=NULL )`: số chiều ma trận
- `cvGetDimSize( const CvArr* arr, int index )`. Kích thước của một chiều xác định
- Lấy ra một phần tử của ma trận:  

```
CvMat* matran = cvCreateMat(5, 5, CV_32FC1);
float element_3_2 = CV_MAT_ELEM(*matran, float, 3, 2);
```

# MA TRẬN

- Đặt giá trị cho một phần tử

```
CvMat* matran = cvCreateMat(5, 5, CV_32FC1);
float element_3_2 = 7.7;
((float)CV_MAT_ELEM_PTR(*matran, 3, 2)) = element_3_2;
```

- Đọc giá trị một phần tử

```
double cvGetReal1D(const CvArr* arr, int idx0);
double cvGetReal2D(const CvArr* arr, int idx0, int idx1);
double cvGetReal3D(const CvArr* arr, int idx0, int idx1, int idx2);
double cvGetRealND(const CvArr* arr, int* idx);
CvScalar cvGet1D(const CvArr* arr, int idx0);
CvScalar cvGet2D(const CvArr* arr, int idx0, int idx1);
CvScalar cvGet3D(const CvArr* arr, int idx0, int idx1, int idx2);
CvScalar cvGetND(const CvArr* arr, int* idx);
```

# MA TRẬN

- Đặt giá trị một phần tử

```
void cvSetReal1D(CvArr* arr, int idx0, double value);
```

```
void cvSetReal2D(CvArr* arr, int idx0, int idx1, double value);
```

```
void cvSetReal3D(CvArr* arr, int idx0, int idx1, int idx2, double value);
```

```
void cvSetRealND(CvArr* arr, int* idx, double value);
```

```
void cvSet1D(CvArr* arr, int idx0, CvScalar value);
```

```
void cvSet2D(CvArr* arr, int idx0, int idx1, CvScalar value);
```

```
void cvSet3D(CvArr* arr, int idx0, int idx1, int idx2, CvScalar value);
```

```
void cvSetND(CvArr* arr, int* idx, CvScalar value);
```

- Với ma trận một chiều dùng các hàm

```
double cvmGet(const CvMat* mat, int row, int col)
```

```
void cvmSet(CvMat* mat, int row, int col, double value)
```



# MA TRẬN

- Cộng các phần tử ma trận 3 chiều

```
float sum(const CvMat* mat) {
 float s = 0.0f;
 for(int row=0; row<mat->rows; row++) {
 const float* ptr = (const float*)(mat->data.ptr + row * mat->step);
 for(col=0; col<mat->cols; col++) {
 s += *ptr++;
 }
 }
 return(s);
}
```

# ẢNH IplImage

- Ảnh có cấu trúc IplImage tựa như cvMat tuy nhiên có rất nhiều tham số khá phức tạp
- Có thể xử lý ảnh như với ma trận

`void cvSetImageROI( IplImage* image, CvRect rect );` lấy vùng ảnh quan tâm region of interest

`void cvResetImageROI( IplImage* image );`

Ví dụ thay đổi giá trị một vùng ảnh

```
// roi_add <image> <x> <y> <width> <height> <add>
```

```
#include <cv.h>
```

```
 #include <highgui.h>
```

```
 int main(int argc, char** argv)
```

```
 {
```

```
 IplImage* src;
```

# BIẾN ĐỔI VÙNG ẢNH

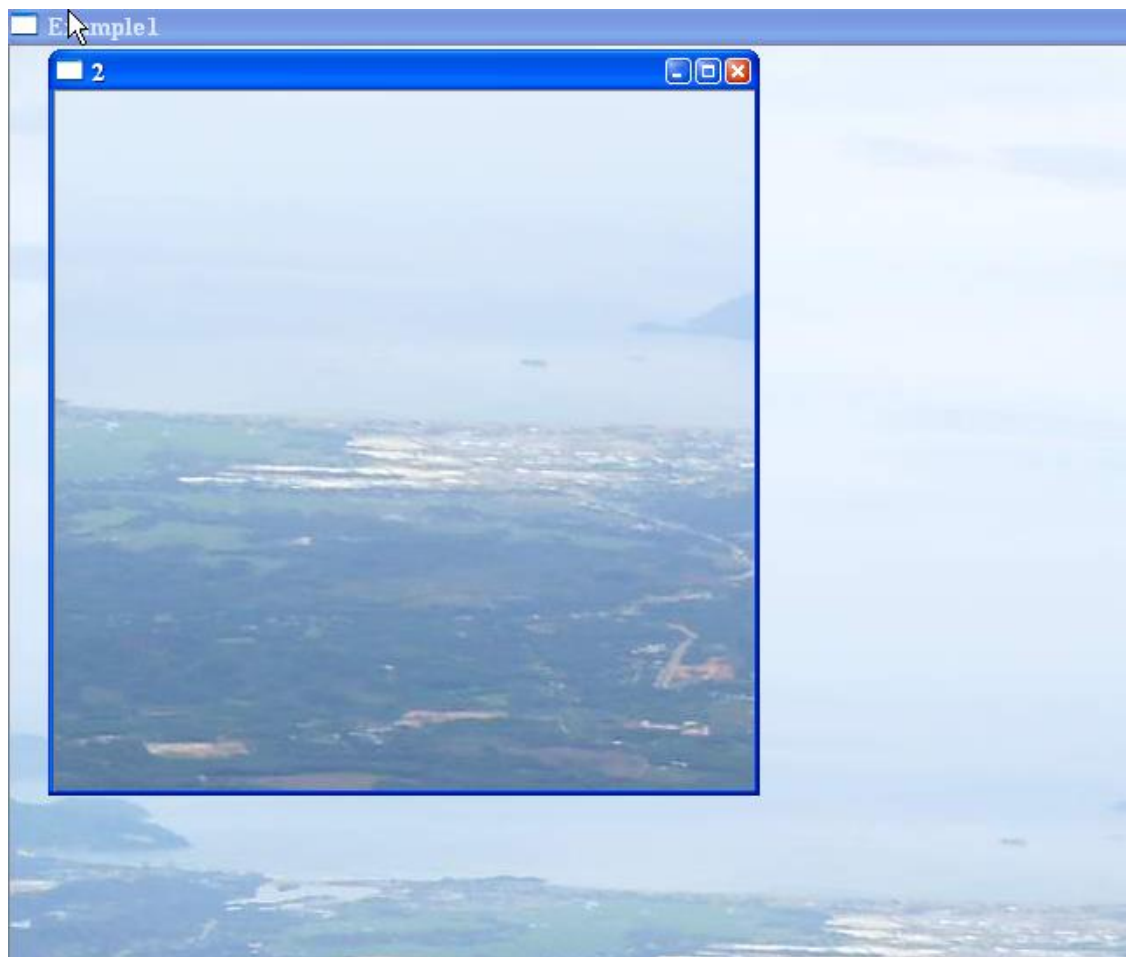


Cộng 150 vào kênh Blue trên vùng mặt của con mèo

TS NGUYỄN ĐỨC THÀNH

# HIỂN THỊ MỘT VÙNG ẢNH

```
#include "stdafx.h"
#include <highgui.h>
#include <cv.h>
int main()
{
 IplImage* img =cvLoadImage("C:/anh.JPG");
 cvShowImage("Example1", img);
 cvSetImageROI(img, cvRect (300,300, 400,400));//Chỉ quan tâm vùng
 chữ nhật , dùng cvResetImageROI(IplImage *image); để reset
 cvShowImage("2", img);
 cvWaitKey(0);
 cvReleaseImage(&img);
 cvDestroyAllWindows();
}
```



CvRect: Offset (usually the top-left corner) and size of a rectangle.

```
(
int x;
int y;
int width;
int height;
)
```

x x-coordinate of the top-left corner

y y-coordinate of the top-left corner (bottom-left for Windows bitmaps)

width Width of the rectangle

height Height of the rectangle

# CÁC HÀM MA TRẬN VÀ ẢNH OPENCV2

- cvAbs Absolute value of all elements in an array
- cvAbsDiff Absolute value of differences between two arrays
- cvAbsDiffS Absolute value of difference between an array and a scalar
- cvAdd Elementwise addition of two arrays
- cvAddS Elementwise addition of an array and a scalar
- cvAddWeighted Elementwise weighted addition of two arrays (alpha blending)
- cvAvg Average value of all elements in an array
- cvAvgSdv Absolute value and standard deviation of all elements in an array
- cvCalcCovarMatrix Compute covariance of a set of *n-dimensional vectors*
- cvCmp Apply selected comparison operator to all elements in two arrays

# CÁC HÀM MA TRẬN VÀ ẢNH OPENCV2

- `cvCmpS` Apply selected comparison operator to an array relative to a scalar
- `cvConvertScale` Convert array type with optional rescaling of the value
- `cvConvertScaleAbs` Convert array type after absolute value with optional rescaling
- `cvCopy` Copy elements of one array to another
- `cvCountNonZero` Count nonzero elements in an array
- `cvCrossProduct` Compute cross product of two three-dimensional vectors
- `cvCvtColor` Convert channels of an array from one color space to another
- `cvDet` Compute determinant of a square matrix
- `cvDiv` Elementwise division of one array by another



# CÁC HÀM MA TRẬN VÀ ẢNH OPENCV2

- `cvDotProduct` Compute dot product of two vectors
- `cvEigenVV` Compute eigenvalues and eigenvectors of a square matrix
- `cvFlip` Flip an array about a selected axis
- `cvGEMM` Generalized matrix multiplication
- `cvGetCol` Copy elements from column slice of an array
- `cvGetCols` Copy elements from multiple adjacent columns of an array
- `cvGetDiag` Copy elements from an array diagonal
- `cvGetDims` Return the number of dimensions of an array
- `cvGetDimSize` Return the sizes of all dimensions of an array

# CÁC HÀM MA TRẬN VÀ ẢNH OPENCV2

- `cvGetRow` Copy elements from row slice of an array
- `cvGetRows` Copy elements from multiple adjacent rows of an array
- `cvGetSize` Get size of a two-dimensional array and return as `CvSize`
- `cvGetSubRect` Copy elements from subregion of an array
- `cvInRange` Test if elements of an array are within values of two other arrays
- `cvInRangeS` Test if elements of an array are in range between two scalars
- `cvInvert` Invert a square matrix

# CÁC HÀM MA TRẬN VÀ ẢNH OPENCV2

- `cvInRangeS` Checks that array elements lie between two scalars.
- `void cvInRangeS(const CvArr* src, CvScalar lower, CvScalar upper,`
- `CvArr * dst);`
  - `src` The first source array
  - `lower` The inclusive lower boundary
  - `upper` The exclusive upper boundary
  - `dst` The destination array, must have 8u or 8s type
  - The function does the range check for every element of the input array:
  - $\text{dst}(I) = \text{lower}_0 \leq \text{src}(I)_0 < \text{upper}_0$
  - For single-channel arrays,
  - $\text{dst}(I) = \text{lower}_0 \leq \text{src}(I)_0 < \text{upper}_0 \wedge \text{lower}_1 \leq \text{src}(I)_1 < \text{upper}_1$
  - For two-channel arrays and so forth,
  - '`dst(I)`' is set to 0xff (all 1-bits) if '`src(I)`' is within the range and 0 otherwise. All the arrays must have the same size (or ROI size).

# **MỘT SỐ KIẾN THỨC CƠ BẢN VỀ OPENCV 3.x DÙNG C, C++, Python**

**[https://docs.opencv.org/3.4.1/index.  
html](https://docs.opencv.org/3.4.1/index.html)**

# CẤU TRÚC

The Core Functionality (core module)

Image Processing (imgproc module)

High Level GUI and Media (highgui module)

Image Input and Output (imgcodecs module)

Video Input and Output (videoio module)

Camera calibration and 3D reconstruction (calib3d module)

2D Features framework (feature2d module)

Video analysis (video module)

Object Detection (objdetect module)

Deep Neural Networks (dnn module)

Machine Learning (ml module)

Computational photography (photo module)

Images stitching (stitching module)

GPU-Accelerated Computer Vision (cuda module)

# CÁC CẤU TRÚC CƠ BẢN

- Open CV dùng kết hợp với các ngôn ngữ như C, C++, Python...
- Opencv C có hàm bắt đầu bởi `cv`, ảnh có hai loại là `IplImage` và `CvMat`
- Opencv C++ có hàm bắt đầu bởi `cv::tên hàm`, ảnh có loại `Mat`
- Open CV 3. trở lên hỗ trợ cả C, C++, Python...
- Python dùng cấu trúc `cv2.tên hàm`
- Trong khai báo với Opencv C++ ta dùng
- `#include <opencv2/opencv.hpp>`
- `using namespace cv` // để khởi viết `cv::`
- Trong phần sau ta sẽ bỏ chữ `cv::` khi dùng các lệnh C++

# CÁC CẤU TRÚC CƠ BẢN

- **Data types:** bool, unsigned char (0..255) (uchar), signed char (-128..127), unsigned short (số nguyên không dấu 2 byte), signed short, int (số nguyên 4 byte), float (số thực 4 byte), double (số thực 8 byte), complex số phức (khi dùng số phức khai báo `#include <complex>` )

- Ví dụ: `int sn = 12; float st = 130.7; complex<float> sp(12,-5);`

- **Point:** khai báo tọa độ 2 chiều có tọa độ (x,y) hay 3 chiều (x,y,z), các tọa độ là số nguyên (i), số thực (f) hay số thực 8 byte (f)

Point2i khai báo tọa độ số nguyên 2 chiều

Point2d khai báo tọa độ số nguyên kép 2 chiều

Point2f khai báo tọa độ số thực 2 chiều

Point3x khai báo tọa độ 3 chiều, x là i, d, hay f

**Ví dụ:**

```
#include "stdafx.h" //compile with /EHsc
```

```
#include <iostream>
```

```
#include <opencv2/opencv.hpp>
```

```
#include <complex>
```

```
using namespace cv; using namespace std; Point2d p1();
```

# CÁC CẤU TRÚC CƠ BẢN

```
int main(){
//khai báo điểm pt có tọa độ cột 32, hàng 100
Point2i pt=(32,100);
Point2f a(0.3f, 0.f), b(0.f, 0.4f); Point pt = (a + b)*10.f;
complex<double> sp(12.0, -5.0);
complex<double> spsq =pow(sp,2) ;
cout << pt.x << ", " << pt.y << endl;// xuất ra màn hình console
cout << " Complex Number sp:"<<real(sp) << ", " << imag(sp)
<<'\n';
cout << " Squared Complex Number:" << real(spsq) << ", " <<
imag(spsq) << '\n';
system("pause"); return(0);}
```



# CÁC CẤU TRÚC CƠ BẢN

`cx=a.x; //lấy tọa độ x của điểm a`

`cy=a.y; //lấy tọa độ y của điểm a`

`Point3i c(1,5,20); //các tọa độ là c.x, c.y, c.z`

**-Size:** khai báo kích thước Width và Height của ảnh

`Size sz(10, 20);`

`cout << sz.width << ", " << sz.height << endl;`

**-Scalar:** là một tập gồm từ một đến bốn phần tử, dùng để khai báo giá trị, truy cập theo chỉ số từ [0] đến [3]

`Scalar s1(5); Scalar(134, 29); s1[0]`

**-Vec :** khai báo vector 2,3,4,6 phần tử

`Vec{n}{type}` n: số phần tử, type: b,s, i,f,d

Ví dụ: `Vec4i vt( 12,4, 5, -4);`

Có thể đổi Vec sang Point hay Scalar

-Rec: khai báo hình chữ nhật (row, column, width, height)

# TẠO ẢNH MAT

Khai báo ảnh Mat với số hàng, số cột, loại dữ liệu, giá trị mỗi pixel, ảnh Mat có thể là hai chiều hay nhiều chiều. Mat là một ảnh nhưng cũng là ma trận, xử lý như với ma trận

| Constructor                                                                                                  | Description                                                           |
|--------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <code>cv::Mat;</code>                                                                                        | Default constructor                                                   |
| <code>cv::Mat( int rows, int cols, int type );</code>                                                        | Two-dimensional arrays by type                                        |
| <code>cv::Mat(<br/>    int rows, int cols, int type,<br/>    const Scalar&amp; s<br/>);</code>               | Two-dimensional arrays by type with initialization value              |
| <code>cv::Mat(<br/>    int rows, int cols, int type,<br/>    void* data, size_t step=AUTO_STEP<br/>);</code> | Two-dimensional arrays by type with preexisting data                  |
| <code>cv::Mat( cv::Size sz, int type );</code>                                                               | Two-dimensional arrays by type (size in sz)                           |
| <code>cv::Mat(<br/>    cv::Size sz,<br/>    int type, const Scalar&amp; s<br/>);</code>                      | Two-dimensional arrays by type with initialization value (size in sz) |

# TẠO ẢNH MAT

| Constructor                                                                                                                 | Description                                                       |
|-----------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| <pre>cv::Mat(<br/>    cv::Size sz, int type,<br/>    void* data, size_t step=AUTO_STEP<br/>);</pre>                         | Two-dimensional arrays by type with preexisting data (size in sz) |
| <pre>cv::Mat(<br/>    int ndims, const int* sizes,<br/>    int type<br/>);</pre>                                            | Multidimensional arrays by type                                   |
| <pre>cv::Mat(<br/>    int ndims, const int* sizes,<br/>    int type, const Scalar&amp; s<br/>);</pre>                       | Multidimensional arrays by type with initialization value         |
| <pre>cv::Mat(<br/>    int ndims, const int* sizes,<br/>    int type, void* data,<br/>    size_t step=AUTO_STEP<br/>);</pre> | Multidimensional arrays by type with preexisting data             |

# TẠO ẢNH MAT

- Type có các dạng sau

`CV_{8U,16S,16U,32S,32F,64F}C{1,2,3}`

Như vậy loại ảnh 8 bit không dấu một kênh xám là 8UC1, ảnh 8 bit không dấu ba kênh màu 8UC3, ảnh màu số thực là 32FC3... `CV_8UC(n)` (mảng n kênh 8 bit nguyên không dấu (n có thể từ 1 to 512) )

- Data là mảng có kích thước số hàng\* số cột \* số kênh
- Scalar là bộ 1 số cho đến 3 số tùy theo ảnh.
- Khai báo ảnh m 3 hàng 10 cột, loại số thực 32 bit có giá trị mỗi pixel cho ba kênh là (1,0,1)

```
Mat m(3, 10, CV_32FC3, cv::Scalar(1.0f, 0.0f, 1.0f));
```

```
Mat img(Size(480, 360), CV_64UC3); //480x360 3-channel
array with 64 bit unsigned integers
```

# OPENCV C++ Mat

Mat A, C; // Khởi tạo ảnh A, C chỉ có phần header

Mat B(A); // B có cấu trúc như A nhưng giá trị chưa có

Mat F=A.clone(); //giống Mat F(A)

C = A; // Assignment operator, gán giá trị của A cho C

A.copyTo(C); //chép A cho C, giống C = A

*Mat D (A, Rect(10, 20, 100, 200) );* // D là một phần của A, là hình chữ nhật góc trên trái tọa độ cột hàng (10,20), bề rộng 100 và cao 200

*Mat E = A(Range::all(), Range(1,3));* // lấy các cột từ 1 đến 3 của A

# TẠO ẢNH MAT

```
#include "stdafx.h"
#include <iostream>#include <opencv2/opencv.hpp>
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main()
{ //khai báo giá trị các pixel
int8_t data[16] = { 20, 20, 30, 40, 30, 50, 70, 60, 20, 200, 30,
10, 40, 100, 30, 70 };
// khai báo ảnh xám 4*4, nguyên
Mat D(4, 4, CV_8UC1, data);
cout << "img = \n" << " " << D << "\n\n";
```

# TẠO ẢNH MAT

```
uchar mybytes[]={1,255,0,4,1,4,76,43,45};
Mat mymat(3,3,CV_8UC1,mybytes);
Mat E, F;
//ảnh trắng 30*30
Mat img= Mat::ones(30, 30, CV_32F);
//phóng to đến kích thước ngang 200 cao 300
resize(D, E, Size(200, 300));
resize(img, F, Size(300, 200));
imshow("1 ", E);
imshow("2", img);
imshow("3", F);
waitKey(0);
return(0);}
```



# TẠO ẢNH MAT

```
Mat imgA = Mat::eye(4, 4, CV_64F); //đường chéo là 1
cout << "imgA = \n " << imgA << "\n\n";

Mat imgB = Mat::ones(2, 2, CV_32F); // tất cả là 1
Mat imgC = Mat::zeros(3,3, CV_8UC1); //tất cả là 0
//Khai báo ma trận 3 chiều nguyên kích thước rộng 2 cao 5 và
sâu 3, mỗi pixel một chiều, giá trị 0
int sz[3] = {2,5,3};
 Mat L(3,sz, CV_8UC(1), Scalar::all(0));
//Đổi ảnh Ipl sang Mat
IplImage* imgIpl = cvLoadImage("a.png", 1);
Mat imgMat = cvarrToMat(imgIpl);
//Đổi ảnh Mat sang Ipl
IplImage* imgIpl= &IplImage(imgMat);
```

# Đổi ảnh MAT <-> IplImage

```
using namespace cv;
using namespace std;
int main()
{Mat img = imread("c:/annachapman.jpg");
IplImage* imgIpl = &IplImage(img); //Mat to IplImage
Mat imgMat = cvarrToMat(imgIpl); //IplImage to Mat
imshow("1", img); //Show Mat Image
cvShowImage("2", imgIpl); //Show IplImage
imshow("3", imgMat);
waitKey(0);
return(0);}
```

# PyThon cv2

Python kết hợp các hàm numpy với các hàm opencv cv2, array cũng là ảnh

```
import cv2, numpy as np
```

```
#create random integer values from 0 to 199, size 300*500*3
```

```
img = np.random.randint(200, size=(300,500,3))
```

```
img1 = np.array(img ,dtype=np.uint8)
```

```
#Two above lines can be replaced by
```

```
#img1=np.array(np.random.randint(200, size=(300,500,3),
dtype=np.uint8))
```

```
zoom=cv2.resize(img1,(100,200));cv2.imshow('1',zoom)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

# PyThon cv2

Khai báo ma trận ảnh

```
img2=np.array([[20, 20, 30, 40],
 [30, 50, 70, 60],
 [20, 200, 30, 10],
 [40, 100, 30, 70]], dtype=np.uint8)
```

Khai báo các ma trận đặc biệt

```
a = np.zeros((2,2)) # ma trận 2*2 các số 0
b = np.ones((1,2)) # vector hàng 2 phần tử 1
c = np.full((2,2), 7) # ma trận 2*2 gồm số 7
d = np.eye(2) # ma trận đơn vị 2*2
```

# TRUY CẬP PIXEL OPENCV3

Lấy giá trị nguyên pixel hàng y cột x ảnh xám

```
Scalar intensity = img.at<uchar>(y, x);
```

```
Scalar intensity = img.at<uchar>(Point(x, y));
```

Lấy giá trị pixel ảnh màu

```
Vec3b intensity = img.at<Vec3b>(y, x);
```

```
uchar blue = intensity.val[0];
```

```
uchar green = intensity.val[1];
```

```
uchar red = intensity.val[2];
```

```
Vec3f intensity = img.at<Vec3f>(y, x);
```

```
float blue = intensity.val[0];
```

```
float green = intensity.val[1];
```

```
float red = intensity.val[2];
```

# TRUY CẬP PIXEL OPENCV3

Thay giá trị pixel

```
img.at<uchar>(y, x) = 128;
```

# ĐỌC ẢNH

C++: `Mat imread(const filename, flags=1 )`

Python: `cv2.imread(filename[, flags]) → retval`

C: `IplImage* cvLoadImage(const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR )`

C: `CvMat* cvLoadImageM(const char* filename, int iscolor=CV_LOAD_IMAGE_COLOR )`

Python: `cv.LoadImage(filename, iscolor=CV_LOAD_IMAGE_COLOR) → None`

Python: `cv.LoadImageM(filename, iscolor=CV_LOAD_IMAGE_COLOR) → None`

Parameters:

`filename` – Name of file to be loaded.

`flags` –

Flags specifying the color type of a loaded image:

# ĐỌC VÀ LƯU ẢNH

`CV_LOAD_IMAGE_ANYDEPTH` - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.

`CV_LOAD_IMAGE_COLOR` - If set, always convert image to the color one

`CV_LOAD_IMAGE_GRAYSCALE` - If set, always convert image to the grayscale one

**>0 Return a 3-channel color image.**

Note In the current implementation the alpha channel, if any, is stripped from the output image. Use negative value if you need the alpha channel.

**=0 Return a grayscale image.**

**<0 Return the loaded image as is (with alpha channel).**

**C++: `bool imwrite(filename, InputArray img)`**

**Python: `cv2.imwrite(filename, img) → retval`**

**C: `int cvSaveImage(filename, const CvArr* image)`**



# TẠO VÀ ĐÓNG CỬA SỔ ẢNH

C++: `void namedWindow( "winname", flags=WINDOW_AUTOSIZE )`

Python: `cv2.namedWindow("winname", flags)] → None`

C: `int cvNamedWindow("winname", flags )`

Python: `cv.NamedWindow("winname", flags) → None`

name – Name of the window in the window caption that may be used as a window identifier.

flags –Flags of the window số nguyên 0 hay 1 mặc định là 1

0: `WINDOW_NORMAL` có thể chỉnh kích thước cửa sổ

1: (Default) `WINDOW_AUTOSIZE`, cửa sổ thay đổi tùy kích thước ảnh, không chỉnh được

C++: `void destroyWindow( " winname")`

Python: `cv2.destroyWindow(winname) → None`

C: `void cvDestroyWindow("winname")`

Python: `cv.DestroyWindow("winname") → None`

C++: `void destroyAllWindows();` Python `cv2. destroyAllWindows()`

C : `void cvDestroyAllWindows(),` Python `cv.DestroyAllWindows(),`

# DI CHUYỂN CỬA SỔ

C++: `void moveWindow("winname", int x, int y)`

Python: `cv2.moveWindow("winname", x, y) → None`

C: `void cvMoveWindow("winname", int x, int y)`

Python: `cv.MoveWindow("winname", x, y) → None`

`winname` – Window name

`x` – Tọa độ x mới của cửa sổ , tính từ bên trái qua

`y` – Tọa độ y mới của cửa sổ , tính từ trên xuống

# HIỆN ẢNH

C++: void imshow("winname", Mat image)

Python: cv2.imshow("winname", Mat image) → None

C: void cvShowImage(("winname", IplImage image)

Python: cv.ShowImage(name, image) → None

//Chờ bấm phím

C++: int waitKey(int delay=0)

Python: cv2.waitKey([delay]) → retval

C: int cvWaitKey(int delay=0 )

Python: cv.WaitKey(delay=0) → int

# ĐỌC CAMERA, VIDEO C++

```
VideoCapture cap(0); // open the default camera
// VideoCapture cap(filename); //open video file
if(!cap.isOpened()) // check if we succeeded
 return -1;
for(;;)
{
 Mat frame;
 cap >> frame; // get a new frame from camera
 imshow("cam", frame);
 if(waitKey(30) >= 0) break;
}
```

# ĐỌC CAMERA VIDEO PYTHON

```
import numpy as np,cv2
cap = cv2.VideoCapture(0)
#cap = cv2.VideoCapture('filename')
while(True):
 ret, frame = cap.read()
 gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
 # Display the resulting frame
 cv2.imshow('frame',gray)
 if cv2.waitKey(1) & 0xFF == ord('q'):
 break
cap.release()
cv2.destroyAllWindows()
```

## LƯU VIDEO CAMERA

Tạo object cho VideoWriter, ví dụ đặt tên là vid

C++: `VideoWriter vid(filename, int fourcc, double fps, Size frameSize, bool isColor=true)`

Python: `vid=cv2.VideoWriter([filename, fourcc, fps, frameSize[, isColor]])`

*fourcc – mã 4 chữ nén video*

**0x7634706d ghi file mp4 có tỷ số nén cao nhất**

**CV\_FOURCC('W', 'M', 'V', '2') ghi file wmv**

**CV\_FOURCC ('M','J','P','G') ghi file avi có tỷ số nén không cao**

//Ghi frame vào object

C++: `vid.write(image)`

Python: `vid.write(image)`

# LU'U VIDEO CAMERA C++

```
#include "stdafx.h"
#include "opencv2/opencv.hpp"
using namespace cv;
int main(int, char**)
{
 VideoCapture cap(0);
 if (!cap.isOpened())return -1;
 int fr_w = cap.get(CV_CAP_PROP_FRAME_WIDTH);
 int fr_h = cap.get(CV_CAP_PROP_FRAME_HEIGHT);
 VideoWriter video1("d:/Result.mp4", 0x7634706d,
30, Size(fr_w, fr_h));
 VideoWriter video2("d:/Result.avi",
CV_FOURCC('M', 'J', 'P', 'G'), 30, Size(fr_w, fr_h));
 VideoWriter video3("d:/Result.wmv",
CV_FOURCC('W', 'M', 'V', '2'), 30, Size(fr_w, fr_h));

```

# LƯU VIDEO CAMERA C++

```
for (;;)
{
 Mat frame;
 cap >> frame;
 if (frame.empty())break;
 imshow("Vid", frame);
 video1.write(frame);
 video2.write(frame);
 if ((char)waitKey(30) == 27) break;
}
cap.release();
video1.release();
video2.release();
destroyAllWindows();
return 0;
}
```



# LU'U VIDEO CAMERA Python

```
#c:/python36/cam.py
```

```
import cv2, numpy as np
```

```
cap = cv2.VideoCapture(0)
```

```
if (cap.isOpened() == False): print("Unable to read camera ")
```

```
fps = cap.get(cv2.CAP_PROP_FPS)
```

```
fr_w = int(cap.get(3));fr_h = int(cap.get(4))
```

```
out =
```

```
cv2.VideoWriter('d:/outpy.wmv',cv2.VideoWriter_fourcc('W',
'M','V','2'), fps, (fr_w,fr_h))
```

```
out1 =
```

```
cv2.VideoWriter('outpy1.mp4',cv2.VideoWriter_fourcc(*'MP
4V'), 30, (fr_w,fr_h))
```

```
out2 =
```

```
cv2.VideoWriter('outpy2.avi',cv2.VideoWriter_fourcc(*'MJPG'), 30, (fr_w,fr_h))
```

# LƯU VIDEO CAMERA Python

```
while(True):
 ret, frame = cap.read()
 if ret == True:
 out.write(frame); out1.write(frame); out2.write(frame);
 cv2.imshow('frame',frame)
 if cv2.waitKey(1) & 0xFF == ord('q'):
 break
 else:
 break
cap.release();out.release(); out1.release(); out2.release();
cv2.destroyAllWindows()
```

# Line

cvLine Draws a line segment connecting two points.

```
void cvLine(CvArr * img,CvPoint pt1,CvPoint pt2, CvScalar color,
int thickness=1,
int lineType=8,
int shift=0);
```

- img The image
- pt1 First point of the line segment
- pt2 Second point of the line segment
- color Line color
- thickness Line thickness
- lineType Type of the line:
  - 8 (or omitted) 8-connected line.
  - 4 4-connected line.
  - CV\_AA antialiased line.
- shift Number of fractional bits in the point coordinates

# Line

The function draws the line segment between pt1 and pt2 points in the image. The line is clipped by the image or ROI rectangle. For non-antialiased lines with integer coordinates the 8-connected or 4-connected Bresenham algorithm is used.

Thick lines are drawn with rounding endings. Antialiased lines are drawn using Gaussian filtering.

To specify the line color, the user may use the macro `CV_RGB( r, g, b )`.

# Line

C++: `void line(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int lineType=8, int shift=0)`

Python: `cv2.line(img, pt1, pt2, color[, thickness[, lineType[, shift]]]) → None`

C: `void cvLine(CvArr* img, CvPoint pt1, CvPoint pt2, CvScalar color, int thickness=1, int line_type=8, int shift=0 )`

Python: `cv.Line(img, pt1, pt2, color, thickness=1, lineType=8, shift=0) → None`

## **cvPolyLine Draws simple or thick polygons.**

```
void cvPolyLine(CvArr*img,CvPoint**pts,int*npts,int contours,int is
 closed,CvScalar color,int thickness=1, int lineType=8,int shift=0);
```

- pts Array of pointers to polygons
- npts Array of polygon vertex counters
- contours Number of contours that bind the filled region
- img Image
- is closed Indicates whether the polylines must be drawn closed. If closed, the function draws the line from the last vertex of every contour to the first vertex.
- color Polyline color
- thickness Thickness of the polyline edges
- lineType Type of the line segments
- shift Number of fractional bits in the vertex coordinates

# DRAWING FUNCTION (CORE)

- **cvCircle** Draws a circle draws a simple or filled circle with a given center and radius.
- `cvCircle( CvArr *img, CvPoint center, int radius, CvScalar color, int thickness=1, int lineType=8, int shift=0 );`
  - `img` Image where the circle is drawn
  - `center` Center of the circle
  - `radius` Radius of the circle
  - `color` Circle color
  - `thickness` Thickness of the circle outline if positive, otherwise this indicates that a filled circle is to be drawn
  - `lineType` Type of the circle boundary, see Line description
  - `Shift` Number of fractional bits in the center coordinates and radius value

# circle

C++: `void circle(Mat& img, Point center, int radius, const Scalar& color, int thickness=1, int lineType=8, int shift=0)`

Python: `cv2.circle(img, center, radius, color[, thickness[, lineType[, shift]]]) → None`

C: `void cvCircle(CvArr* img, CvPoint center, int radius, CvScalar color, int thickness=1, int line_type=8, int shift=0 )`

Python: `cv.Circle(img, center, radius, color, thickness=1, lineType=8, shift=0) → None`



# DRAWING FUNCTION (CORE)

**cvRectangle** Draws a simple, thick, or filled rectangle. with two opposite corners pt1 and pt2

```
cvRectangle(CvArr *img,CvPoint pt1,CvPoint pt2,CvScalar color,
int thickness=1, int lineType=8, int shift=0);
```

- img Image
- pt1 One of the rectangle's vertices
- pt2 Opposite rectangle vertex
- color Line color (RGB) or brightness (grayscale image)
- thickness Thickness of lines that make up the rectangle. Negative values, e.g., CV\_FILLED, cause the function to draw a filled rectangle.
- lineType Type of the line, see Line description
- shift Number of fractional bits in the point coordinates

# rectangle

C++: `void rectangle(Mat& img, Point pt1, Point pt2, const Scalar& color, int thickness=1, int lineType=8, int shift=0)`

C++: `void rectangle(Mat& img, Rect rec, const Scalar& color, int thickness=1, int lineType=8, int shift=0 )`

Python: `cv2.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]]) → None`

C: `void cvRectangle(CvArr* img, CvPoint pt1, CvPoint pt2, CvScalar color, int thickness=1, int line_type=8, int shift=0 )`

Python: `cv.Rectangle(img, pt1, pt2, color, thickness=1, lineType=8, shift=0) → None`

# VẼ HÌNH TRÒN C (CXCORE)

The function circle draws a simple or filled circle with a given center and radius

```
void circle(Mat& img, Point center, int radius,
 const Scalar& color, int thickness=1,
 int lineType=8, int shift=0);
```

img Image where the circle is drawn

center Center of the circle

radius Radius of the circle

color Circle color

thickness Thickness of the circle outline if positive; negative thickness means that a filled circle is to be drawn

lineType Type of the circle boundary, see cv::line description

shift Number of fractional bits in the center coordinates and radius value

# VẼ HÌNH CHỮ NHẬT C (CXCORE)

Draws a simple, thick, or filled up-right rectangle.

```
void rectangle(Mat& img, Point pt1, Point pt2,
```

- const Scalar& color, int thickness=1,
- int lineType=8, int shift=0);
- img Image
- pt1 One of the rectangle's vertices
- pt2 Opposite to pt1 rectangle vertex
- color Rectangle color or brightness (grayscale image)
- thickness Thickness of lines that make up the rectangle. Negative values, e.g. CV\_FILLED,

mean that the function has to draw a filled rectangle.

- lineType Type of the line, see cv::line description
- shift Number of fractional bits in the point coordinates

The function rectangle draws a rectangle outline or a filled rectangle, which two opposite corners are pt1 and pt2.

# IN CHỮ

- **cvPutText** Draws a text string.

`cvPutText( CvArr *img, const char* text, CvPoint org, const CvFont *font, CvScalar color );`

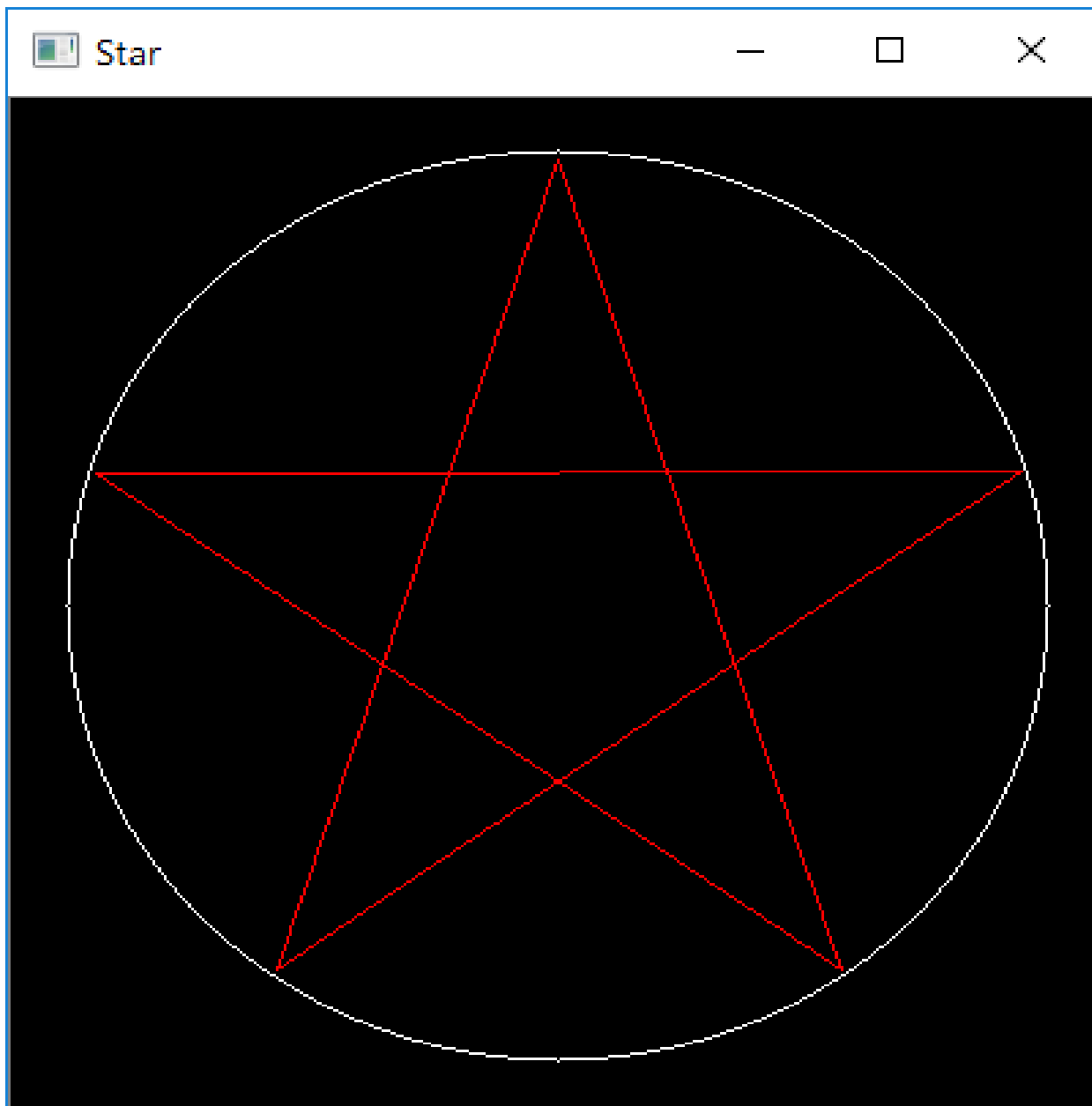
- `img` Input image
- `text` String to print
- `org` Coordinates of the bottom-left corner of the first letter
- `font` Pointer to the font structure
- `color` Text color
- The function renders the text in the image with the specified font and color. The printed text is clipped by the ROI rectangle. Symbols that do not belong to the specified font are replaced with the symbol for a rectangle.

# Draw red star

```
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main()
{
 //tạo ảnh đen với kích thước 400 x 400 pixels
 Mat img = Mat(400, 400, CV_8UC3, Scalar(0, 0, 0));
 //vẽ hình tròn
 int r = 180;
 circle(img, Point(200, 200), r, Scalar(255, 255, 255));
 //tạo danh sách các điểm
 vector<Point> sao;
 sao.push_back(Point(200, 24));
}
```

# Draw red star

```
sao.push_back(Point(97, 344));
sao.push_back(Point(370, 147));
sao.push_back(Point(31, 148));
sao.push_back(Point(304, 344));
const Point *pts = (const Point*) Mat(sao).data;
int npts = sao.size();
//vẽ ngôi sao
bool closed = true;
polylines(img, &pts, &npts, 1, closed, Scalar(0, 0, 255), 1);
imshow("Star", img);
waitKey(0); // Wait for a keystroke in the window
getchar();
}
```





# cvtColor

- Chuyển đổi không gian màu, thứ tự màu là BGR từ 0 đến 255, với HSV H từ 0 đến 180, S và V từ 0 đến 255, YCrCb từ 0 đến 255

**C++:** `void cvtColor(InputArray src, OutputArray dst, int code, int dstCn=0 )`

**Python:** `cv2.cvtColor(src, code[, dst[, dstCn]])` → dst

**C:** `void cvCvtColor(const CvArr* src, CvArr* dst, int code)`

**Python:** `cv.CvtColor(src, dst, code)` → None

**Parameters:**

- **src** – input image: 8-bit unsigned, 16-bit unsigned ( `cv_16UC...` ), or single-precision floating-point.
- **dst** – output image of the same size and depth as `src`.
- **code** – color space conversion code (see the description below).
- **dstCn** – number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from `src` and `code` .

# cvtColor

**Parameters:**

- **src** – input image: 8-bit unsigned, 16-bit unsigned ( `cv_16UC...` ), or single-precision floating-point.
- **dst** – output image of the same size and depth as `src`.
- **code** – color space conversion code (see the description below).
- **dstCn** – number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from `src` and `code` .

The function converts an input image from one color space to another. In case of a transformation to-from RGB color space, the order of the channels should be specified explicitly (RGB or BGR). Note that the default color format in OpenCV is often referred to as RGB but it is actually BGR (the bytes are reversed). So the first byte in a standard (24-bit) color image will be an 8-bit Blue component, the second byte will be Green, and the third byte will be Red. The fourth, fifth, and sixth bytes would then be the second pixel (Blue, then Green, then Red), and so on.

# cvtColor

- RGB <-> GRAY : CV\_BGR2GRAY, CV\_RGB2GRAY, CV\_GRAY2BGR, CV\_GRAY2RGB
- RGB <-> YCrCb : CV\_BGR2YCrCb, CV\_RGB2YCrCb, CV\_YCrCb2BGR, CV\_YCrCb2RGB
- RGB <-> HSV: CV\_BGR2HSV, CV\_RGB2HSV, CV\_HSV2BGR, CV\_HSV2RGB
- RGB <-> HLS: CV\_BGR2HLS, CV\_RGB2HLS, CV\_HLS2BGR, CV\_HLS2RGB

# Split Tách các kênh màu

**C++:** `void split(const Mat& src, Mat* mvbegin)`

**C++:** `void split(InputArray m, OutputArrayOfArrays mv)`

**Python:** `cv2.split(m[, mv]) → mv`

**C:** `void cvSplit(const CvArr* src, CvArr* dst0, CvArr* dst1, CvArr* dst2, CvArr* dst3)`

**Python:** `cv.Split(src, dst0, dst1, dst2, dst3) → None`

- Parameters:**
- **src** – input multi-channel array.
  - **mv** – output array or vector of arrays; in the first variant of the function the number of arrays must match `src.channels()`; the arrays themselves are reallocated, if needed.

The functions `split` split a multi-channel array into separate single-channel arrays:

$$mv[c](I) = src(I)_c$$

```
Mat image; Mat rgbchannel[3];
image = imread("C:/lenna.jpg", CV_LOAD_IMAGE_COLOR);
namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
imshow("Original Image", image);
namedWindow("Blue",CV_WINDOW_AUTOSIZE);
imshow("Red", rgbchannel[0]);
namedWindow("Green",CV_WINDOW_AUTOSIZE);
imshow("Green", rgbchannel[1]);
namedWindow("Red",CV_WINDOW_AUTOSIZE);
imshow("Blue", rgbchannel[2]);
waitKey(0); return 0;
```

# Merge

**C++:** `void merge(const Mat* mv, size_t count, OutputArray dst)`

**C++:** `void merge(InputArrayOfArrays mv, OutputArray dst)`

**Python:** `cv2.merge(mv[, dst]) → dst`

**C:** `void cvMerge(const CvArr* src0, const CvArr* src1, const CvArr* src2, const CvArr* src3, CvArr* dst)`

**Python:** `cv.Merge(src0, src1, src2, src3, dst) → None`

- Parameters:**
- **mv** – input array or vector of matrices to be merged; all the matrices in `mv` must have the same size and the same depth.
  - **count** – number of input matrices when `mv` is a plain C array; it must be greater than zero.
  - **dst** – output array of the same size and the same depth as `mv[0]`; The number of channels will be the total number of channels in the matrix array.

The functions `merge` merge several arrays to make a single multi-channel array. That is, each element of the output array will be a concatenation of the elements of the input arrays, where elements of *i*-th input array are treated as `mv[i].channels()`-element vectors.

# Merge

```
Mat src=imread("image.jpg",1);
 namedWindow("src",1);imshow("src",src);
vector<Mat> rgbChannels(3);
 split(src, rgbChannels);
 Mat g, fin_img;
 g = Mat::zeros(Size(src.cols, src.rows), CV_8UC1);
// Showing Red Channel, G and B channels are kept as zero
matrix for visual perception
{ vector<Mat> channels;
 channels.push_back(g);
 channels.push_back(g);
 channels.push_back(rgbChannels[2]);
merge(channels, fin_img); imshow("R", fin_img); }
```

# Merge

```
// Showing Green Channel
{ vector<Mat> channels;
 channels.push_back(g);
 channels.push_back(rgbChannels[1]);
 channels.push_back(g);
 merge(channels, fin_img);
 namedWindow("G",1);imshow("G", fin_img); }

// Showing Blue Channel
{ vector<Mat> channels;
 channels.push_back(rgbChannels[0]);
 channels.push_back(g);
 channels.push_back(g);
 merge(channels, fin_img);
 namedWindow("B",1);imshow("B", fin_img); } waitKey(0); return 0;
```



# bitwise\_and

Calculates the per-element bit-wise conjunction of two arrays or an array and a scalar.

**C++:** `void bitwise_and(InputArray src1, InputArray src2, OutputArray dst, InputArray mask=noArray())`

**Python:** `cv2.bitwise_and(src1, src2[, dst[, mask]]) → dst`

**C:** `void cvAnd(const CvArr* src1, const CvArr* src2, CvArr* dst, const CvArr* mask=NULL)`

**C:** `void cvAndS(const CvArr* src, CvScalar value, CvArr* dst, const CvArr* mask=NULL)`

**Python:** `cv.And(src1, src2, dst, mask=None) → None`

**Python:** `cv.AndS(src, value, dst, mask=None) → None`

- Parameters:**
- **src1** – first input array or a scalar.
  - **src2** – second input array or a scalar.
  - **src** – single input array.
  - **value** – scalar value.
  - **dst** – output array that has the same size and type as the input arrays.
  - **mask** – optional operation mask, 8-bit single channel array, that specifies elements of the output array to be changed.

# bitwise\_and

The function calculates the per-element bit-wise logical conjunction for:

- Two arrays when `src1` and `src2` have the same size:

$$\text{dst}(I) = \text{src1}(I) \wedge \text{src2}(I) \quad \text{if } \text{mask}(I) \neq 0$$

- An array and a scalar when `src2` is constructed from `scalar` or has the same number of elements as `src1.channels()`:

$$\text{dst}(I) = \text{src1}(I) \wedge \text{src2} \quad \text{if } \text{mask}(I) \neq 0$$

- A scalar and an array when `src1` is constructed from `scalar` or has the same number of elements as `src2.channels()`:

$$\text{dst}(I) = \text{src1} \wedge \text{src2}(I) \quad \text{if } \text{mask}(I) \neq 0$$

# Tách vùng dựa vào ngưỡng

- Cần phải có ngưỡng
- Dùng hàm `cvinrangeS`, `inRange` (ảnh màu) hay `cvThreshold`, `threshold` (ảnh xám, một kênh)

`cvInRange` and `cvInRangeS`

```
void cvInRange(
 const CvArr* src,
 const CvArr* lower,
 const CvArr* upper,
 CvArr* dst
);
void cvInRangeS(
 const CvArr* src,
 CvScalar lower,
 CvScalar upper,
 CvArr* dst
);
```

# Tách vùng- Đổ ra ảnh nhị phân

**cvThreshold( const Mat& src, Mat& dst, double threshVal, double max, int thresholdType )**

- applies a fix level threshold to the each element of 'src' array write a value to corresponding array element of 'dst'
- Arguements -
  - const Mat& src - Source array (This should be single channel)
  - Mat& dst - Destination array which has the same size and same type as the 'src'
  - double threshVal - Threshold value
  - double max - Maximum value to use with 'THRESH\_BINARY' and 'THRESH\_BINARY\_INV' which are thresholding types
  - int thresholdType - You can use one of the following for this arguement

- THRESH\_BINARY

|                               |                                         |
|-------------------------------|-----------------------------------------|
| $\text{dst}(x,y)=\text{max},$ | if $\text{src}(x,y) > \text{ThreshVal}$ |
| $\text{dst}(x,y)=0,$          | if $\text{src}(x,y) < \text{ThreshVal}$ |

# Tách vùng dựa vào ngưỡng

- THRESH\_BINARY\_INV
$$\begin{aligned} \text{dst}(x,y) &= 0, & \text{if } \text{src}(x,y) > \text{ThreshVal} \\ \text{dst}(x,y) &= \text{max}, & \text{if } \text{src}(x,y) < \text{ThreshVal} \end{aligned}$$
- THRESH\_TOZERO
$$\begin{aligned} \text{dst}(x,y) &= \text{src}(x,y), & \text{if } \text{src}(x,y) > \text{ThreshVal} \\ \text{dst}(x,y) &= 0, & \text{if } \text{src}(x,y) < \text{ThreshVal} \end{aligned}$$
- THRESH\_TOZERO\_INV
$$\begin{aligned} \text{dst}(x,y) &= 0, & \text{if } \text{src}(x,y) > \text{ThreshVal} \\ \text{dst}(x,y) &= \text{src}(x,y), & \text{if } \text{src}(x,y) < \text{ThreshVal} \end{aligned}$$
- THRESH\_TRUNC
$$\begin{aligned} \text{dst}(x,y) &= \text{threshVal}, & \text{if } \text{src}(x,y) > \text{ThreshVal} \\ \text{dst}(x,y) &= \text{src}(x,y), & \text{if } \text{src}(x,y) < \text{ThreshVal} \end{aligned}$$

# Contour

- **cvFindContours( CvArr\* img, CvMemStorage\* str, CvSeq\*\* first\_contour, int header\_size, int mode, int method, CvPoint offset )**
- Find all contours in a binary image  
Arguments -
  - CvArr\* img - Source image (This should be 8 bit single channel). All non-zero pixels are considered as 1 and all zero remain zero.
  - CvMemStorage\* str - Memory blocks to store all obtained contours
  - CvSeq\*\* first\_contour - store a pointer to the first contour in the memory block, 'str'
  - int header\_size - size of the sequence header
  - int mode - mode of retrieval of contours from the image

•

- You have to choose one of the following
  - CV\_RETR\_LIST - Retrieves all of the contours and put them in a list
  - CV\_RETR\_EXTERNAL - Retrieves only the extreme outer contours
  - CV\_RETR\_CCOMP - Retrieves all of the contours and organizes them into a two-level hierarchy: on the top level are the external boundaries of the components, on the second level are the boundaries of the holes
  - CV\_RETR\_TREE - Retrieves all of the contours and reconstructs the full hierarchy of nested contour

- int method - Approximation method

You have to choose one of the following

- CV\_CHAIN\_CODE - Outputs contours in the Freeman chain code
- CV\_CHAIN\_APPROX\_NONE - Translates all of the points from the chain code into points
- CV\_CHAIN\_APPROX\_SIMPLE - Compresses horizontal, vertical, and diagonal segments and leaves only their end points
- CV\_CHAIN\_APPROX\_TC89\_L1, CV\_CHAIN\_APPROX\_TC89\_KCOS - Applies one of the flavors of the Teh-Chin chain approximation algorithm.
- CV\_LINK\_RUNS - uses a completely different contour retrieval algorithm by linking horizontal segments of 1's. Only the 'CV\_RETR\_LIST' retrieval mode can be used with this method.



- CvPoint offset - Offset by which every contour point should be shifted. This is useful when we have set ROI (Region Of Interest) in the image. Normally we set the offset to 'cvPoint(0,0)'
  - cvCreateMemStorage(int byteSize)
- Creates memory storage which has the capacity specified by the parameter 'byteSize'. But if byteSize=0, the allocated capacity is the default value(usually 64 Kb)
- **cvGetSeqElem( const CvSeq\* seq, int index )**
  - Returns a pointer to the element of 'seq' at 'index'
  - **cvReleaseMemStorage( CvMemStorage\*\* storage )**
  - Deallocate memory blocks which have been allocated by 'cvCreateMemStorage()' function

- **cvApproxPoly( const void\* src, int header\_size, CvMemStorage\* storage, int method, double para1, int para2 )**
- Approximate polygonal curves with specified precision
- arguments -
  - const void\* src - Sequence of points
  - int header\_size - size of the sequence header
  - CvMemStorage\* storage - memory block that contains all contours
  - int method - Approximation method. (The only method, available to use for this argument is 'CV\_POLY\_APPROX\_DP')
  - double para1 - approximation accuracy
  - int para2 - Determines whether the single sequence should be approximated or all sequences in the same level or below

# PHÁT GIÁC MÀU ĐỎ C

```
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
//This function threshold the HSV image and create a binary image
IplImage* GetThresholdedImage(IplImage* imgHSV){
 IplImage* imgThresh=
cvCreateImage(cvGetSize(imgHSV),IPL_DEPTH_8U, 1);
 cvInRangeS(imgHSV, cvScalar(170,160,60),
cvScalar(180,256,256), imgThresh);
 return imgThresh;
}
int main(){
 CvCapture* capture =0;
 capture = cvCaptureFromCAM(0);
 if(!capture){
printf("Capture failure\n");
return -1; }
```

# PHÁT GIÁC MÀU ĐỎ C

```
IpImage* frame=0;
cvNamedWindow("Video");
cvNamedWindow("Ball");
//iterate through each frames of the video
while(true){
 frame = cvQueryFrame(capture);
 if(!frame) break;
 frame=cvCloneImage(frame);
 cvSmooth(frame, frame, CV_GAUSSIAN,3,3); //smooth the
original image using Gaussian kernel
 IpImage* imgHSV = cvCreateImage(cvGetSize(frame),
IPL_DEPTH_8U, 3);
 cvCvtColor(frame, imgHSV, CV_BGR2HSV); //Change the
color format from BGR to HSV
 IpImage* imgThresh = GetThresholdedImage(imgHSV);
```

# PHÁT GIÁC MÀU ĐỎ C

```
cvSmooth(imgThresh, imgThresh, CV_GAUSSIAN,3,3); //smooth the
binary image using Gaussian kernel
 cvShowImage("Ball", imgThresh);
 cvShowImage("Video", frame);
 //Clean up used images
 cvReleaseImage(&imgHSV);
 cvReleaseImage(&imgThresh);
 cvReleaseImage(&frame);
 //Wait 50mS
 int c = cvWaitKey(10);
 //If 'ESC' is pressed, break the loop
 if((char)c==27) break;
}
cvDestroyAllWindows() ;
cvReleaseCapture(&capture);
return 0;
}
```

# Phát giác màu đỏ

## loại bỏ các màu khác C++

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <vector>
using namespace cv;
using namespace std;
void main() {
 Mat bgr_image = imread("d:/circles_rectangles.jpg");
 namedWindow("OriginPicture", 0);
 imshow("OriginPicture", bgr_image);
 Mat orig_image = bgr_image.clone();
```

# **Phát giác màu đỏ**

## **loại bỏ các màu khác C++**

```
medianBlur(bgr_image, bgr_image, 3);
// Convert input image to HSV
Mat hsv_image; Mat h_lower_red_hue_range;
Mat h_upper_red_hue_range;
Mat h_red_hue_image; Mat fin_img;
cvtColor(bgr_image, hsv_image, COLOR_BGR2HSV);
// Threshold the HSV image, keep only the red pixels
inRange(hsv_image, Scalar(0, 100, 100), Scalar(10, 255, 255),
h_lower_red_hue_range);
inRange(hsv_image, Scalar(160, 100, 100), Scalar(179, 255,
255), h_upper_red_hue_range);
```

# Phát giác màu đỏ

## loại bỏ các màu khác C++

```
// Combine the above two images
addWeighted(h_lower_red_hue_range, 1.0,
h_upper_red_hue_range, 1.0, 0.0, h_red_hue_image);
imshow("H", h_red_hue_image);
//Remove other color
bitwise_and(hsv_image, hsv_image, fin_img, h_red_hue_image);
imshow("HSV Thresholded", fin_img);
cvtColor(fin_img, fin_img, COLOR_HSV2BGR);
imshow("Red Color Only", fin_img);
waitKey(0);
}
```



# Phát giác vòng tròn đỏ C++

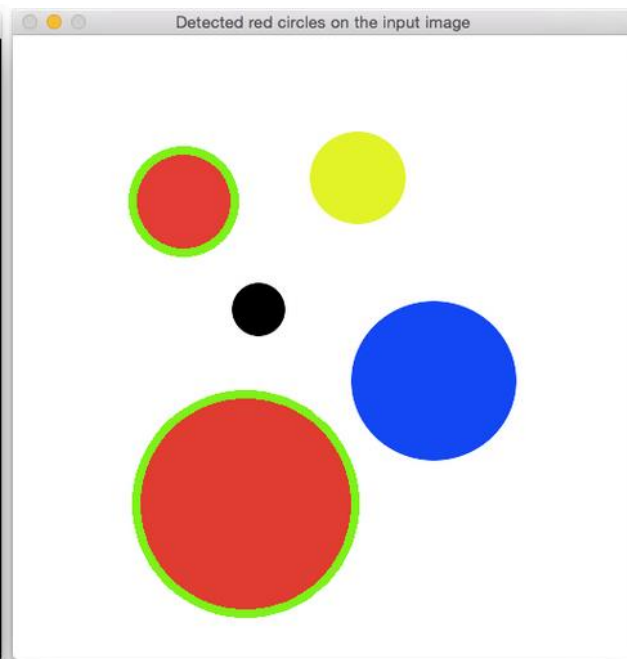
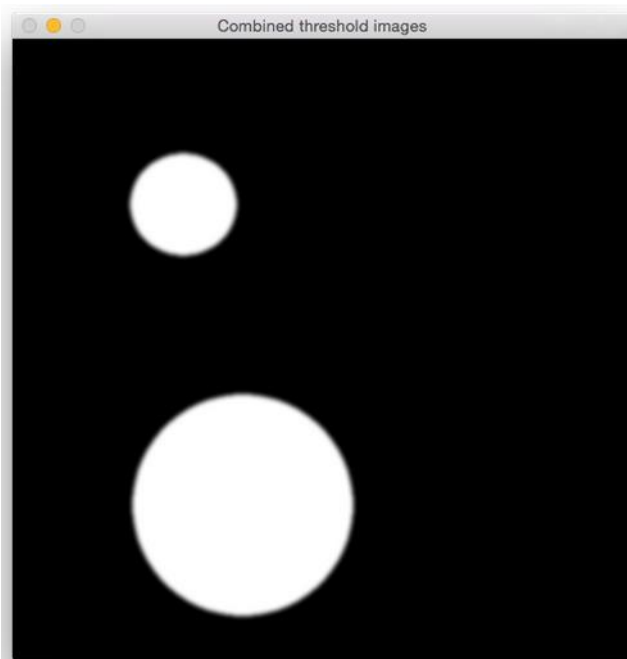
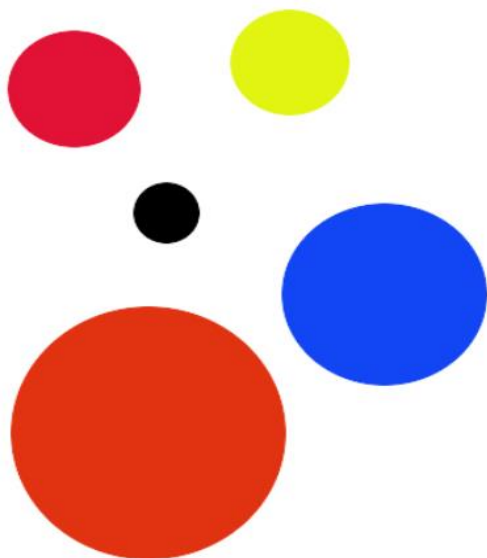
```
// OpenCV H has values from 0 to 180, S and V from 0 to 255.
//The red color has the hue values approximately in the range of
0 to 10 and 160 to 180. Convert input image to HSV
Mat hsv_image; Mat lower_red_hue_range;
Mat upper_red_hue_range; Mat red_hue_image;
cvtColor(bgr_image, hsv_image, COLOR_BGR2HSV);
inRange(hsv_image, Scalar(0, 100, 100), Scalar(10, 255, 255),
lower_red_hue_range);
inRange(hsv_image, Scalar(160, 100, 100), Scalar(179, 255,
255), upper_red_hue_range);
// Combine the above two images
addWeighted(lower_red_hue_range, 1.0, upper_red_hue_range,
1.0, 0.0, red_hue_image);
GaussianBlur(red_hue_image, red_hue_image, cv::Size(9, 9), 2,
2);
```

# Phát giác vòng tròn đỏ C++

```
// Use the Hough transform to detect circles in the combined
threshold image
vector<Vec3f> circles;
HoughCircles(red_hue_image, circles,
CV_HOUGH_GRADIENT, 1, red_hue_image.rows/8, 100, 20, 0,
0);
// Loop over all detected circles and outline them on the original
image
if(circles.size() == 0) std::exit(-1);
for(size_t current_circle = 0; current_circle < circles.size();
++current_circle) {
Point center(round(circles[current_circle][0]),
round(circles[current_circle][1]));
int radius = round(circles[current_circle][2]);
```

# Phát giác vòng tròn đỏ C++

```
circle(orig_image, center, radius, Scalar(0, 255, 0), 5); }
// Show images
namedWindow("Threshold lower image", WINDOW_AUTOSIZE);
imshow("Threshold lower image", lower_red_hue_range);
namedWindow("Threshold upper image", WINDOW_AUTOSIZE);
imshow("Threshold upper image", upper_red_hue_range);
namedWindow("Combined threshold images", WINDOW_AUTOSIZE);
imshow("Combined threshold images", red_hue_image);
namedWindow("Detected red circles on the input image",
WINDOW_AUTOSIZE);
imshow("Detected red circles on the input image",
orig_image);waitKey(0); return 0;
```



## How to Find Exact Range for 'Hue', 'Saturation' and 'Value' for a Given Object

```
#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
int lowerH=0;
int lowerS=0;
int lowerV=0;
int upperH=180;
int upperS=256;
int upperV=256;
//This function threshold the HSV image and create a binary image
IplImage* GetThresholdedImage(IplImage* imgHSV){
IplImage*imgThresh=cvCreateImage(cvGetSize(imgHSV),IPL_DEPTH_8U, 1);
cvInRangeS(imgHSV, cvScalar(lowerH,lowerS,lowerV),
cvScalar(upperH,upperS,upperV), imgThresh);
return imgThresh;
}
```

## How to Find Exact Range for 'Hue', 'Saturation' and 'Value' for a Given Object

```
//This function create two windows and 6 trackbars for the "Ball"
window
void setwindowSettings(){
cvNamedWindow("Video");
cvNamedWindow("Ball");
cvCreateTrackbar("LowerH", "Ball", &lowerH, 180, NULL);
 cvCreateTrackbar("UpperH", "Ball", &upperH, 180, NULL);
cvCreateTrackbar("LowerS", "Ball", &lowerS, 256, NULL);
cvCreateTrackbar("UpperS", "Ball", &upperS, 256, NULL);
cvCreateTrackbar("LowerV", "Ball", &lowerV, 256, NULL);
cvCreateTrackbar("UpperV", "Ball", &upperV, 256, NULL);
}
```

# How to Find Exact Range for 'Hue', 'Saturation' and 'Value' for a Given Object

```
int main(){
 CvCapture* capture =0;
 capture = cvCaptureFromCAM(0);
 if(!capture){
 printf("Capture failure\n");
 return -1;
 }
 IplImage* frame=0;
 setWindowSettings();
 //iterate through each frames of the video
 while(true){
 frame = cvQueryFrame(capture);
 if(!frame) break;
 frame=cvCloneImage(frame);
 IplImage* imgHSV = cvCreateImage(cvGetSize(frame),
 IPL_DEPTH_8U, 3);
```

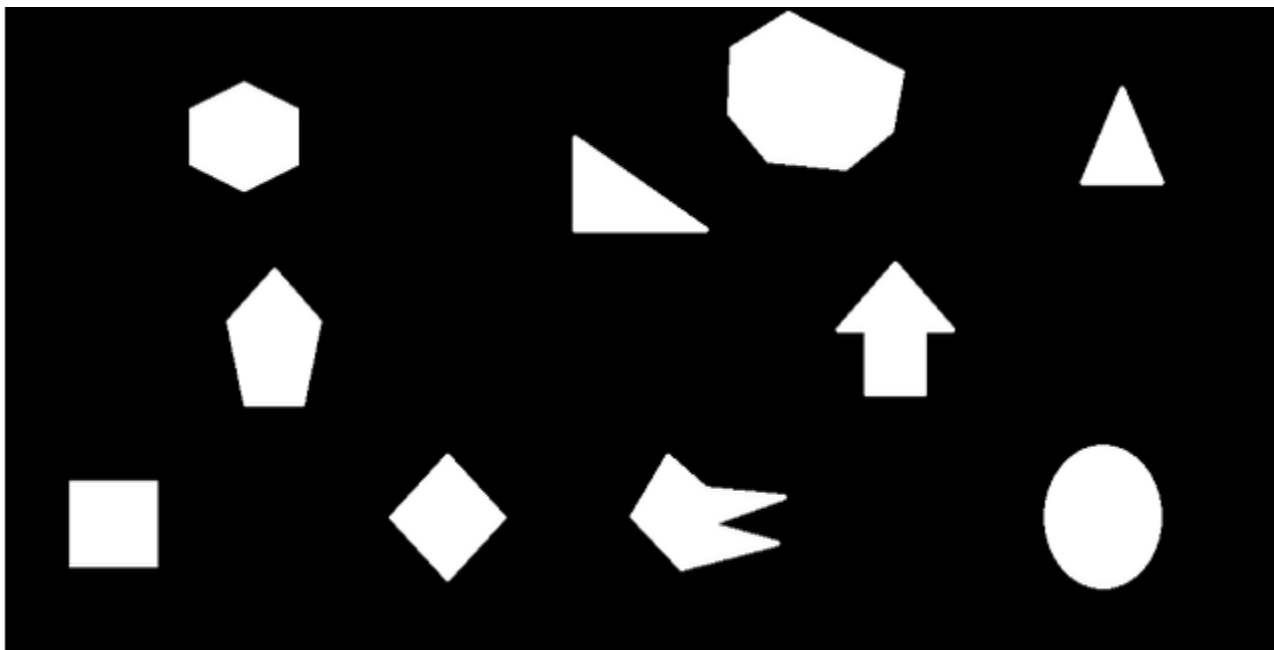
## How to Find Exact Range for 'Hue', 'Saturation' and 'Value' for a Given Object

```
cvCvtColor(frame, imgHSV, CV_BGR2HSV); //Change the color format
from BGR to HSV
IplImage* imgThresh = GetThresholdedImage(imgHSV);
cvShowImage("Ball", imgThresh);
cvShowImage("Video", frame);
//Clean up used images
cvReleaseImage(&imgHSV);
cvReleaseImage(&imgThresh);
cvReleaseImage(&frame);
//Wait 80mS
int c = cvWaitKey(80);
//If 'ESC' is pressed, break the loop
if((char)c==27) break;
}
cvDestroyAllWindows();
cvReleaseCapture(&capture);
return 0;
}
```



# Identification of Shapes of Object using Contours

- Using contours with OpenCV, you can get a sequence of points of vertices of each white patch (White patches are considered as polygons). As example, you will get 3 points (vertices) for a triangle, and 4 points for quadrilaterals. So, you can identify any polygon by the number of vertices of that polygon. You can even identify features of polygons such as convexity, concavity, equilateral and etc by calculating and comparing distances between vertices.
- All you need, is a binary image in which your objects should be white and the background should be black.
- Now I am going to identify triangles and quadrilaterals and heptagon in the above image using OpenCV. I'll draw a line along the perimeter of every identified polygon with colors blue for triangle, green for quadrilaterals and red for heptagons.



```

#include "stdafx.h"
#include <cv.h>
#include <highgui.h>
using namespace std;
int main() {
 IplImage* img = cvLoadImage
("C:/Users/SHERMAL/Desktop/FindingContours.png");
cvNamedWindow("Raw");
cvShowImage("Raw",img); //show the original image
 //converting the original image into grayscale
 IplImage* imgGrayScale = cvCreateImage(cvGetSize(img), 8, 1);
 cvCvtColor(img,imgGrayScale,CV_BGR2GRAY);
 //thresholding the grayscale image to get better results
 cvThreshold(imgGrayScale,imgGrayScale,128,255,CV_THRESH_B
INARY); //hoặc CV_THRESH_BINARY_INV

```

```

CvSeq* contours; //hold the pointer to a contour in the memory block
CvSeq* result; //hold sequence of points of a contour
 CvMemStorage *storage = cvCreateMemStorage(0); //storage area
 for all contours
 //finding all contours in the image
 cvFindContours(imgGrayScale, storage, &contours,
 sizeof(CvContour), CV_RETR_LIST,
 CV_CHAIN_APPROX_SIMPLE, cvPoint(0,0));
 //iterating through each contour
 while(contours)
 {
 //obtain a sequence of points of contour, pointed by the variable
 'contour'
 result = cvApproxPoly(contours, sizeof(CvContour), storage,
 CV_POLY_APPROX_DP, cvContourPerimeter(contours)*0.02, 0);

```

```
//if there are 3 vertices in the contour(It should be a triangle)
if(result->total==3)
{
 //iterating through each point
 CvPoint *pt[3];
 for(int i=0;i<3;i++){
 pt[i] = (CvPoint*)cvGetSeqElem(result, i);
 }
 //drawing lines around the triangle
 cvLine(img, *pt[0], *pt[1], cvScalar(255,0,0),4);
 cvLine(img, *pt[1], *pt[2], cvScalar(255,0,0),4);
 cvLine(img, *pt[2], *pt[0], cvScalar(255,0,0),4);
}
```

```
//if there are 4 vertices in the contour(It should be a quadrilateral)
else if(result->total==4)
{
//iterating through each point
CvPoint *pt[4];
for(int i=0;i<4;i++){
pt[i] = (CvPoint*)cvGetSeqElem(result, i);
}
//drawing lines around the quadrilateral
cvLine(img, *pt[0], *pt[1], cvScalar(0,255,0),4);
cvLine(img, *pt[1], *pt[2], cvScalar(0,255,0),4);
cvLine(img, *pt[2], *pt[3], cvScalar(0,255,0),4);
cvLine(img, *pt[3], *pt[0], cvScalar(0,255,0),4);
}
```

```

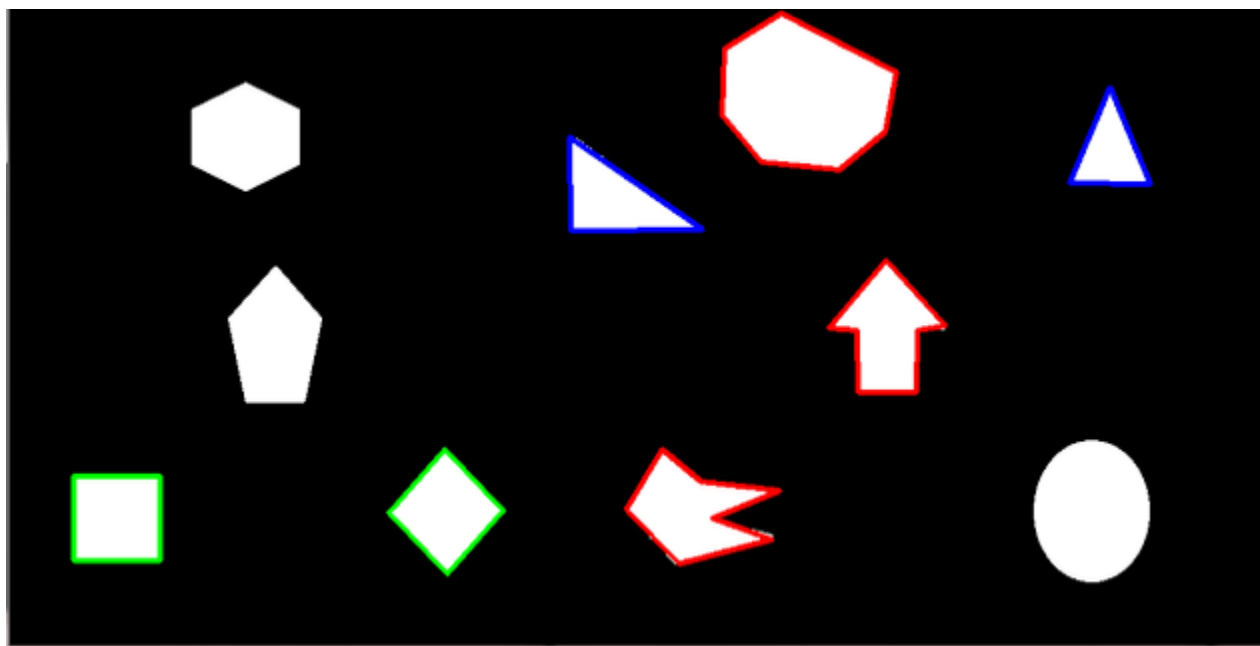
//if there are 7 vertices in the contour(It should be a heptagon)
else if(result->total ==7)
{
//iterating through each point
CvPoint *pt[7];
for(int i=0;i<7;i++){
pt[i] = (CvPoint*)cvGetSeqElem(result, i);
}
//drawing lines around the heptagon
cvLine(img, *pt[0], *pt[1], cvScalar(0,0,255),4);
cvLine(img, *pt[1], *pt[2], cvScalar(0,0,255),4);
cvLine(img, *pt[2], *pt[3], cvScalar(0,0,255),4);
cvLine(img, *pt[3], *pt[4], cvScalar(0,0,255),4);
cvLine(img, *pt[4], *pt[5], cvScalar(0,0,255),4);
cvLine(img, *pt[5], *pt[6], cvScalar(0,0,255),4);
cvLine(img, *pt[6], *pt[0], cvScalar(0,0,255),4);
}

```

```
//obtain the next contour
 contours = contours->h_next;
}

//show the image in which identified shapes are marked
cvNamedWindow("Tracked");
cvShowImage("Tracked",img);
cvWaitKey(0); //wait for a key press
//cleaning up
cvDestroyAllWindows();
cvReleaseMemStorage(&storage);
cvReleaseImage(&img);
cvReleaseImage(&imgGrayScale);
return 0;
}
```





# Skin Detect

- Khi cần nhận dạng khuôn mặt, cử chỉ bàn tay, bước thứ nhất là tìm những vùng da tay để tách riêng ra và bám theo (với video)
- Thông thường đổi ảnh RGB sang HSV hay YCbCr để loại trừ ảnh hưởng sáng tối.
- Chọn nền khác biệt màu da để tránh nhầm lẫn
- Tầm giá trị của H S phụ thuộc chủng tộc, cá nhân thường thường gần màu đỏ



# Skin Detect

- Với HSV thường chọn  $H = 0, 20$  ,  $S = 10..150$ ,  $V = 60..255$   
Với YCbCr  $Cr = 133..173$ ;  $Cb = 77..127$
- Nên dùng MsPaint để tìm giá trị cụ thể cho từng trường hợp