Computer Vision Winter-2017

Tutorial 0

Xichen Zhou

Introduction to OpenCV

OpenCV is a computer vision related package developed for over 16 years, originally developed in C, now also has C++/Java/Python bindings.

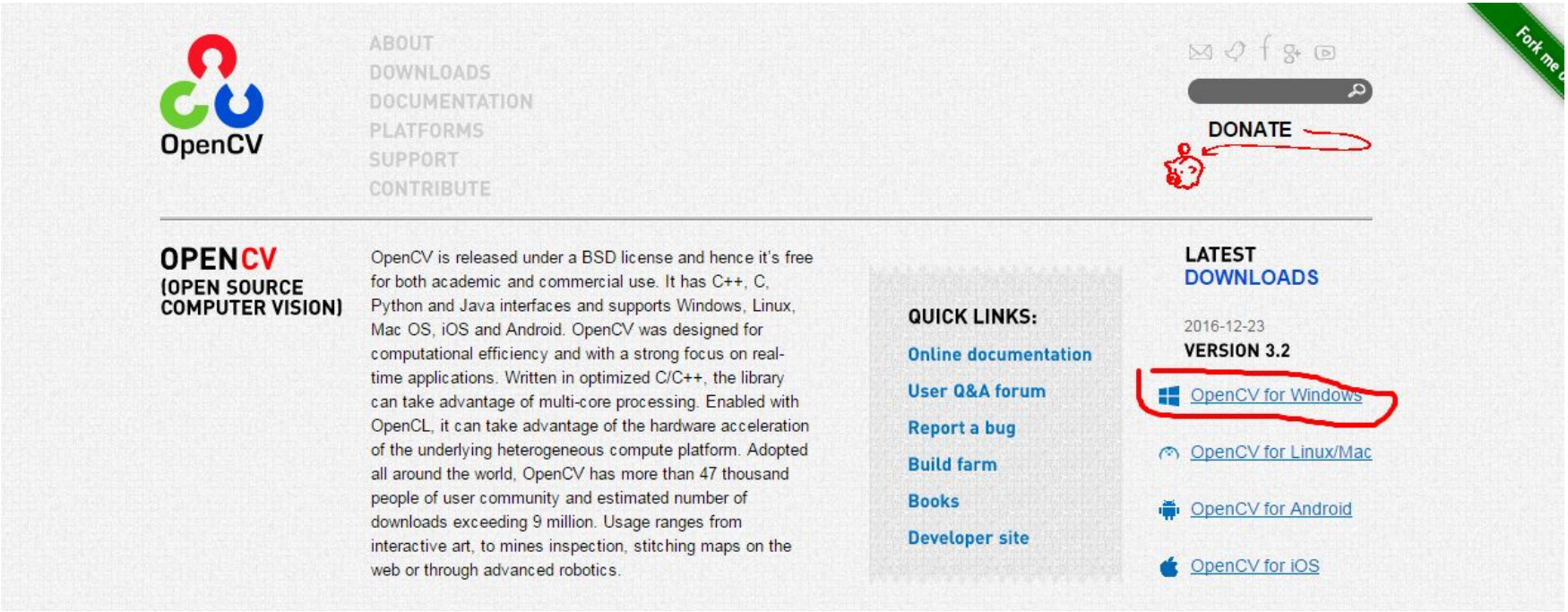
- It's powerful
- Easy to start
- Has a huge community

Like all the old software, it has inherited problem, it has no C++11 support.

Install OpenCV

- For windows guys, follow these **three** step
  - Get package from opencv.org
  - Extract the pre-compiled package
  - Setup ENV variables.
- Then you need to setup for IDE, various IDE has different settings(we will use VS2015 as a example)

Install On Windows

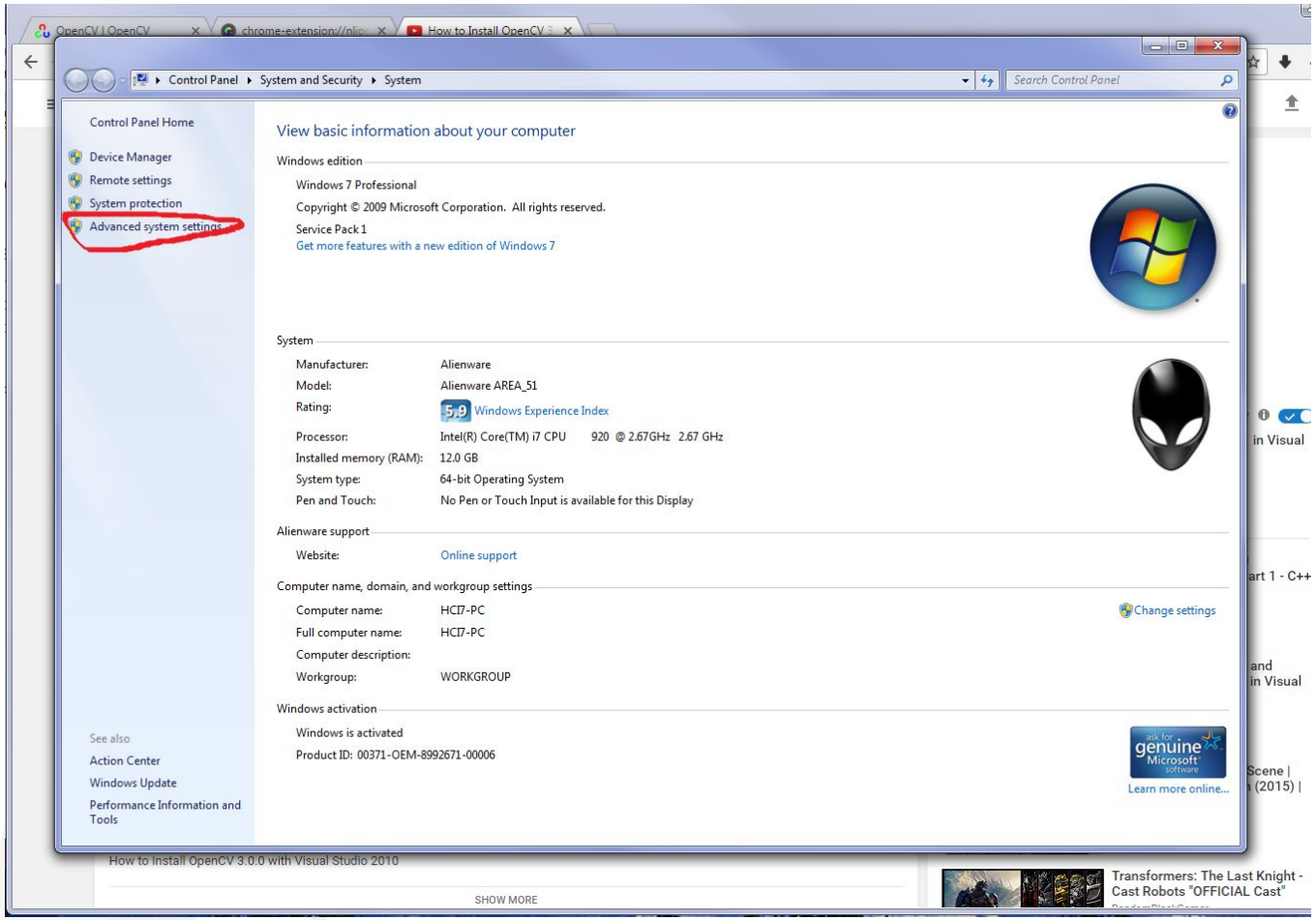


Download from opencv.org

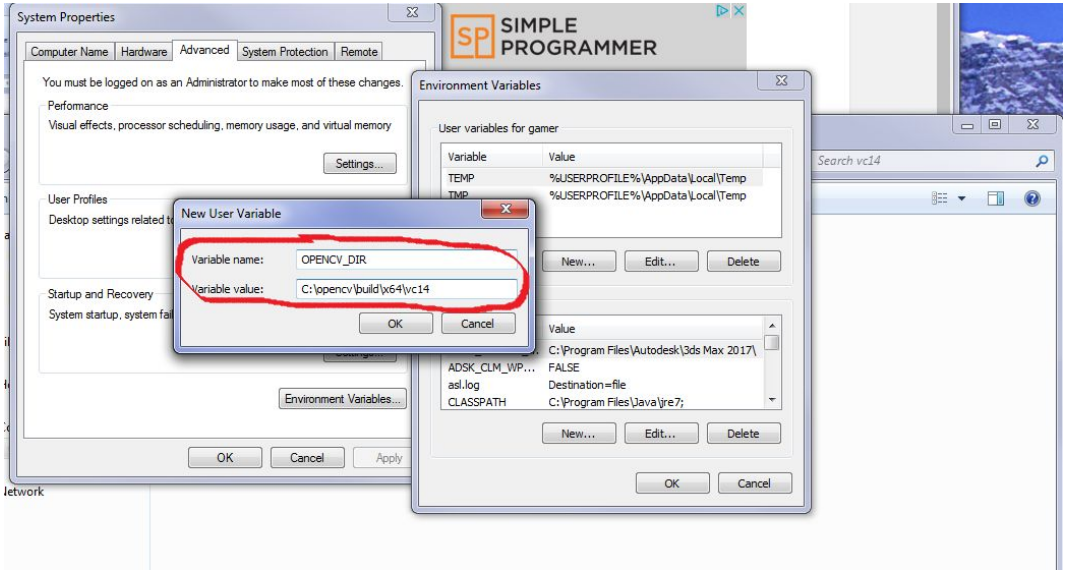
Install On Windows

Insert to where-you-want

Install On Windows



Install On Windows



Also **APPEND** %PATH with

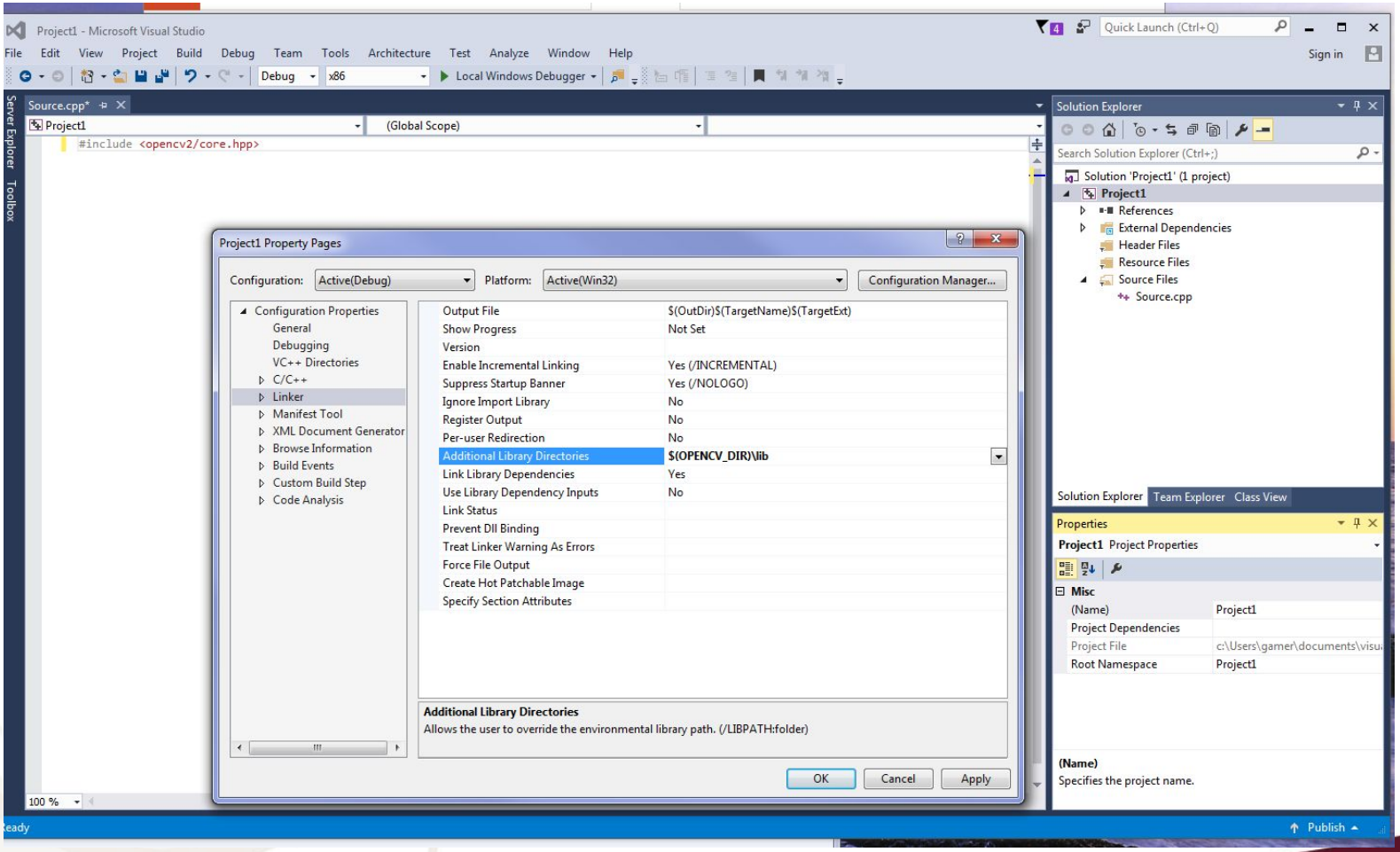
;%OPENCV_DIR\bin

- Setup OPENCV_DIR
- Here ours is C:\opencv\build\x64\build\vc14

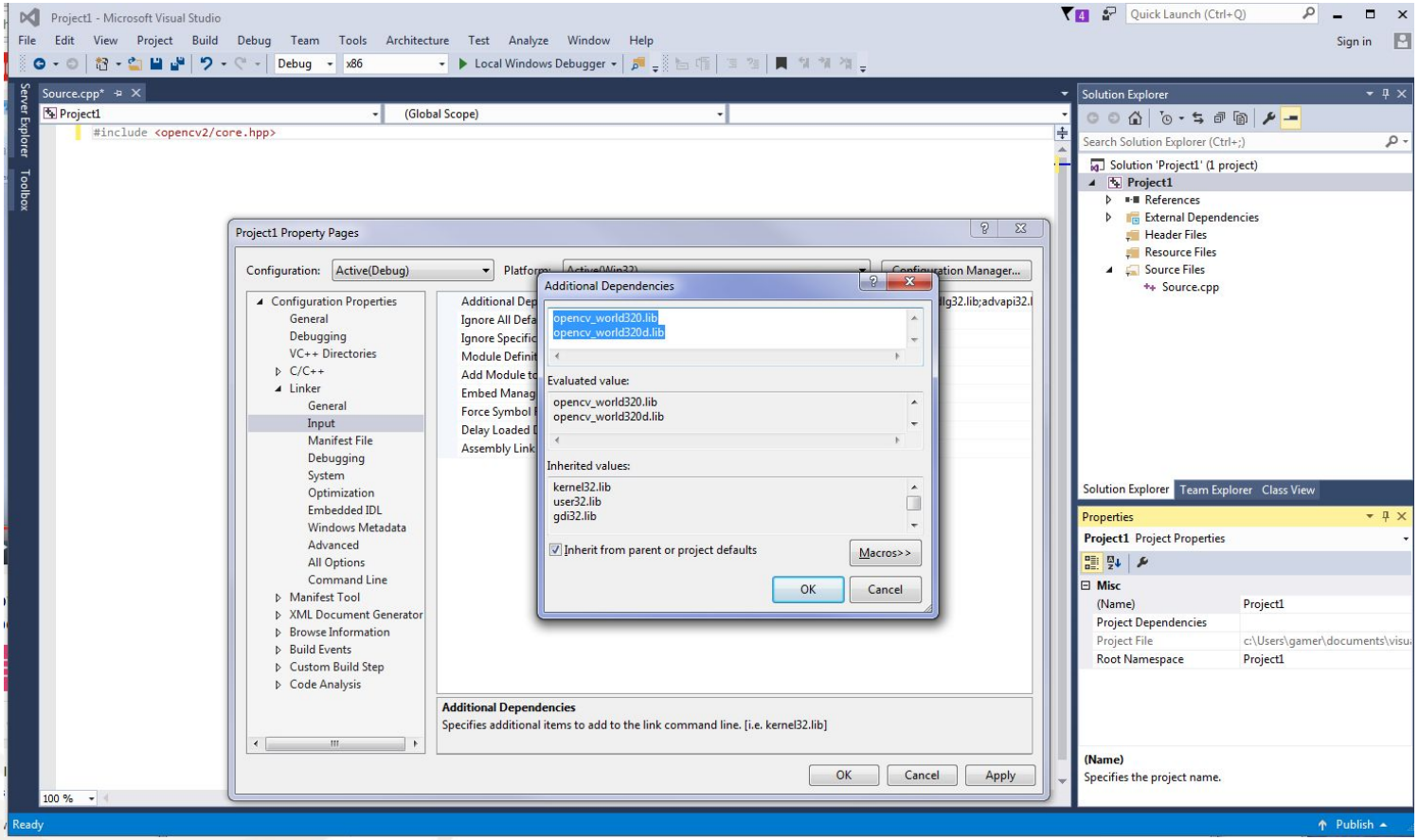
Setup Visual studio

- Creating a VC++ project
- Add additional include path with ${OPENCV_DIR}/../../include
- Add additional link path with ${OPENCV_DIR}/lib
- Add additional dependencies with
  - –opencv_ts3000.lib
  - –opencv_world300d.lib

Setup Visual studio



Setup Visual studio



On Unix(Linux/MacOS)

There are two types of installation

- By Package manager
- Using CMAKE to install

First OpenCV program

Reading and displaying images

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

int main(int argc, char **argv)
{
        cv::Mat img = cv::imread(argv[1]);
        cv::imshow("img", img);
        cv::waitKey(0);
}
```

OpenCV Mat data structure

```
//initialize a matrix
cv::Mat a = cv::Mat::zeros(100,100, CV_8UC1);
//initialize through other matrix
cv::Mat b = a.clone();
//retrieve part of the matrix using stride
cv::Mat c = a(cv::Rec(0,0,30,30));
//initialize a matrix with a scalar
cv::Mat d(100, 100, CV_32SC1, cv::scalar(0.0f));
//print a matrix
std::cout <<  “the matrix is: “ << std::endl << cv::format(c, cv::Formatter::FMT_PYTHON) << std::endl << std::endl;
//operations on matrix
cv::Mat f = a + b; //this works
//this works
cv::Mat g = a * 10;
//this doesn’t work
cv::Mat h = a % 10;
//element access
cv::Mat::at<T>(i,j); // equals cv::Mat::ptr(i,j)
//row access
cv::Mat.ptr(i);
```

Filtering image with OpenCV

```
…...
///Apply a mean filter. First create it 1/9 *  3x3 matrix containing ones.
int kernel_size = 3;
cv::Mat kernel = cv::Mat::ones( kernel_size, kernel_size, CV_32F )/ (float)(kernel_size*kernel_size);

cv::Point anchor = cv::Point( -1, -1 );
double delta = 0;
int ddepth = -1;

 ///Apply the filter
cv::Mat result_image;
cv::filter2D(image, result_image, ddepth , kernel, anchor, delta, BORDER_DEFAULT );
…...
```

Change Image contrast

```
int main( int, char** argv ) {
double alpha = 1.0; /*< Simple contrast control */
int beta = 0; /*< Simple brightness control */
Mat image = imread( argv[1] );
Mat new_image = Mat::zeros( image.size(), image.type() );
cout << " Basic Linear Transforms " << endl;
cout << "-------------------------" << endl;
cout << "* Enter the alpha value [1.0-3.0]: "; cin >> alpha;
cout << "* Enter the beta value [0-100]: "; cin >> beta;
for( int y = 0; y < image.rows; y++ ) {
        for( int x = 0; x < image.cols; x++ ) {
                for( int c = 0; c < 3; c++ ) {
                        new_image.at<Vec3b>(y,x)[c] =
                        saturate_cast<uchar>( alpha*( image.at<Vec3b>(y,x)[c] ) + beta );
                }
        }
}
namedWindow("Original Image", WINDOW_AUTOSIZE);
namedWindow("New Image", WINDOW_AUTOSIZE);
imshow("Original Image", image);
imshow("New Image", new_image);
waitKey();
return 0;
}
```

End