



FREE eBook

LEARNING opencv

Unaffiliated free eBook created from
Stack Overflow contributors.

#opencv

Table of Contents

About.....	1
Chapter 1: Getting started with opencv.....	2
Remarks.....	2
Versions.....	2
OpenCV 3.....	2
OpenCV 2.....	2
Examples.....	3
Build and install OpenCV from source.....	3
Prepare for the Build.....	3
Build and Install.....	3
Test Installation.....	4
Hello world example in Java.....	5
Get image from webcam.....	5
Java.....	5
C++.....	6
Python.....	6
Getting Started with OpenCV 3.1 on Windows.....	7
What & Why OPENCV?.....	12
Load and display an image with OpenCV.....	13
Chapter 2: Basic Structures.....	15
Introduction.....	15
Examples.....	15
DataType.....	15
Mat.....	15
Vec.....	16
Chapter 3: Blob Detection.....	17
Examples.....	17
Circular Blob Detection.....	17
Chapter 4: Build and Compile opencv 3.1.0-dev for Python2 on Windows using CMake and Visua.	

Remarks.....	19
Examples.....	28
Reading Image and Converting into grayscale.....	28
Chapter 5: Cascade Classifiers.....	30
Examples.....	30
Using Cascade Classifiers to detect face.....	30
Python.....	30
Code.....	30
Result.....	30
Face detection using haar cascade classifier.....	31
C++.....	31
Cascade Classifiers to detect face with Java.....	32
Java.....	32
Chapter 6: Contrast and Brightness in C++.....	34
Syntax.....	34
Parameters.....	34
Remarks.....	34
$g(i,j) = .f(i,j) +$	34
Examples.....	35
Adjusting brightness and contrast of an image in c++.....	35
Chapter 7: Creating a Video.....	37
Introduction.....	37
Examples.....	37
Creating a video with OpenCV (Java).....	37
Chapter 8: Display Image OpenCV.....	38
Examples.....	38
Display Image OpenCV Java.....	38
Reading MJPEG from IP camera.....	38
Basic reading and display of an image.....	39
Chapter 9: Drawing Functions in Java.....	40
Examples.....	40

Draw rectangle on image.....	40
Chapter 10: Drawing Shapes (Line, Circle, ..., etc) in C++.....	41
Introduction.....	41
Examples.....	41
Drawing Shapes Sample.....	41
Chapter 11: Edge detection.....	44
Syntax.....	44
Parameters.....	44
Examples.....	44
Canny algorithm.....	44
Canny Algorithm - C++.....	45
Calculating Canny Thresholds.....	46
Canny Edge Thresholds prototyping using Trackbars.....	46
Canny Edge Video from Webcam Capture - Python.....	47
Chapter 12: Image Content Modification.....	48
Examples.....	48
Set Whole Image to a Solid Color.....	48
Pixel by pixel modification of images.....	48
Image color modification in OpenCV - kmeans(). To scan all the pixels of an image and repl.....	48
Chapter 13: Image Processing.....	50
Syntax.....	50
Parameters.....	50
Remarks.....	50
Examples.....	50
Smoothing Images with Gaussian Blur in C++.....	50
Thresholding.....	51
Bilateral Filtering.....	52
Chapter 14: Loading and Saving Various Media Formats.....	54
Examples.....	54
Loading Images.....	54
Loading Videos.....	54
Live Capture.....	55

Saving Videos.....	55
Saving Images.....	56
Chapter 15: Object Detection.....	58
Examples.....	58
Template Matching with Java.....	58
Java Source Code.....	58
RESULT.....	58
Chapter 16: OpenCV initialization in Android.....	60
Examples.....	60
Async Initialization.....	60
OpenCV Manager.....	61
Static Initialization.....	61
Chapter 17: OpenCV Installation.....	63
Introduction.....	63
Examples.....	63
OpenCV Installation on Ubuntu.....	63
Chapter 18: Pixel Access.....	66
Remarks.....	66
Examples.....	66
Setting and getting pixel values of a Gray image in C++.....	66
Alternative pixel access with Matiterator.....	67
Efficient pixel access using cv::Mat::ptr pointer.....	70
Access individual pixel values with cv::Mat::at().....	71
Pixel Access in Mat.....	72
Chapter 19: Using Cascade Classifiers In Java.....	74
Syntax.....	74
Parameters.....	74
Examples.....	75
Getting a static image, detecting items on it, and outputting the results.....	75
Detecting images from a video device.....	76
Converting an Mat object to an BufferedImage object.....	77
Detections within Detections.....	77

Chapter 20: Using VideoCapture With OpenCV Python	81
Examples.....	81
Reading frames from a pre-captured video.....	81
Using VideoCapture With OpenCV Java.....	81
Credits	83

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [opencv](#)

It is an unofficial and free opencv ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official opencv.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with opencv

Remarks

This section provides an overview of what opencv is, and why a developer might want to use it.

It should also mention any large subjects within opencv, and link out to the related topics. Since the Documentation for opencv is new, you may need to create initial versions of those related topics.

Versions

OpenCV 3

Version	Release Date
3.2	2016-12-23
3.1	2015-12-18
3.0	2015-06-03
3.0 RC1	2015-04-23
3.0 beta	2014-11-07
3.0 alpha	2014-08-21

OpenCV 2

Version	Release Date
2.4.13	2016-05-19
2.4.12	2015-07-30
2.4.11	2015-02-25
2.4.10	2014-10-01
2.4.9	2014-04-14
2.3.1	2011-08-17

Version	Release Date
2.3.0	2011-07-04
2.2.0	2010-12-05
2.1.0	2010-04-06
2.0.0	2009-10-01
1.0.0	2006-10-19

Examples

Build and install OpenCV from source

This is a step-by-step guide to installing OpenCV 3 on a Debian-based Linux system from source. The steps should stay the same for other distros, just replace the relevant package manager commands when installing packages for the build.

Note: If you don't feel like wasting time building stuff or dislike the terminal, you can most likely install OpenCV from the Synaptic package manager GUI. However, these libraries are often out of date.

Prepare for the Build

Issue the following commands in your terminal to install the required packages:

```
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install cmake git libgtk2.0-dev pkg-config \
    libavcodec-dev libavformat-dev libswscale-dev
```

The following packages are optional:

```
sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev \
    libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev
```

Issue the following command to get the OpenCV source code and prepare the build:

```
mkdir ~/src
cd ~/src
git clone https://github.com/opencv/opencv.git
cd opencv
mkdir build && cd build
```

Build and Install

We include the examples in the build, but feel free to leave them out. Also feel free to set other flags and customise your build as you see fit.

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \  
-D CMAKE_INSTALL_PREFIX=/usr/local \  
-D INSTALL_PYTHON_EXAMPLES=ON \  
-D INSTALL_C_EXAMPLES=ON ..
```

If CMake didn't report any errors or missing libraries, continue with the build.

```
make -j$(nproc)
```

If no errors were produced, we can carry on with installing OpenCV to the system:

```
sudo make install
```

Now OpenCV should be available to your system. You can use the following lines to know where OpenCV was installed and which libraries were installed:

```
pkg-config --cflags opencv # get the include path (-I)  
pkg-config --libs opencv   # get the libraries path (-L) and the libraries (-l)
```

Test Installation

We first build the C++ examples:

```
cd ~/src/opencv/samples  
cmake .  
make
```

If no errors were produced, run a any sample, e.g.

```
./cpp/cpp-example-edge
```

If the sample runs, then the C++ libraries are properly installed.

Next, test the Python bindings:

```
python  
>> import cv2  
>> print cv2.__version__
```

If these commands import OpenCV and print the correct version without complaining, then the Python bindings are properly installed.

Congrats, you just built and installed OpenCV. Happy programming!

For Mac refer here [OpenCV installation on Mac OS X](#)

Hello world example in Java

OpenCv image read from file system in Java

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;

public class Giris {
    public static void main(String[] args) {
        //Load native library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        //image container object
        Mat imageArray;
        //Read image from file system
        imageArray=Imgcodecs.imread("C:\\Users\\mesutpiskin\\sample.jpg");
        //Get image with & height
        System.out.println(imageArray.rows());
        System.out.println(imageArray.cols());
    }
}
```

Get image from webcam

Display a live video feed taken from a webcam using OpenCV's VideoCapture class with Java, C/C++ and Python.

Java

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.videoio.VideoCapture;

public class Camera {
    public static void main(String[] args) {
        // Load Native Library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        // image container object
        Mat imageArray = new Mat();
        // Video device acces
        VideoCapture videoDevice = new VideoCapture();
        // 0:Start default video device 1,2 etc video device id
        videoDevice.open(0);
        // is connected
        if (videoDevice.isOpened()) {
            // Get frame from camera
            videoDevice.read(imageArray);
            // image array
            System.out.println(imageArray.toString());
            // Release video device
            videoDevice.release();
        } else {
            System.out.println("Error.");
        }
    }
}
```

```
}
```

C++

```
#include "opencv2/opencv.hpp"
#include "iostream"

int main(int, char**) {
    // open the first webcam plugged in the computer
    cv::VideoCapture camera(0);
    if (!camera.isOpened()) {
        std::cerr << "ERROR: Could not open camera" << std::endl;
        return 1;
    }

    // create a window to display the images from the webcam
    cv::namedWindow("Webcam", CV_WINDOW_AUTOSIZE);

    // this will contain the image from the webcam
    cv::Mat frame;

    // capture the next frame from the webcam
    camera >> frame;

    // display the frame until you press a key
    while (1) {
        // show the image on the window
        cv::imshow("Webcam", frame);
        // wait (10ms) for a key to be pressed
        if (cv::waitKey(10) >= 0)
            break;
    }
    return 0;
}
```

Python

```
import numpy as np
import cv2

# Video source - can be camera index number given by 'ls /dev/video*'
# or can be a video file, e.g. '~/Video.avi'
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()
```

Getting Started with OpenCV 3.1 on Windows

We install OpenCV 3.1.0 on Windows and get started. There are two ways to install OpenCV on windows. One is to download the installer and run it. Other is to build from source.

This is the easiest way to install OpenCV and get started. OpenCV gives pre-build binaries to install on Windows [here](#). After it finishes downloading, extract it and install at the chosen path.

ProTip: Make sure your OpenCV path doesn't include any spaces. So, it's better if you just install it in C:\ or D:\ root directory

The problem with the above method is that you cannot use the `opencv_contrib` modules. Also, it doesn't come with all the 3rd party tools and libraries. So, if you want to use all of these, just follow along.


I will explain the least minimum to install OpenCV from source. For more advanced one, refer [here](#).

- Install [CMake](#).
- Clone OpenCV source from <https://github.com/Itseez/opencv.git> in some directory which doesn't have spaces. Lets refer to it as "OpenCVdir".

 Itseez / opencv

 Code

 Issues 880

 Pull requests 32

 Wiki

 Pulse

Open Source Computer Vision Library <http://opencv.org>

 18,556 commits

 3 branches

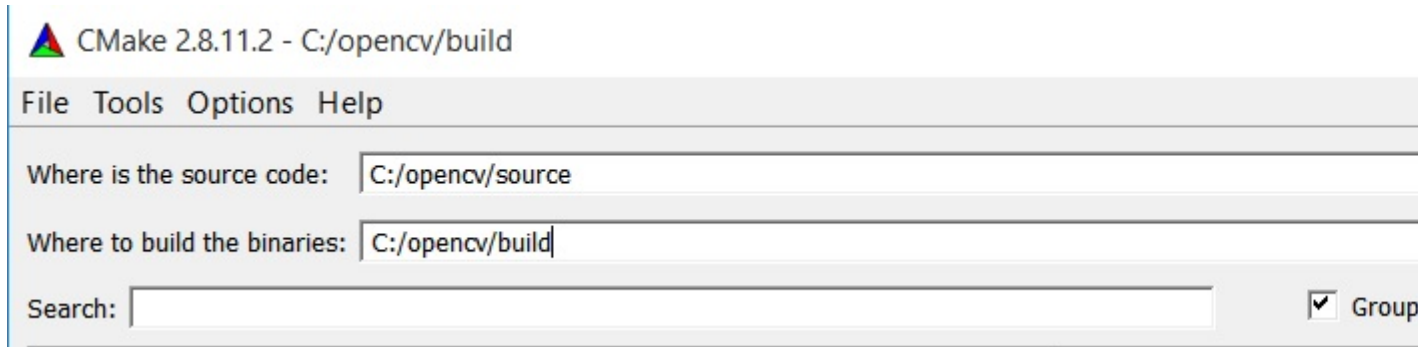
Branch: master ▼

New pull request

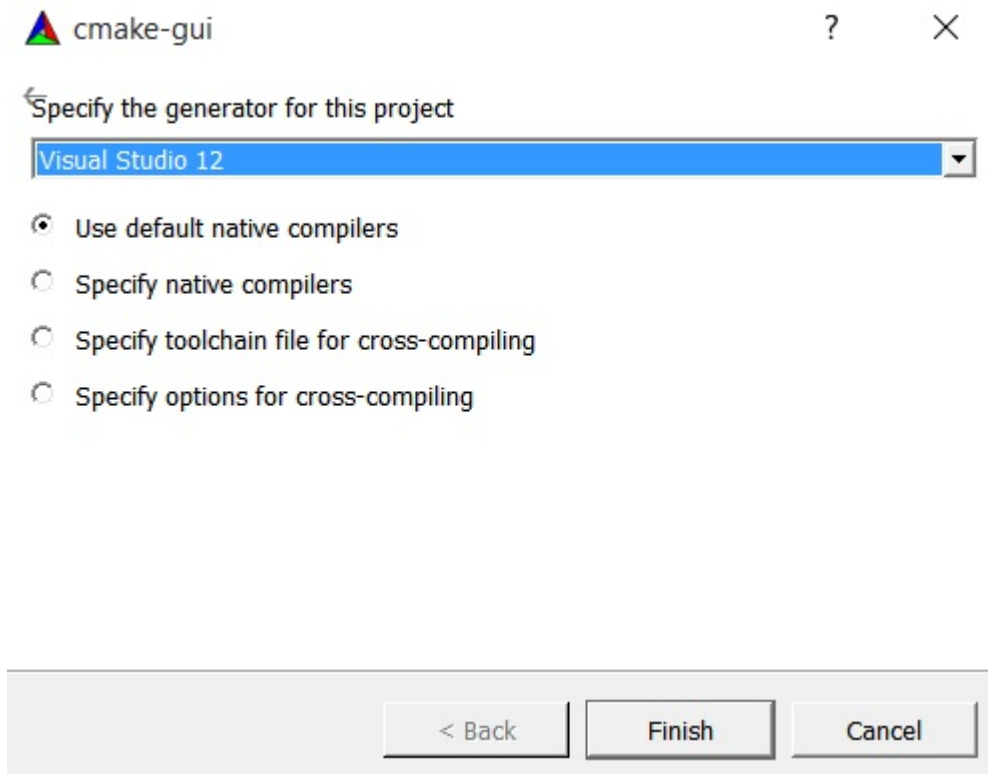
New file

File

- Now, open CMake GUI and add your source directory (OpenCVdir) to the Sources menu and build the directory to the build menu. **Tip:** If there's no build directory, create one in your opencv folder.



- Click on Configure and select your Visual Studio compiler version. I had Visual Studio 2013 Professional 32-bit, so I chose Visual Studio 12 compiler.



Tip: You can download Visual Studio 2013 Professional from [here](#). It comes with 30 days trial + 90 days extended trail after signing in.

- Press Finish and CMake will load all the packages automatically. You can add or remove packages. Press Configure again.
- If you want to build with extra opencv_contrib modules, you need to download them from [here](#). Then, extract them and add the opencv_contrib/modules directory to your CMake as shown below.

Where is the source code: C:/opencv/source

Where to build the binaries: C:/opencv/build

Search:

☒ Grouped

Name	Value
+ CUDA	
+ DOXYGEN	
+ ENABLE	
+ GIGAPI	
+ INSTALL	
+ JAVA	
+ MATLAB	
- OPENCV	
OPENCV_CONFIG_FILE_INCLUDE_DIR	C:/opencv/build11
OPENCV_EXTRA_MODULES_PATH	C:/opencv/source/opencv
OPENCV_WARNINGS_ARE_ERRORS	<input type="checkbox"/>
+ PYTHON2	

Press Configure to update and display new values in red, then press Generate to generate the build files.

Configure

Generate

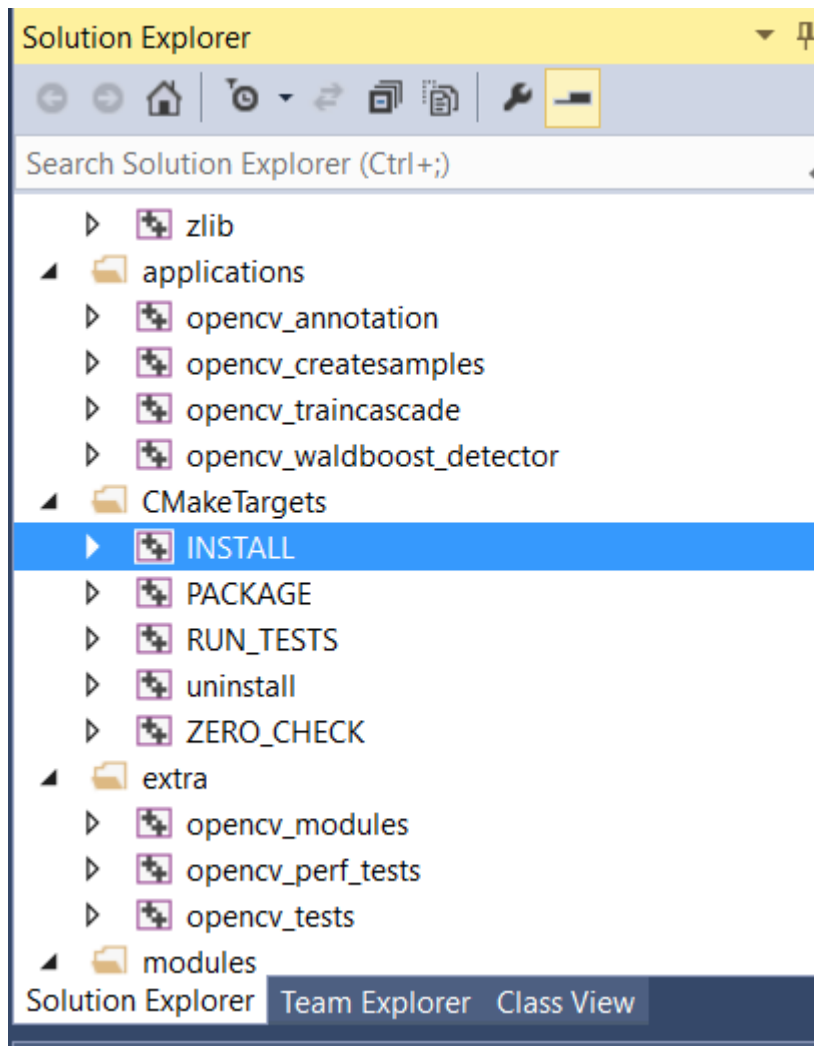
Current Generator: Visual Studio 12

- Now press Configure again and then press Generate.
- Close CMake. Go to your _opencv\build folder and open the file named 'OpenCV.sln' file. - It will open Visual Studio. Now, Run it in both Debug

Local Windows Debugger mode and Release

Local Windows Debugger mode.

- Now, in the solution explorer at the top right of your Visual Studio, select INSTALL project and build it.







Hurray!! Enjoy your OpenCV.

Adding OpenCV include directory to Environment Variables' PATH variable:

- Go to System Properties and Click on Advanced System Settings.

Control Panel Home

-  Device Manager
-  Remote settings
-  System protection
-  [Advanced system settings](#)



View basic information about your computer

Windows edition

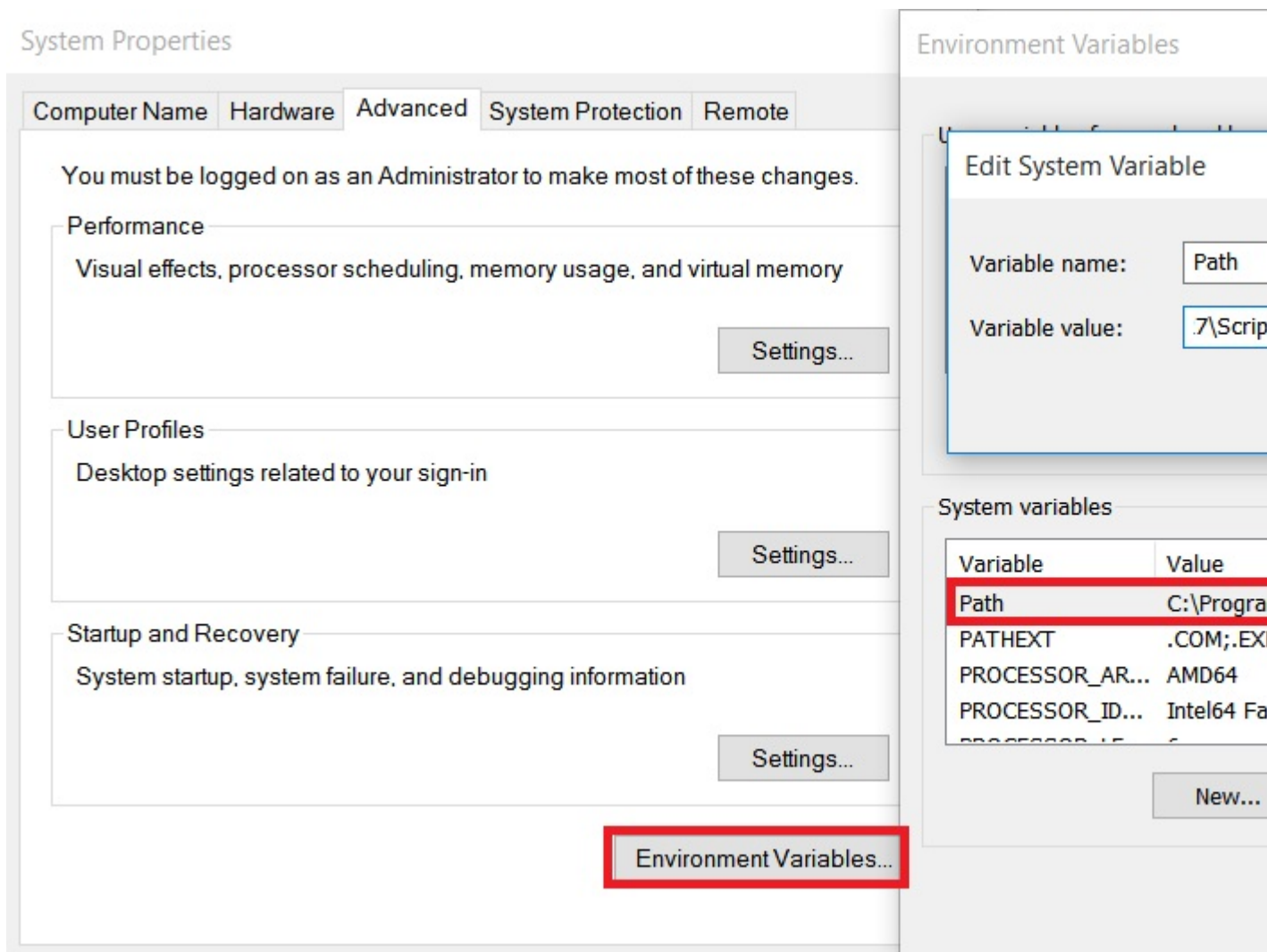
Windows 10 Home Single Language

© 2015 Microsoft Corporation. All rights reserved.

System

Processor:	Intel(R) Core(TM) i5-4210U CPU (1.7 GHz)
Installed memory (RAM):	6.00 GB (5.89 GB usable)
System type:	64-bit Operating System, x64-based processor
Pen and Touch:	No Pen or Touch Input is available for this device

- Now, Click on Environment Variables >> Path >> Edit.



- Here, add the bin folder located in your OpenCVdir/build/install/x86/vc**/bin to this variable. Be careful not to replace the existing Path values.
- After this, you need to restart your system for the Environment variables to change and now you're ready to go.

What & Why OPENCV?

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. It was built for various purpose such as machine learning, computer vision, algorithm, mathematical operations, video capturing, image processing etc. Over the years it has become very popular among the researchers and developers as for its support in different platforms (windows, Linux, android, ios). Also it has wrapper in various renowned programming languages. Under the license agreement, it has access for businesses to utilize and modify the code.

The library contains more than 2500 optimized algorithms, which has excellent accuracy in performance and speed. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database,

remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has great people and community involved as users, developers and researchers, the number is more than 47 thousand and estimated number of downloads exceeding 7 million. The library is extensively in professional companies, research groups and other groups.

Many well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan. It has C++, C, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

Information collected from the [official website](#)

Load and display an image with OpenCV

With this example, we will see how to load a color image from disk and display it using OpenCV's built-in functions. We can use the C/C++, Python or Java bindings to accomplish this.

In C++:

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

#include <iostream>

using namespace cv;

int main(int argc, char** argv) {
    // We'll start by loading an image from the drive
    Mat image = imread("image.jpg", CV_LOAD_IMAGE_COLOR);

    // We check that our image has been correctly loaded
    if(image.empty()) {
        std::cout << "Error: the image has been incorrectly loaded." << std::endl;
        return 0;
    }

    // Then we create a window to display our image
    namedWindow("My first OpenCV window");

    // Finally, we display our image and ask the program to wait for a key to be pressed
    imshow("My first OpenCV window", image);
```

```

    waitKey(0);

    return 0;
}

```

In Python:

```

import sys
import cv2

# We load the image from disk
img = cv2.imread("image.jpg", cv2.CV_LOAD_IMAGE_COLOR)

# We check that our image has been correctly loaded
if img.size == 0:
    sys.exit("Error: the image has not been correctly loaded.")

# We create a window to display our image
cv2.namedWindow("My first OpenCV window")

# We display our image and ask the program to wait until a key is pressed
cv2.imshow("My first OpenCV window", img)
cv2.waitKey(0)

# We close the window
cv2.destroyAllWindows()

```

In Java:

```

import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.CvType;
import org.opencv.highgui.Highgui;
public class Sample{
    public static void main (String[] args) {

        //Load native opencv library
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);

        //Read image from file first param:file location ,second param:color space
        Mat img = imread("image.jpg",CV_LOAD_IMAGE_COLOR);

        //If the image is successfully read.
        if (img.size() == 0) {
            System.exit(1);
        }
    }
}

```

HighGui has no namedwindows or imshow equivalents in opencv java. Use swing or swt to display image.

Read Getting started with opencv online: <http://www.riptutorial.com/opencv/topic/800/getting-started-with-opencv>

Chapter 2: Basic Structures

Introduction

This topic covers basic structures in OpenCV. The structures that will be discussed in this topic are `DataType`, `Point`, `Vec`, `Size`, `Rect`, `Scalar`, `Ptr` and `Mat`.

Examples

DataType

The primitive types in OpenCV are `unsigned char`, `bool`, `signed char`, `unsigned short`, `signed short`, `int`, `float`, `double`. Any data type in OpenCV is defined as `CV_<bit-depth>{U|S|F}C(<number_of_channels>)` where `U`: unsigned, `S`:signed and `F`:floating point.

For example, `CV_32FC2` is a 32-bit, floating-point, and 2-channels structure. and the definition of basic, one channel types are

```
#define CV_8U    0
#define CV_8S    1
#define CV_16U   2
#define CV_16S   3
#define CV_32S   4
#define CV_32F   5
#define CV_64F   6
#define CV_USRTYPE1 7
```

The other types with higher channel are produced from these by the following definition:

```
#define CV_MAKETYPE(depth,cn) (CV_MAT_DEPTH(depth) + (((cn)-1) << CV_CN_SHIFT))
```

Using these datatypes other structures can be created.

Mat

`Mat` (Matrix) is an n-dimensional array that can be used to store various type of data, such as RGB, HSV or grayscale images, vectors with real or complex values, other matrices etc.

A `Mat` contains the following information: `width`, `height`, `type`, `channels`, `data`, `flags`, `datastart`, `dataend` and so on.

It has several methods, some of them are: `create`, `copyTo`, `convertTo`, `isContinuous` etc.

There are many ways to create a `Mat` variable. Consider I want to create a matrix with 100 rows, 200 columns, type `CV_32FC3`:

```
int R = 100, C = 200;
```

```
Mat m1; m1.create(R,C,CV_32FC3); //creates empty matrix
Mat m2(cv::Size(R, C), CV_32FC3); // creates a matrix with R rows, C columns with data type T
where R and C are integers,
Mat m3(R,C,CV_32FC3); // same as m2
```

Initializing Mat:

```
Mat m1 = Mat::zeros(R,C,CV_32FC3); // This initialized to zeros, you can use one, eye or
cv::randn etc.
Mat m2(R,C,CV_32FC3);
for (int i = 0; i < m2.rows; i++)
    for (int j = 0; j < m2.cols; j++)
        for (int k = 0; k < m2.channels(); k++)
            m2.at<Vec3f>(i,j)[k] = 0;
//Note that, because m2 is a float type and has 3 channels, we used Vec3f, for more info see
Vec

Mat m3(3, out, CV_32FC1, cv::Scalar(0));
```

Vec

`Vec` (Vector) is a template class for numerical values. Unlike `c++ vectors`, it generally stores short vectors (only a few elements).

The way a `Vec` is defined is as follows:

```
typedef Vec<type, channels> Vec< channels>< one char for the type>;
```

where `type` is one of `uchar`, `short`, `int`, `float`, `double` and the characters for each type are `b`, `s`, `i`, `f`, `d`, respectively.

For example, `Vec3b` indicates an unsigned char vector of 3 channels. Each index in an RGB image is in this format.

```
Mat rgb = imread('path/to/file', CV_LOAD_IMAGE_COLOR);
cout << rgb.at<Vec3b>(0,0); //The output is [r g b] values as ASCII character.
// To print integer values of RED value
cout << (int)rgb.at<Vec3b>(0,0)[0]; //The output will be an integer in [0, 255].
```

In `Vec` class the following operators are defined

```
v1 = v2 + v3
v1 = v2 - v3
v1 = v2 * scale
v1 = scale * v2
v1 = -v2
v1 += v2 and other augmenting operations
v1 == v2, v1 != v2
```

For more information, see the [link](#)

Read Basic Structures online: <http://www.riptutorial.com/opencv/topic/9099/basic-structures>

Chapter 3: Blob Detection

Examples

Circular Blob Detection

This example shows how to find circular blobs in an grayscale image. The evaluation of the circularity of a blob is done using the area and the perimeter (arc length) of the contour. The center point gets evaluated using the moments of the contour.

```
#include "opencv/cv.h"
#include "opencv/highgui.h"
#include "opencv/cxcore.h"

using namespace cv;

int main(int argc, char** argv)
{
    Mat img = imread("image.jpg", CV_LOAD_IMAGE_GRAYSCALE);
    Mat resultImg;
    cvtColor(img, resultImg, CV_GRAY2BGR);

    // threshold the image with gray value of 100
    Mat binImg;
    threshold(img, binImg, 100, 255, THRESH_BINARY);

    // find the contours
    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;
    findContours(binImg, contours, hierarchy, CV_RETR_CCOMP, CV_CHAIN_APPROX_SIMPLE);

    if(contours.size() <= 0)
    {
        printf("no contours found");
        return 0;
    }
    // filter the contours
    vector<vector<Point>> filteredBlobs;
    Mat centers = Mat::zeros(0,2,CV_64FC1);
    for(int i = 0; i < contours.size(); i++)
    {
        // calculate circularity
        double area = contourArea(contours[i]);
        double arclength = arcLength(contours[i], true);
        double circularity = 4 * CV_PI * area / (arclength * arclength);
        if(circularity > 0.8)
        {
            filteredBlobs.push_back(contours[i]);

            //calculate center
            Moments mu = moments(contours[i], false);
            Mat centerpoint = Mat(1,2,CV_64FC1);
            centerpoint.at<double>(i,0) = mu.m10 / mu.m00; // x-coordinate
            centerpoint.at<double>(i,1) = mu.m01 / mu.m00; // y-coordinate
            centers.push_back(centerpoint);
        }
    }
}
```

```
}

if(filteredBlobs.size() <= 0)
{
    printf("no circular blobs found");
    return 0;
}
drawContours(resultImg, filteredBlobs, -1, Scalar(0,0,255), CV_FILLED, 8);

imshow("Blobs",resultImg);
waitKey(0);
return 0;
}
```

Read Blob Detection online: <http://www.riptutorial.com/opencv/topic/6589/blob-detection>

Chapter 4: Build and Compile opencv 3.1.0-dev for Python2 on Windows using CMake and Visual Studio

Remarks

Building and Compiling `opencv 3.1.0-dev` to get an access for **non free modules** can be a headache for some people especially on Windows machine. Unlike Ubuntu, setting up opencv for Windows takes some time and requires a couple of dependencies to be installed first before building and compiling.

The programs you should download and install before going further in any step are:

1. [Python 2.7.x](#) or [Python 3.x.x](#)
2. [CMake](#)

If you are going to download either Python for Win32, you should also download CMake for Win32 even if you are using a 64-bit machine.

It's recommended to download the 32-bit programs because some Python libraries are only supported for 32-bit machines, so to stay away from troubles, just install everything in 32-bit version.

3. [Visual Studio Community 2013](#)
4. [Numpy](#) for Python2.7 Win32

After Installing all the above dependencies, **restart** your PC and then you will be ready to continue to the next step.

Step 2:

If you are not the type of person who prefers to read, you can watch this [tutorial](#). The tutorial takes you from here to the end of this documentation.

You will need to get **opencv** and **opencv_contrib** from **github**. You can find both at:

1. [opencv](#)
2. [opencv_contrib](#)

Create a directory named `opencv-3.1.0` where in this director you will make another two directories one for the *build* and one for the *sources*. You will put the two downloaded zip files in the sources file after extraction.

For example your `opencv-3.1.0` directory is located in the C drive, so you will have three paths:

1. C:\opencv-3.1.0
2. C:\opencv-3.1.0\build
3. C:\opencv-3.1.0\sources

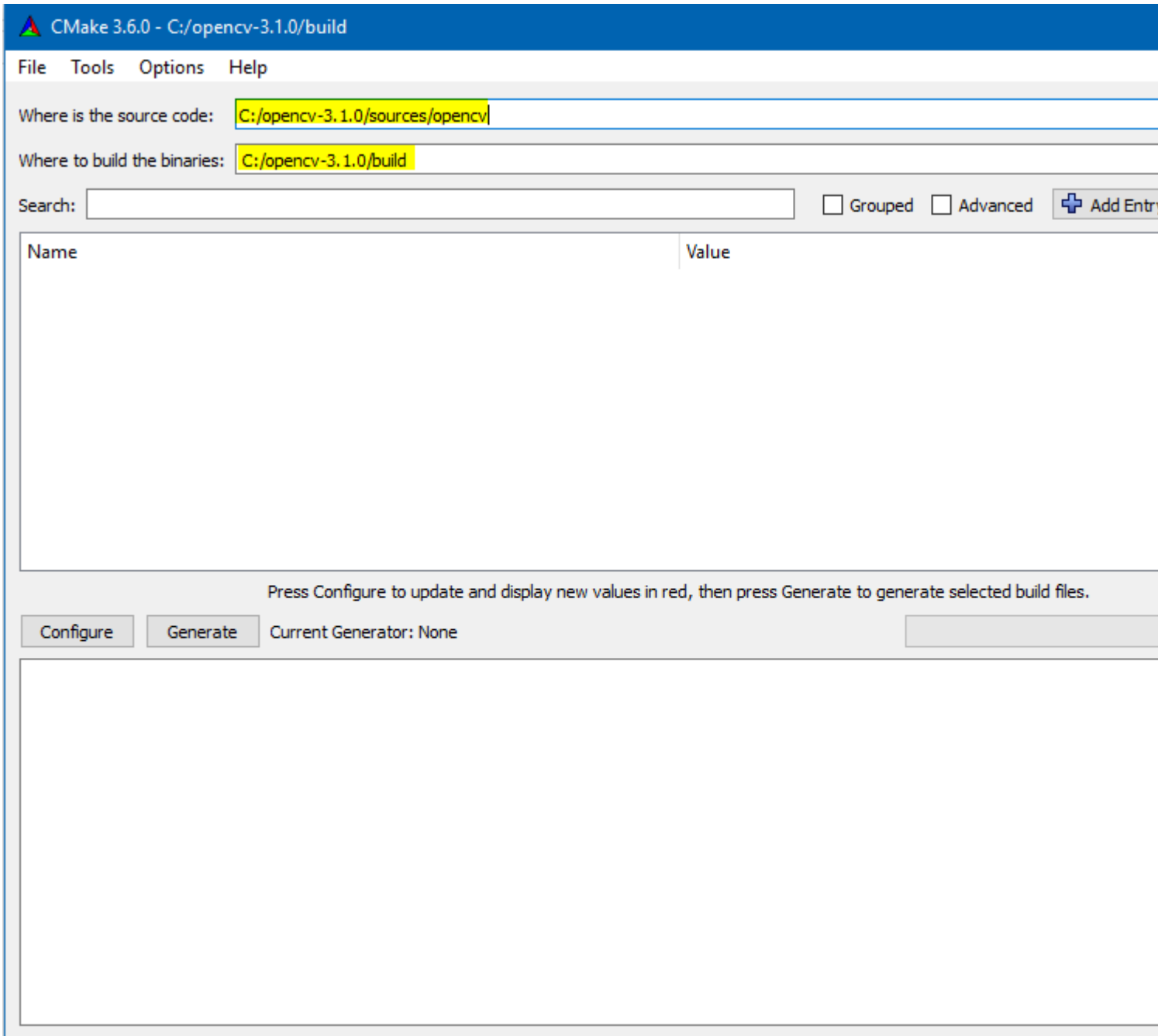
The third directory will include two paths:

1. C:\opencv-3.1.0\sources\opencv
2. C:\opencv-3.1.0\sources\opencv_contrib

Now it's done with preparation. Lets make some useful stuff.

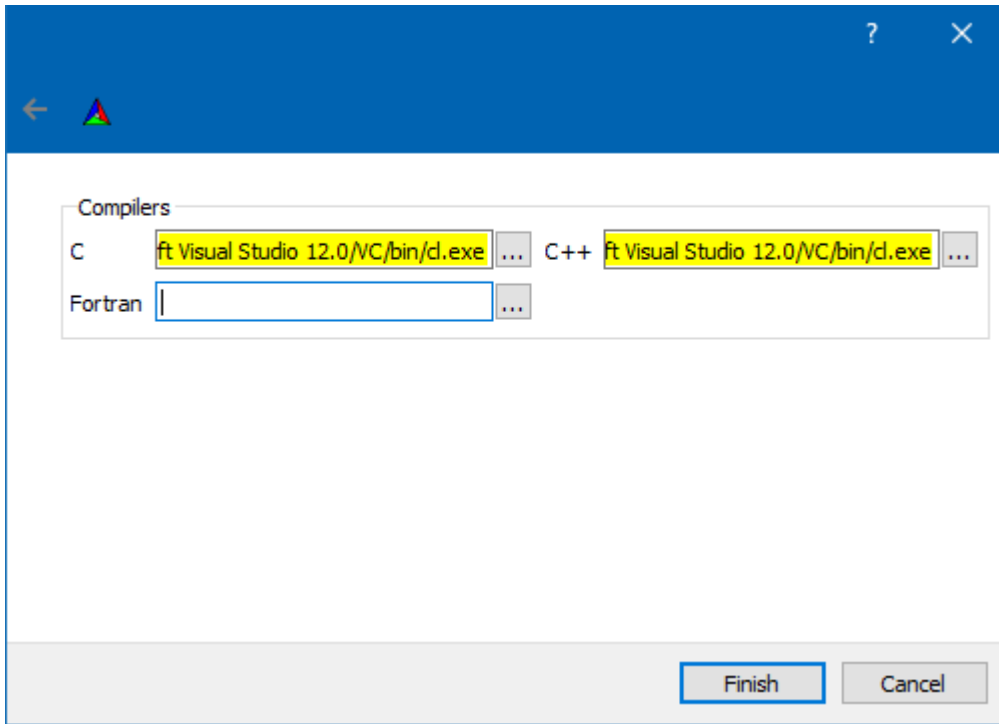
Step 3:

Run CMake as an administrator. A window like this will appear and you will have to provide two directories one for the sources and the other for where the opencv will be compiled. The below image can help you more better than words.

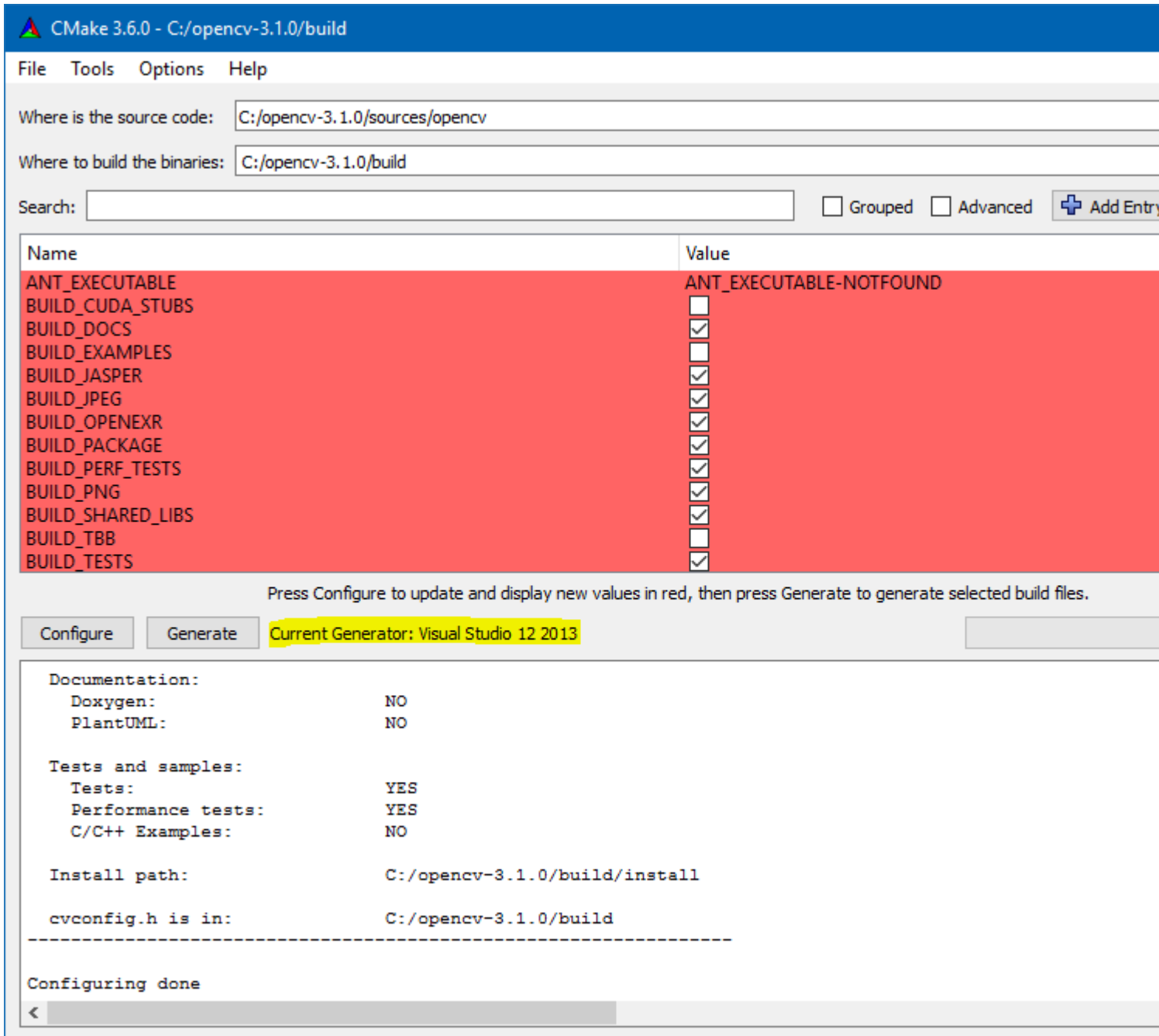


Next click **configure** and you will be promoted to provide the generators; i.e. compilers; for opencv. You have to provide the `cl.exe` located in Microsoft Visual Studio 2013. Click **specify native generators** and a pop up window like the following will appear,

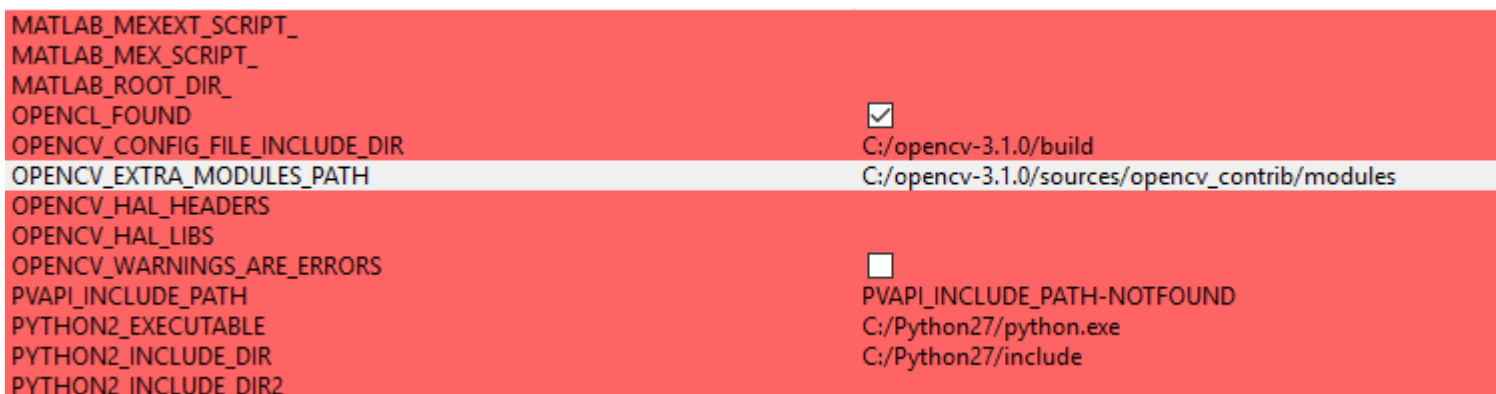
The paths will be something like this: `C:/Program Files (x86)/Microsoft Visual Studio 12.0/VC/bin/cl.exe`. Provide your path for both C and C++ fields. Click finish and wait until configuring is done. You should get zero errors if you were following all the previous steps correctly.



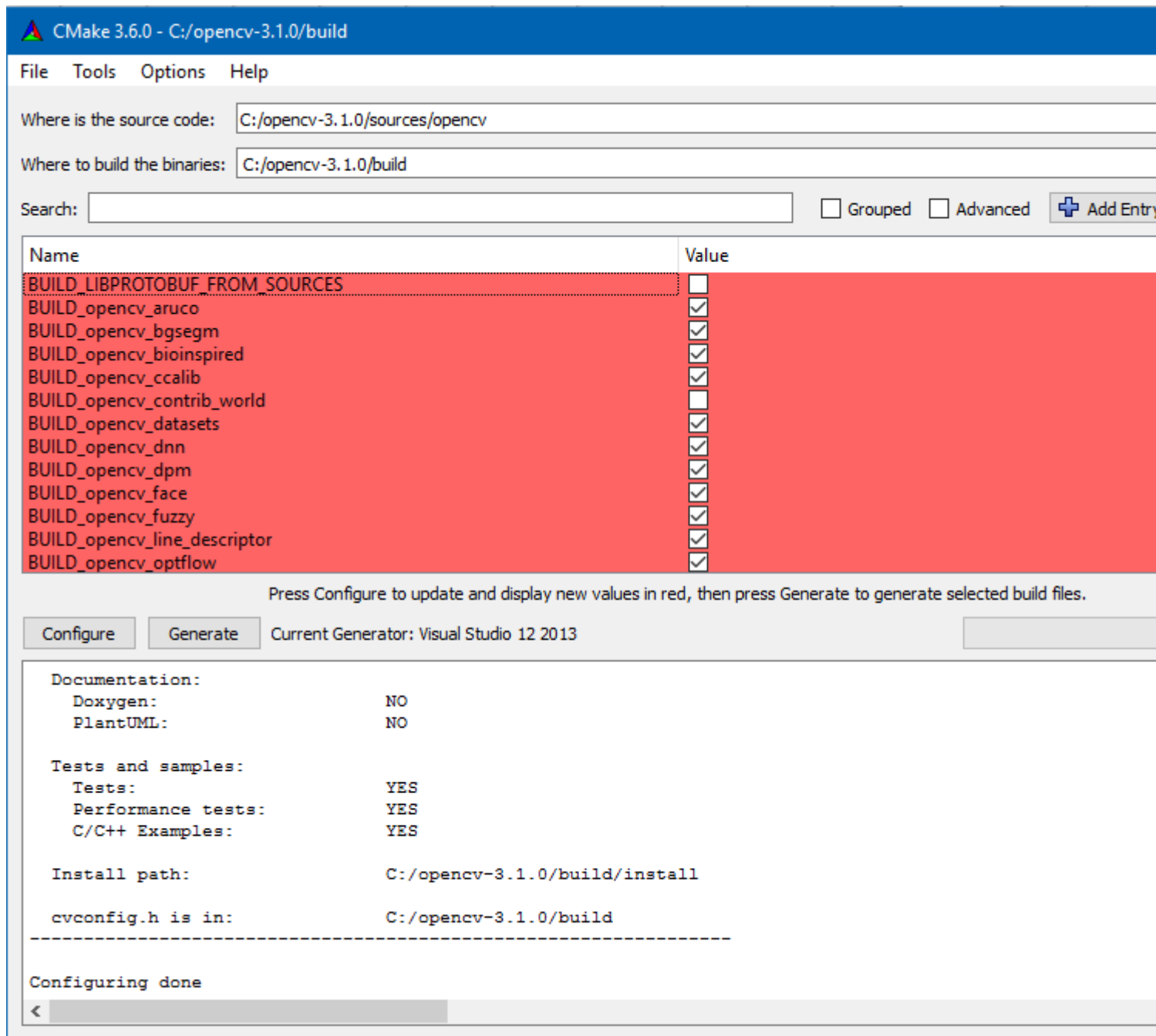
After CMake finishes configuring, you will see new items appearing in the CMake window that are highlighted in red. It will be something like:



Check the builds you need by clicking on the small square box. Search for `OPENCV_EXTRA_MODULES_PATH` line and provide the modules directory within `opencv_contrib` within the sources directory.



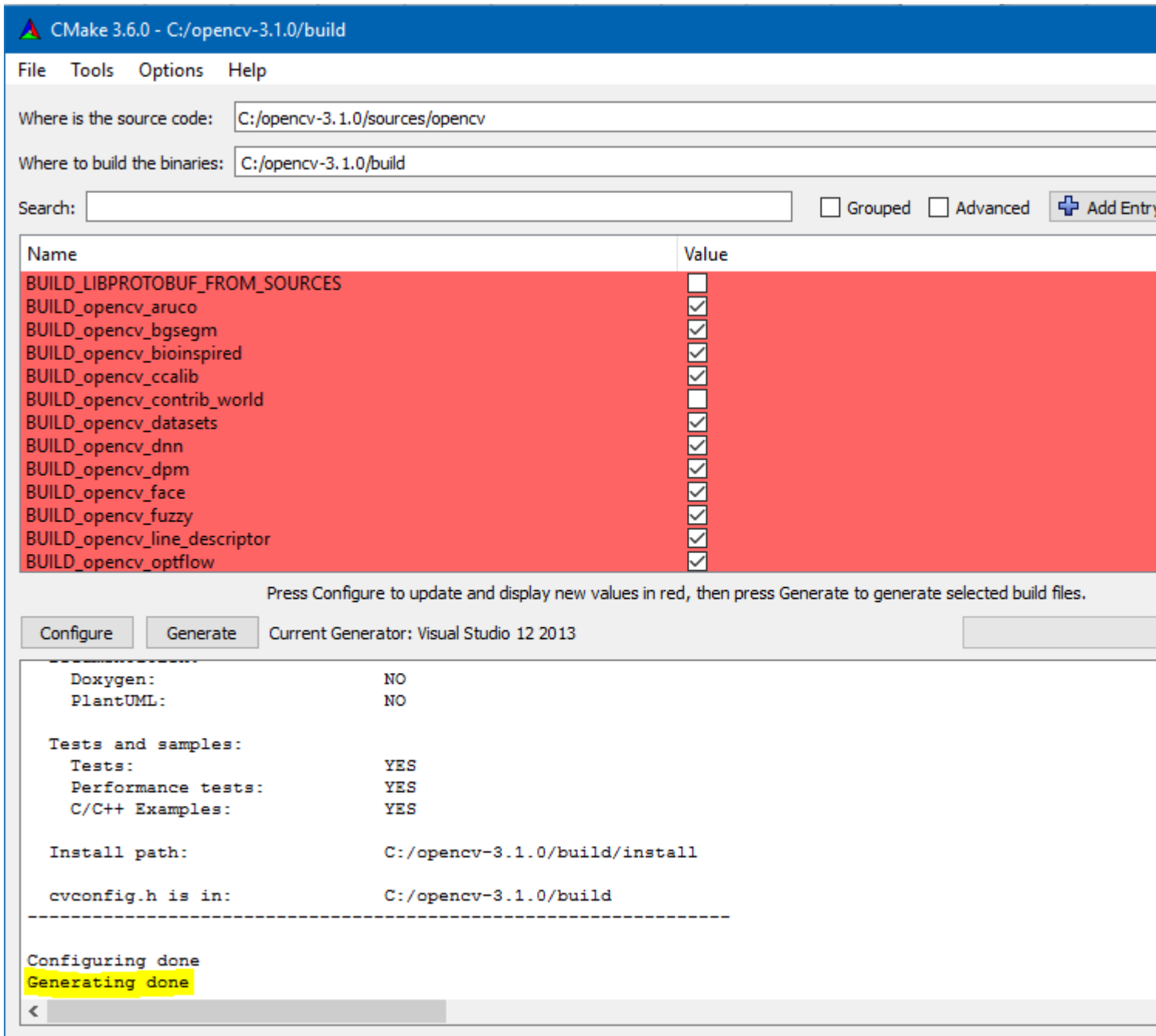
Once you have finished all you need and provided the path for the extra modules press configure again to update. The previously highlighted lines will no longer be highlighted and new fields will be highlighted in red instead.



Also check the boxes for whatever you need to build.

Make sure that **BUILD_opencv_contrib_world** and **BUILD_opencv_world** are both **unchecked**. There is probably a bug where an error occurs when any of the latter are checked.

At the end of this step click **Generate** and you will be done with CMake and you can close it. *If there are no errors, you will get a message at the end of the lower pane saying **Generating done**.*



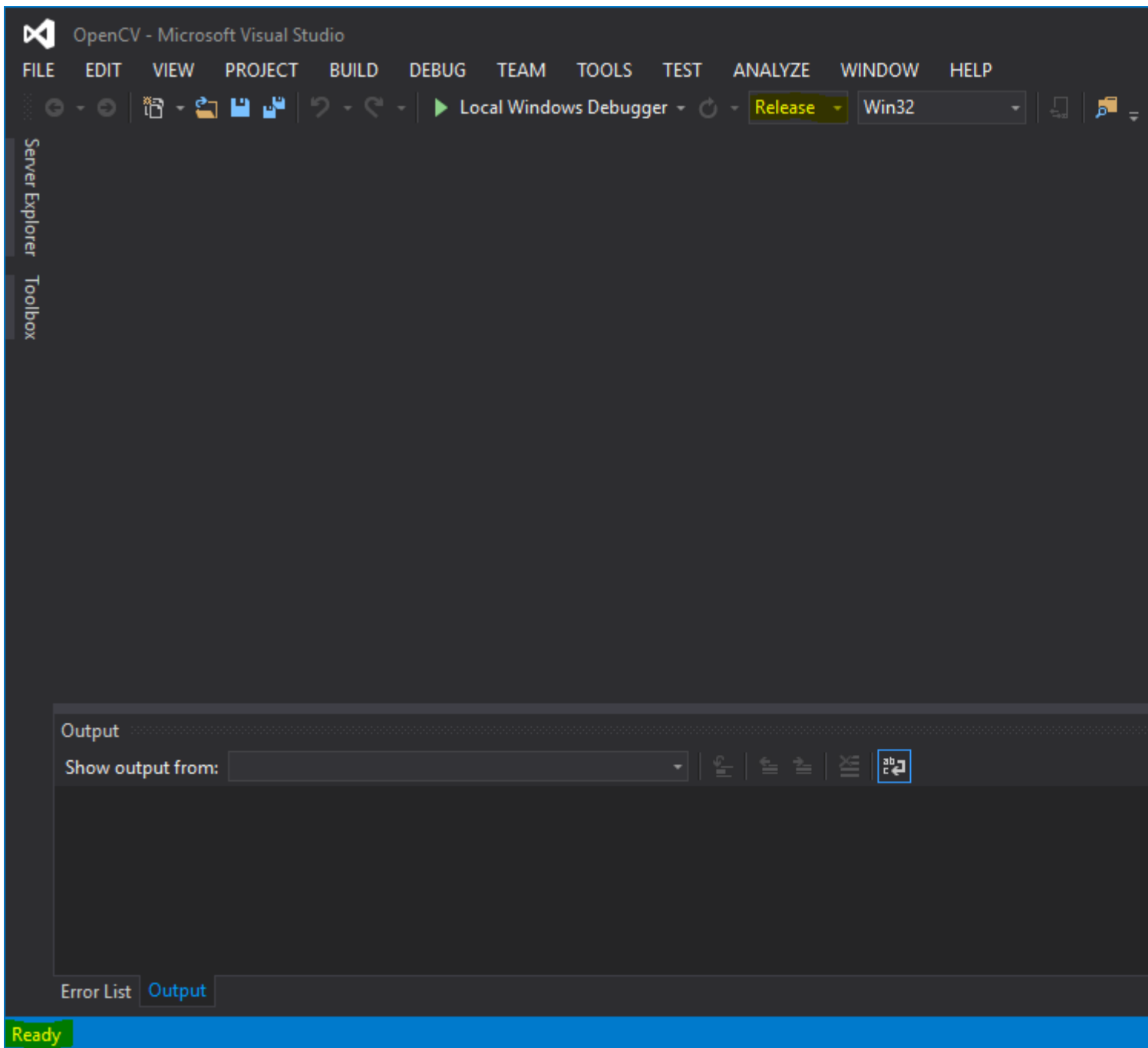
Step 4:

Open up the build directory located in opencv-3.1.0 and you will find a bunch of new folders and files inside it. It was an empty folder at the beginning of this process.

You will only deal with `opencv.sln` file and don't do anything with the rest files. Open this file with the version that used while compiling in the CMake in the previous step. It has to be `visual Microsoft 2013`.

Name	Date modified	Type	Size
samples	7/30/2016 8:52 PM	File folder	
test-reports	7/30/2016 8:38 PM	File folder	
unix-install	7/30/2016 8:46 PM	File folder	
win-install	7/30/2016 8:46 PM	File folder	
ALL_BUILD.vcxproj	7/30/2016 8:52 PM	VC++ Project	88 KB
ALL_BUILD.vcxproj.filters	7/30/2016 8:52 PM	VC++ Project Filte...	1 KB
cmake_install.cmake	7/30/2016 8:52 PM	CMAKE File	7 KB
cmake_uninstall.cmake	7/30/2016 8:38 PM	CMAKE File	2 KB
CMakeCache.txt	7/30/2016 8:46 PM	Text Document	244 KB
CMakeVars.txt	7/30/2016 8:46 PM	Text Document	407 KB
CPackConfig.cmake	7/30/2016 8:46 PM	CMAKE File	10 KB
CPackSourceConfig.cmake	7/30/2016 8:46 PM	CMAKE File	10 KB
CTestTestfile.cmake	7/30/2016 8:52 PM	CMAKE File	1 KB
custom_hal.hpp	7/30/2016 8:38 PM	C/C++ Header	1 KB
cvconfig.h	7/30/2016 8:38 PM	C/C++ Header	5 KB
INSTALL.vcxproj	7/30/2016 8:52 PM	VC++ Project	7 KB
INSTALL.vcxproj.filters	7/30/2016 8:52 PM	VC++ Project Filte...	1 KB
OpenCV.sln	7/30/2016 8:53 PM	Microsoft Visual S...	948 KB
opencv_modules.vcxproj	7/30/2016 8:52 PM	VC++ Project	28 KB
opencv_modules.vcxproj.filters	7/30/2016 8:52 PM	VC++ Project Filte...	1 KB
opencv_perf_tests.vcxproj	7/30/2016 8:52 PM	VC++ Project	24 KB
opencv_perf_tests.vcxproj.filters	7/30/2016 8:52 PM	VC++ Project Filte...	1 KB
opencv_tests.vcxproj	7/30/2016 8:52 PM	VC++ Project	26 KB
opencv_tests.vcxproj.filters	7/30/2016 8:52 PM	VC++ Project Filte...	1 KB
OpenCVConfig.cmake	7/30/2016 8:46 PM	CMAKE File	19 KB
OpenCVConfig-version.cmake	7/30/2016 8:38 PM	CMAKE File	1 KB
OpenCVModules.cmake	7/30/2016 8:53 PM	CMAKE File	47 KB
PACKAGE.vcxproj	7/30/2016 8:52 PM	VC++ Project	7 KB
PACKAGE.vcxproj.filters	7/30/2016 8:52 PM	VC++ Project Filte...	1 KB

When you open the .sln file, please be patient as it takes some time to prepare everything for building. When **Ready** is steady (not changing) you can start building your targets. Start building as numbered in the image below. Also make sure that the Solution Configuration is Release not Debug.



Step 5:

When building is finished, you will need to copy and paste a couple of files from the build directory into the `Python27` directory.

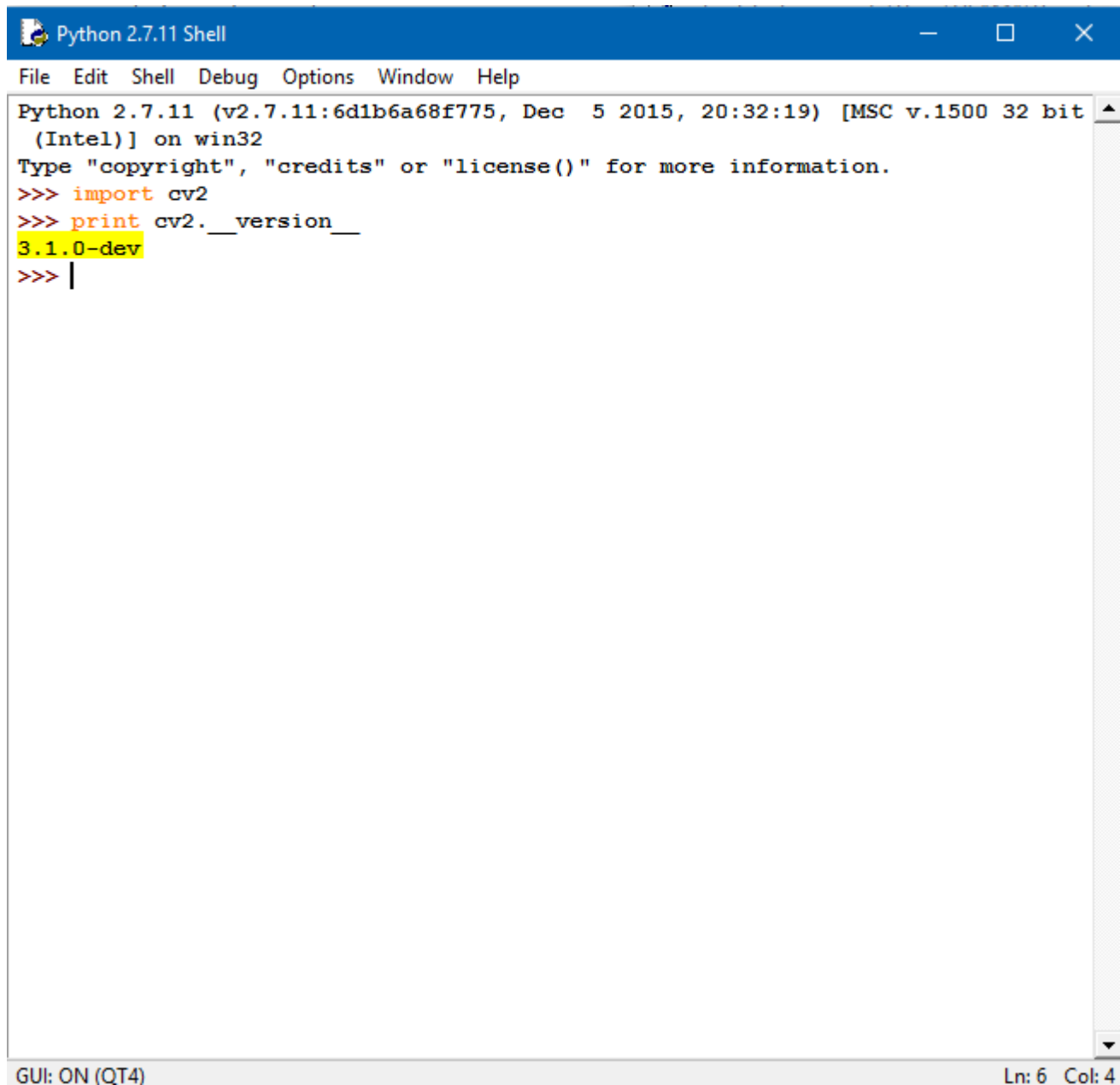
Search for `cv2.pyd` file and copy it to the `site-packages` directory in `Python27`. The `cv2.pyd` should be present in `C:\opencv-3.1.0\build\lib\Release`. After that, copy **only** the `.dll` files inside `C:\opencv-3.1.0\build\bin\Release` into the parent directory of `Python27` at this location `C:\Python27`.

At the end of this step, restart you PC.

Verification:

Open IDLE and within the Python shell type:

```
>>> import cv2
>>> print cv2.__version__
3.1.0-dev
```



Examples

Reading Image and Converting into grayscale

```
import cv2
import numpy as np

img = cv2.imread('<your_image>')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cv2.imshow('image', img)
```

```
cv2.imshow('gray', gray)

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

Read Build and Compile opencv 3.1.0-dev for Python2 on Windows using CMake and Visual Studio online: <http://www.riptutorial.com/opencv/topic/6100/build-and-compile-opencv-3-1-0-dev-for-python2-on-windows-using-cmake-and-visual-studio>

Chapter 5: Cascade Classifiers

Examples

Using Cascade Classifiers to detect face

Python

Code

```
import numpy as np
import cv2

#loading haarcascade classifiers for face and eye
#You can find these cascade classifiers here
#https://github.com/opencv/opencv/tree/master/data/haarcascades
#or where you download opencv inside data/haarcascades

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

#loading the image
img = cv2.imread('civil_war.jpg')

#converting the image to gray scale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#detecting face in the grayscale image
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

#iterate through each detected face
for (x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2) #draw rectangle to each detected face

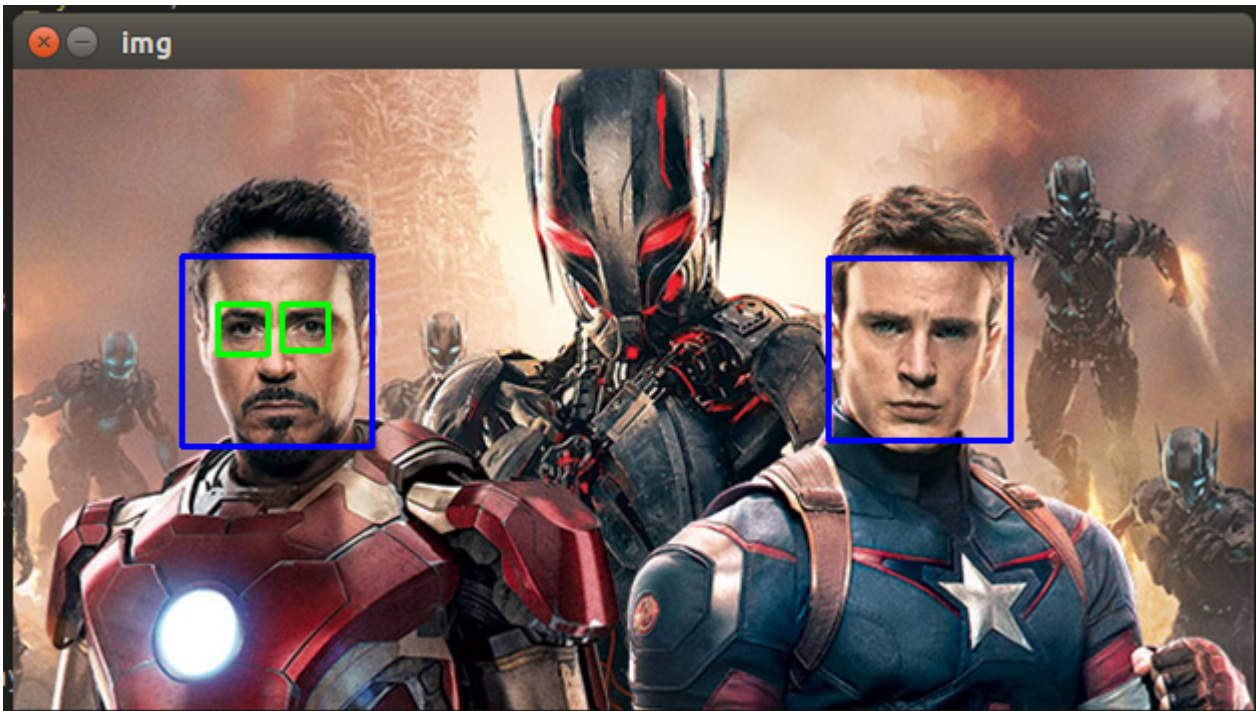
    #take the roi of the face (region of interest)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]

    #detect the eyes
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:

        #draw rectangle for each eye
        cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh), (0,255,0),2)

#show the image
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result



Face detection using haar cascade classifier

C++

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>
#include <stdio.h>

using namespace std;
using namespace cv;

// Function Headers
void detectAndDisplay(Mat frame);

// Global variables
string face_cascade_name = "../data/haarcascade_frontalface_alt2.xml";
CascadeClassifier face_cascade;

// Function main
int main(void)
{
    // Load the cascade
    if (!face_cascade.load(face_cascade_name)) {
        printf("--(!)Error on cascade loading\n");
        return (-1);
    }

    // Read the image file
    Mat frame = imread("d:/obama_01.jpg");

    // Apply the classifier to the frame
    if (!frame.empty())
        detectAndDisplay(frame);
}
```

```

    waitKey(0);
    return 0;
}

// Function detectAndDisplay
void detectAndDisplay(Mat frame)
{
    std::vector<Rect> faces;
    Mat frame_gray;

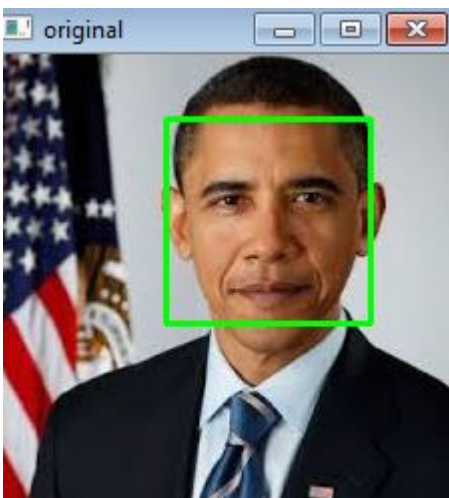
    cvtColor(frame, frame_gray, COLOR_BGR2GRAY);
    equalizeHist(frame_gray, frame_gray);

    // Detect faces
    face_cascade.detectMultiScale(frame_gray, faces, 1.1, 2, 0 | CASCADE_SCALE_IMAGE, Size(30,
30));

    for (int ic = 0; ic < faces.size(); ic++) // Iterate through all current elements
(detected faces)
    {
        Point pt1(faces[ic].x, faces[ic].y); // Display detected faces on main window - live
stream from camera
        Point pt2((faces[ic].x + faces[ic].height), (faces[ic].y + faces[ic].width));
        rectangle(frame, pt1, pt2, Scalar(0, 255, 0), 2, 8, 0);
    }

    imshow("original", frame);
}

```



Cascade Classifiers to detect face with Java

Java

Code

```

import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.highgui.Highgui;

```

```

import org.opencv.highgui.VideoCapture;
import org.opencv.objdetect.CascadeClassifier;

public class FaceDetector{

    public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        //Create object
        CascadeClassifier faceDetector = new
CascadeClassifier(FaceDetector.class.getResource("haarcascade_frontalface_default.xml").getPath());

        //Read image
        Mat image = Highgui.imread("sourceimage.jpg");

        /*
        //Or read from webcam

        * Mat image=new Mat();
        *VideoCapture videoCapture=new VideoCapture(0);
        *videoCapture.read(image);
        */
        MatOfRect faceDetections = new MatOfRect();
        //Result list
        faceDetector.detectMultiScale(image, faceDetections);

        for (Rect rect : faceDetections.toArray()) {
            //Draw rectangle on result

            Core.rectangle(image, new Point(rect.x, rect.y), new Point(rect.x + rect.width,
rect.y + rect.height),
                new Scalar(0, 255, 0));
        }

        //write result
        Highgui.imwrite("result.png", image);
        System.out.println("Succesfull");
    }

}

```

Result



Read Cascade Classifiers online: <http://www.riptutorial.com/opencv/topic/6562/cascade-classifiers>

Chapter 6: Contrast and Brightness in C++

Syntax

- `void cv::Mat::convertTo(OutputArray m, int rtype, double alpha = 1, double beta = 0) const`

Parameters

Parameter	Details
m	output matrix; if it does not have a proper size or type before the operation, it is reallocated
rtype	desired output matrix type or, rather, the depth since the number of channels are the same as the input has; if rtype is negative, the output matrix will have the same type as the input
alpha	optional scale factor. This changes the contrast of an image. Values below 1 decrease the contrast and above one increases the contrast
beta	optional delta added to the scaled values. Positive values increases the brightness and negative values decreases the brightness

Remarks

Contrast:

Contrast is the difference in luminance or colour that makes an object (or its representation in an image or display) distinguishable. The higher the difference between a pixel and its neighbors the higher the contrast is in that area.

Brightness:

In other words, brightness is the perception elicited by the luminance of a visual target. In terms of pixels, the higher the value of a pixel is the brighter that pixel is.

Contrast and Brightness adjustments:

$$g(i,j) = \alpha \cdot f(i,j) + \beta$$

$f(x)$ as the source image pixels and $g(x)$ as the output image pixels.

i and j indicates that the pixel is located in the i -th row and j -th column.

The parameters $\alpha > 0$ and β are often called the gain and bias parameters; sometimes these

parameters are said to control *contrast* and *brightness* respectively.

Opencv has a function called `convertTo()` which can apply these two operations.

Sources:

http://docs.opencv.org/trunk/d3/d63/classcv_1_1Mat.html#adf88c60c5b4980e05bb556080916978b

<http://opencv-srf.blogspot.ca/2013/07/change-contrast-of-image-or-video.html> <http://opencv-srf.blogspot.ca/2013/07/change-brightness.html>

Examples

Adjusting brightness and contrast of an image in c++

```
// main.cpp : Defines the entry point for the console application.
//
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, const char** argv)
{
    Mat img = imread("lena30.jpg", CV_LOAD_IMAGE_COLOR); //open and read the image

    if (img.empty())
    {
        cout << "Image cannot be loaded..!!" << endl;
        return -1;
    }

    Mat img_higher_contrast;
    img.convertTo(img_higher_contrast, -1, 2, 0); //increase the contrast (double)

    Mat img_lower_contrast;
    img.convertTo(img_lower_contrast, -1, 0.5, 0); //decrease the contrast (halve)

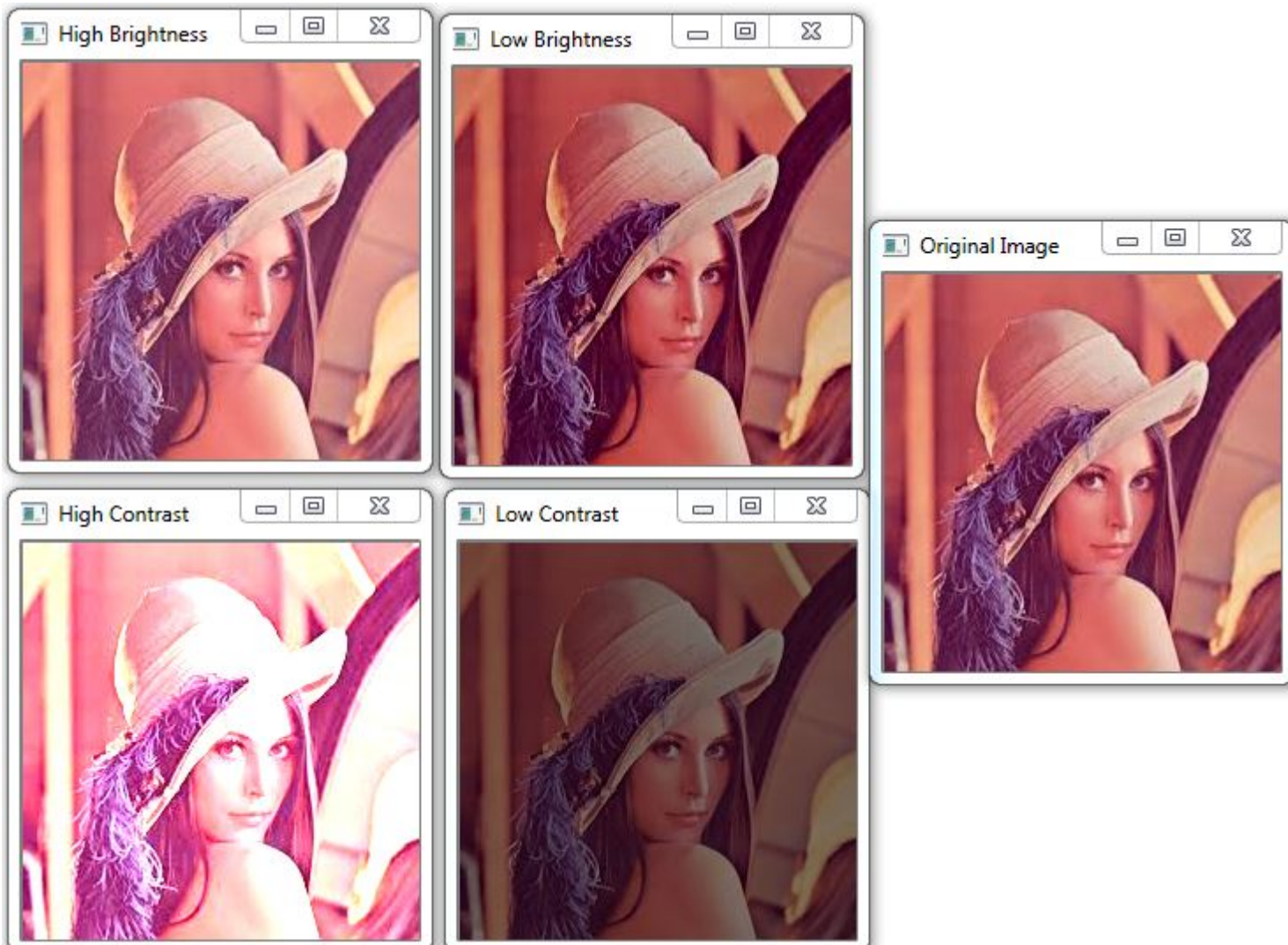
    Mat img_higher_brightness;
    img.convertTo(img_higher_brightness, -1, 1, 20); //increase the brightness by 20 for each
    pixel

    Mat img_lower_brightness;
    img.convertTo(img_lower_brightness, -1, 1, -20); //decrease the brightness by 20 for each
    pixel

    //create windows
    namedWindow("Original Image", CV_WINDOW_AUTOSIZE);
    namedWindow("High Contrast", CV_WINDOW_AUTOSIZE);
    namedWindow("Low Contrast", CV_WINDOW_AUTOSIZE);
    namedWindow("High Brightness", CV_WINDOW_AUTOSIZE);
    namedWindow("Low Brightness", CV_WINDOW_AUTOSIZE);
    //show the image
    imshow("Original Image", img);
    imshow("High Contrast", img_higher_contrast);
```

```
imshow("Low Contrast", img_lower_contrast);  
imshow("High Brightness", img_higher_brightness);  
imshow("Low Brightness", img_lower_brightness);  
  
waitKey(0); //wait for key press  
destroyAllWindows(); //destroy all open windows  
return 0;  
}
```

Output of the program:



Read Contrast and Brightness in C++ online:

<http://www.riptutorial.com/opencv/topic/6917/contrast-and-brightness-in-cplusplus>

Chapter 7: Creating a Video

Introduction

Whenever you work with video feeds you may eventually want to save your image processing result in a form of a new video file. For simple video outputs you can use the OpenCV built-in VideoWriter class, designed for this. It is useful to look at some concepts before using them. These concepts are codec ie decoder and FourCC (Four character code).

Examples

Creating a video with OpenCV (Java)

```
VideoWriter videoWriter;
videoWriter = new VideoWriter(outputFile, VideoWriter.fourcc('x', '2','6','4'),
                             fps, frameSize, isRGB);
//We have stated that we will use x264 as codec with FourCC
//For writing, we add the following method and it will write the image we give as parameter in
this call.
public void Write(Mat frame) {
    if(videoWriter.isOpened()==false){
        videoWriter.release();
        throw new IllegalArgumentException("Video Writer Exception: VideoWriter not
opened,"
                                     + "check parameters.");
    }
    //Write video
    videoWriter.write(frame);
}

//With Video Capture for example, we can read images from the camera and write the same video

VideoCapture videoCapture = new VideoCapture(0);
Size frameSize = new Size((int) videoCapture.get(Videoio.CAP_PROP_FRAME_WIDTH), (int)
videoCapture.get(Videoio.CAP_PROP_FRAME_HEIGHT));
VideoWriter videoWriter = new VideoWriter("test.avi", VideoWriter.fourcc('x', '2','6','4'),
        videoCapture.get(Videoio.CAP_PROP_FPS), frameSize, true);
while (videoCapture.read(mat)) {
    videoWriter.write(mat);
}
videoCapture.release();
videoWriter.release();
```

Read Creating a Video online: <http://www.riptutorial.com/opencv/topic/9196/creating-a-video>

Chapter 8: Display Image OpenCV

Examples

Display Image OpenCV Java

Basic reading image from java

```
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;

//Load native library
System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
//Mat object used to host the image
Mat imageArray;
//Read image file from file system
imageArray=Imgcodecs.imread("path/to/image");
```

If you want to view images you can not use imshow because OpenCV-java does not have this method either. Instead, you can write the following method.

```
private static BufferedImage ConvertMat2Image(Mat imgContainer{
    MatOfByte byteMatData = new MatOfByte();
    //image formatting
    Imgcodecs.imencode(".jpg", imgContainer,byteMatData);
    // Convert to array
    byte[] byteArray = byteMatData.toArray();
    BufferedImage img= null;
    try {
        InputStream in = new ByteArrayInputStream(byteArray);
        //load image
        img= ImageIO.read(in);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
    return img;
}
```

You can view the result object in the JFrame, JLabel (jlabel icon) etc.

Reading MJPEG from IP camera

```
import cv2
import numpy as np
import urllib

stream=urllib.urlopen('http://96.10.1.168/mjpg/video.mjpg')
bytes=''
while True:
    bytes+=stream.read(1024)
```

```

a = bytes.find('\xff\xd8') # JPEG start
b = bytes.find('\xff\xd9') # JPEG end
if a!=-1 and b!=-1:
    jpg = bytes[a:b+2] # actual image
    bytes= bytes[b+2:] # other informations

    # decode to colored image ( another option is cv2.IMREAD_GRAYSCALE )
    img = cv2.imdecode(np.fromstring(jpg, dtype=np.uint8),cv2.IMREAD_COLOR)
    cv2.imshow('Window name',img) # display image while receiving data
    if cv2.waitKey(1) ==27: # if user hit esc
        exit(0) # exit program

```

Every JPEG starts with 0xff 0xd8 and ends with 0xff 0xd9. Between those are actual image. Detailed information in [this SO answer](#)

Basic reading and display of an image

```

import cv2

image_path= #put your image path here

#use imread() function to read image data to variable img.
img = cv2.imread(image_path)

#display image data in a new window with title 'I am an image display window'
cv2.imshow('I am an image display window',img)

#wait until user hits any key on keyboard
cv2.waitKey(0)

#close any windows opened by opencv
cv2.destroyAllWindows()

```

To control the size of the display window on the screen, add the following commands before the cv2.imshow command:

```

window_width=800 #size of the display window on the screen
window_height=600

#open an empty window with a title.
#The flag cv2.WINDOW_NORMAL allows the window to be scaleable.
cv2.namedWindow('I am an image display window', cv2.WINDOW_NORMAL)

#scale the image display window to desired size
cv2.resizeWindow('I am an image display window', window_width, window_height)

```

see [openCV docs](#) for further details

Read Display Image OpenCV online: <http://www.riptutorial.com/opencv/topic/3306/display-image-opencv>

Chapter 9: Drawing Functions in Java

Examples

Draw rectangle on image

```
public class DrawRectangle {  
  
    public static void main(String[] args) {  
        //Load native library  
        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);  
        //image container object  
        Mat goruntuDizisi=new Mat();  
        //Read image in file system  
        goruntuDizisi=Imgcodecs.imread("C:\\image.jpg");  
        //Draw rectangle  
        //Parameters: mat object for drawing, point coordinates (x1,y1,x2,y2) and color BGR  
        Imgproc.rectangle(goruntuDizisi, new Point(10,100), new Point(100,200),new  
        Scalar(76,255,0));  
        Imgcodecs.imwrite("C:\\Yeni_kiz_kulesi.jpg", goruntuDizisi);  
        System.out.println("Writed");  
    }  
}
```

Read Drawing Functions in Java online: <http://www.riptutorial.com/opencv/topic/6153/drawing-functions-in-java>

Chapter 10: Drawing Shapes (Line, Circle, ..., etc) in C++

Introduction

In OpenCV, one can draw numerous shapes such as point, line, circle, ..., etc. There is an optional for filling a shape. The following code is self-explanatory which shows how shapes are drawn.

Examples

Drawing Shapes Sample

```
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc.hpp> // drawing shapes
#include <iostream>

int main( int argc, char** argv )
{
    // First create a black image.
    cv::Mat image(500,500, CV_8UC3, cv::Scalar(0,0,0));

    // Check if the image is created successfully.
    if( !image.data ){
        std::cout << "Could not open or find the image" << std::endl ;
        exit(EXIT_FAILURE);
    }

    //#####( Draw Line )#####
    cv::Point p1(100,100), p2(200,100);
    cv::Scalar colorLine(0,255,0); // Green
    int thicknessLine = 2;

    cv::line(image, p1, p2, colorLine, thicknessLine);

    //#####( Draw Circle )#####
    // unfilled circle
    cv::Point centerCircle1(250,250);
    int radiusCircle = 30;
    cv::Scalar colorCircle1(0,0,255);
    int thicknessCircle1 = 2;

    cv::circle(image, centerCircle1, radiusCircle, colorCircle1, thicknessCircle1);

    // filled circle
    cv::Point centerCircle2(400,100);
    cv::Scalar colorCircle2(0,100,0);

    cv::circle(image, centerCircle2, radiusCircle, colorCircle2, CV_FILLED);

    //#####( Draw Rectangle )#####
```

```

// unfilled
cv::Point p3(400,400), p4(450,450);
cv::Scalar colorRectangle1(0,0,255);
int thicknessRectangle1 = 3;

cv::rectangle(image, p3, p4, colorRectangle1,thicknessRectangle1);

// filled
cv::Point p5(100,400), p6(150,450);
cv::Scalar colorRectangle2(255,0,255);

cv::rectangle(image, p5, p6, colorRectangle2, CV_FILLED);

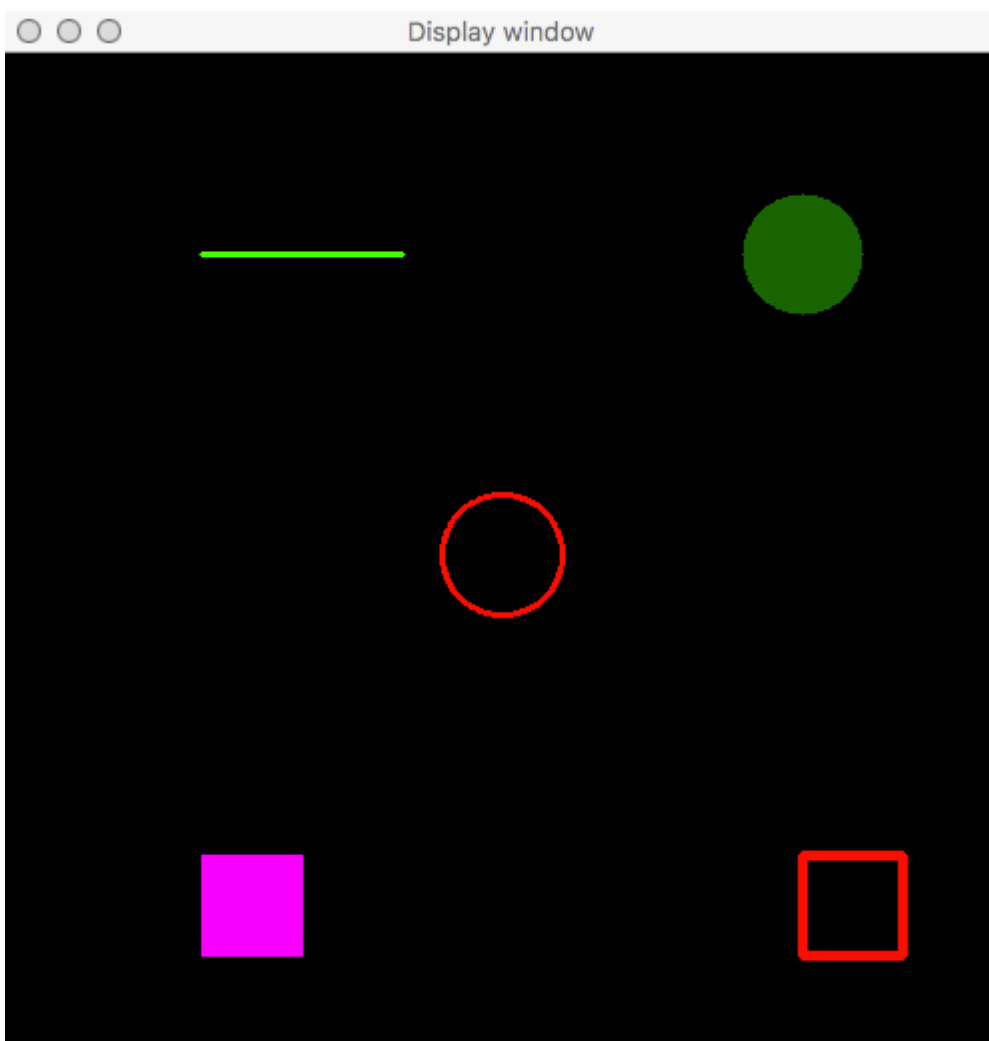
//#####( Draw Shapes on Image )#####
cv::namedWindow( "Display window", cv::WINDOW_AUTOSIZE );
cv::imshow( "Display window", image );

cv::waitKey(0);

return 0;
}

```

The output is



OpenCV 3.2 Mac with g++ compiler


```
g++ main2.cpp -o main `pkg-config --cflags --libs opencv`
```

Read Drawing Shapes (Line, Circle, ..., etc) in C++ online:

<http://www.riptutorial.com/opencv/topic/9749/drawing-shapes--line--circle-----etc--in-cplusplus>

Chapter 11: Edge detection

Syntax

- `edges = cv2.Canny(image, threshold1, threshold2[, edges[, apertureSize[, L2gradient]]])`
- `void Canny(InputArray image, OutputArray edges, double threshold1, double threshold2, int apertureSize=3, bool L2gradient=false)`

Parameters

Parameter	Details
image	Input image
edges	Output image
threshold1	First threshold for hysteresis procedure
threshold2	Second threshold for hysteresis procedure
apertureSize	Aperture size for Sobel operator
L2gradient	Flag indicating whether a more accurate algorithm for image gradient should be used

Examples

Canny algorithm

The Canny algorithm is a more recent edge detector designed as a signal processing problem. In OpenCV, it outputs a binary image marking the detected edges.

Python:

```
import cv2
import sys

# Load the image file
image = cv2.imread('image.png')

# Check if image was loaded improperly and exit if so
if image is None:
    sys.exit('Failed to load image')

# Detect edges in the image. The parameters control the thresholds
edges = cv2.Canny(image, 100, 2500, apertureSize=5)
```

```
# Display the output in a window
cv2.imshow('output', edges)
cv2.waitKey()
```

Canny Algorithm - C++

Below is an usage of canny algorithm in c++. Note that the image is first converted to grayscale image, then Gaussian filter is used to reduce the noise in the image. Then Canny algorithm is used for edge detection.

```
// CannyTutorial.cpp : Defines the entry point for the console application.
// Environment: Visual studio 2015, Windows 10
// Assumptions: Opecv is installed configured in the visual studio project
// Opencv version: OpenCV 3.1

#include "stdafx.h"
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<string>
#include<iostream>

int main()
{
    //Modified from source:
    https://github.com/MicrocontrollersAndMore/OpenCV_3_Windows_10_Installation_Tutorial
    cv::Mat imgOriginal;           // input image
    cv::Mat imgGrayscale;          // grayscale of input image
    cv::Mat imgBlurred;            // intermediate blurred image
    cv::Mat imgCanny;              // Canny edge image

    std::cout << "Please enter an image filename : ";
    std::string img_addr;
    std::cin >> img_addr;

    std::cout << "Searching for " + img_addr << std::endl;

    imgOriginal = cv::imread(img_addr);           // open image

    if (imgOriginal.empty()) {                     // if unable to open image
        std::cout << "error: image not read from file\n\n"; // show error message on
command line                                     // and exit program
        return(0);
    }

    cv::cvtColor(imgOriginal, imgGrayscale, CV_BGR2GRAY); // convert to grayscale

    cv::GaussianBlur(imgGrayscale,                // input image
imgBlurred,                                     // output image
cv::Size(5, 5),                                // smoothing window width and height in pixels
1.5);                                           // sigma value, determines how much the image
will be blurred

    cv::Canny(imgBlurred,                        // input image
imgCanny,                                       // output image
100,                                           // low threshold
```

```

        200);                                // high threshold

// Declare windows
// Note: you can use CV_WINDOW_NORMAL which allows resizing the window
// or CV_WINDOW_AUTOSIZE for a fixed size window matching the resolution of the image
// CV_WINDOW_AUTOSIZE is the default
cv::namedWindow("imgOriginal", CV_WINDOW_AUTOSIZE);
cv::namedWindow("imgCanny", CV_WINDOW_AUTOSIZE);

//Show windows
cv::imshow("imgOriginal", imgOriginal);
cv::imshow("imgCanny", imgCanny);

cv::waitKey(0);                            // hold windows open until user presses a key
return 0;
}

```

Calculating Canny Thresholds

Automatic calculation of low and high thresholds for the Canny operation in opencv

Canny Edge Thresholds prototyping using Trackbars

```

"""
CannyTrackbar function allows for a better understanding of
the mechanisms behind Canny Edge detection algorithm and rapid
prototyping. The example includes basic use case.

2 of the trackbars allow for tuning of the Canny function and
the other 2 help with understanding how basic filtering affects it.
"""
import cv2

def empty_function(*args):
    pass

def CannyTrackbar(img):
    win_name = "CannyTrackbars"

    cv2.namedWindow(win_name)
    cv2.resizeWindow(win_name, 500,100)

    cv2.createTrackbar("canny_th1", win_name, 0, 255, empty_function)
    cv2.createTrackbar("canny_th2", win_name, 0, 255, empty_function)
    cv2.createTrackbar("blur_size", win_name, 0, 255, empty_function)
    cv2.createTrackbar("blur_amp", win_name, 0, 255, empty_function)

    while True:
        cth1_pos = cv2.getTrackbarPos("canny_th1", win_name)
        cth2_pos = cv2.getTrackbarPos("canny_th2", win_name)
        bsize_pos = cv2.getTrackbarPos("blur_size", win_name)
        bamp_pos = cv2.getTrackbarPos("blur_amp", win_name)

        img_blurred = cv2.GaussianBlur(img.copy(), (trackbar_pos3 * 2 + 1, trackbar_pos3 * 2 +
1), bamp_pos)
        canny = cv2.Canny(img_blurred, cth1_pos, cth2_pos)
        cv2.imshow(win_name, canny)

```

```

        key = cv2.waitKey(1) & 0xFF
        if key == ord("c"):
            break

    cv2.destroyAllWindows()
    return canny

img = cv2.imread("image.jpg")
canny = CannyTrackbar(img)
cv2.imwrite("result.jpg", canny)

```

Canny Edge Video from Webcam Capture - Python

```

import cv2

def canny_webcam():
    "Live capture frames from webcam and show the canny edge image of the captured frames."

    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read() # ret gets a boolean value. True if reading is successful (I
        think). frame is an
        # uint8 numpy.ndarray

        frame = cv2.GaussianBlur(frame, (7, 7), 1.41)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        edge = cv2.Canny(frame, 25, 75)

        cv2.imshow('Canny Edge', edge)

        if cv2.waitKey(20) == ord('q'): # Introduce 20 milisecond delay. press q to exit.
            break

    canny_webcam()

```

Read Edge detection online: <http://www.riptutorial.com/opencv/topic/6099/edge-detection>

Chapter 12: Image Content Modification

Examples

Set Whole Image to a Solid Color

Given a non-empty `cv::Mat img` of some size, we can fill it to a solid color in several ways:

```
img = cv::Scalar(blueVal, greenVal, redVal);
```

or, the more general, mask supporting, `cv::Mat::setTo()`:

```
img.setTo(cv::Scalar(blueVal, greenVal, redVal));
```

If you are using the older OpenCV C API with `IplImage* img`:

Use:

```
cvSet(img, CV_RGB(redVal, greenVal, blueVal));
```

Pixel by pixel modification of images

In OpenCV, images can be RGB/BGR, HSV, grayscale, black-white and so on. It is crucial to know the data type before dealing with images.

The image data types are mainly `CV_8UC3` (Matrix of uchar with 3 channels) and `CV_8U` (Matrix of uchar with 1 channel), however, the conversion to other types such as `CV_32FC3`, `CV_64F` are also possible. (see [data types](#))

Consider, the image is an RGB image which is read by `imread` function.

```
Mat rgb = imread('path/to/rgb/image', CV_LOAD_IMAGE_COLOR);  
//to set RED pixel value of (i,j)th to X,  
rgb.at<Vec3b>(i, j)[0] = X;
```

Similarly, if the image is grayscale,

```
gray.at<uchar>(i, j) = X;
```

Note that, in OpenCV, Black&White images are stored as `CV_8U` type with the values 0 and 255. Therefore, changing BW images are same as gray images.

Image color modification in OpenCV - `kmeans()`. To scan all the pixels of an image and replace the pixel values with generic colors.

```
#include <opencv2/opencv.hpp> #include <vector> using namespace std; using namespace cv; int  
main() { Mat3b img = imread("test.jpg");
```

```

imshow("Original", img);

// Cluster

int K = 8;
int n = img.rows * img.cols;
Mat data = img.reshape(1, n);
data.convertTo(data, CV_32F);

Mat labels;
Mat1f colors;
kmeans(data, K, labels, cv::TermCriteria(), 1, cv::KMEANS_PP_CENTERS, colors);

for (int i = 0; i < n; ++i)
{
    data.at<float>(i, 0) = colors(labels.at<int>(i), 0);
    data.at<float>(i, 1) = colors(labels.at<int>(i), 1);
    data.at<float>(i, 2) = colors(labels.at<int>(i), 2);
}

Mat reduced = data.reshape(3, img.rows);
reduced.convertTo(reduced, CV_8U);

imshow("Reduced", reduced);
waitKey(0);

return 0;

}

```

Read Image Content Modification online: <http://www.riptutorial.com/opencv/topic/6307/image-content-modification>

Chapter 13: Image Processing

Syntax

1. **Gaussian Blur Syntax C++:** `void GaussianBlur(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0, int borderType=BORDER_DEFAULT)`

Parameters

Parameters of Gaussian Blur	Details
src	Input image, the image can have any number of channels, which are processed independently, but the depth should be <code>CV_8U</code> , <code>CV_16U</code> , <code>CV_16S</code> , <code>CV_32F</code> or <code>CV_64F</code> .
dst	Output image of the same size and type as <code>src</code>
ksize	Gaussian kernel size. <code>ksize.width</code> and <code>ksize.height</code> can differ but they both must be positive and odd . Or, they can be zero's and then they are computed from <code>sigma*</code> .
sigmaX	Gaussian kernel standard deviation in X direction .
sigmaY	Gaussian kernel standard deviation in Y direction . if <code>sigmaY</code> is zero, it is set to be equal to <code>sigmaX</code> , if both sigmas are zeros, they are computed from <code>ksize.width</code> and <code>ksize.height</code> . To fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of <code>ksize</code> , <code>sigmaX</code> , and <code>sigmaY</code> .
borderType	Pixel extrapolation method.

Remarks

I don't think it makes sense to put syntax and parameters specific to gaussian blur in this place as the topic is so broad that it should include many other examples

Examples

Smoothing Images with Gaussian Blur in C++

Smoothing, also known as **blurring**, is one of the most commonly used operation in Image Processing.

The most common use of the smoothing operation is to **reduce noise** in the image for further processing.

There are many algorithms to perform smoothing operation.

We'll look at one of the most commonly used filter for blurring an image, the **Gaussian Filter** using the OpenCV library function `GaussianBlur()`. This filter is designed specifically for removing *high-frequency noise* from images.

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

using namespace std;
using namespace cv;

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

    Mat image , blurredImage;

    // Load the image file
    image = imread(argv[1], CV_LOAD_IMAGE_COLOR);

    // Report error if image could not be loaded
    if(!image.data){
        cout<<"Error loading image" << "\n";
        return -1;
    }

    // Apply the Gaussian Blur filter.
    // The Size object determines the size of the filter (the "range" of the blur)
    GaussianBlur( image, blurredImage, Size( 9, 9 ), 1.0);

    // Show the blurred image in a named window
    imshow("Blurred Image" , blurredImage);

    // Wait indefinitely untill the user presses a key
    waitKey(0);

    return 0;
}
```

For the detailed mathematical definition and other types of filters you can check the [original documentation](#).

Thresholding

In Python:



```
import cv2
image_path= 'd:/contour.png'
img = cv2.imread(image_path)

#display image before thresholding
cv2.imshow('I am an image display window',img)
cv2.waitKey(0)

#convert image to gray scale - needed for thresholding
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

#apply threshold to gray image to obtain binary image

threshold=150 #value above which pixel values will be set to max_value
max_value=255 #value to which pixels above threshold will be set
threshold_stype=cv2.THRESH_BINARY #default threshold method

ret, img_binary = cv2.threshold(img_gray, threshold, max_value, threshold_stype)

#display image after thresholding
cv2.imshow('image after applying threshold',img_binary)
cv2.waitKey(0)

#save the binary image
cv2.imwrite('d:/binary.png',img_binary)
cv2.destroyAllWindows()
```



Bilateral Filtering

In image processing applications, the bilateral filters are a special type of **non-linear filters**.

There is a trade off between loosing structure and noise removal, because the most popular method to remove noise is Gaussian blurring which is not aware of structure of image; therefore, it also removes the edges. Most of the time, edges contain valuable information about the scene and we don't want to loose it. The **bilateral filter** is aware of structure of the scene and it tends to

act like a classical blurring filter when it is on a area without edges; however, when it sees an edge, it changes its behavior; so that, blurring does not work across the edges, but it works along the edges meaning that they are **edge-preserving filters**.

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

void main(int argc, char* argv[]) {
    if(argc==1) {
        std::cout << argv[0] << " <image>" << endl;
        return;
    }

    cv::Mat image, output;
    image = cv::imread(argv[1]);
    if(image.empty()) {
        std::cout << "Unable to load the image: " << argv[1] << endl;
        return;
    }

    cv::bilateralFilter(image, output, 3, 5, 3);
}
```

Read Image Processing online: <http://www.riptutorial.com/opencv/topic/2032/image-processing>

Chapter 14: Loading and Saving Various Media Formats

Examples

Loading Images

```
#include <highgui.h>

//...

cv::Mat img = cv::imread("img.jpg");
```

...

Loading Videos

Show how to use `cv::VideoCapture`. Here is the example of loading video from file:

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/core/core.hpp"
#include <iostream>

using namespace cv;
VideoCapture videoSource;
Mat frame;
#define VIDEO_PATH "video.avi"

int main()
{
    //Open video
    if (!videoSource.open(VIDEO_PATH))
    {
        std::cout<<"Video not found at "<<VIDEO_PATH<<std::endl;
        return 1;    // Exit if fail
    }
    videoSource.set(CV_CAP_PROP_CONVERT_RGB, 1);

    int cameraWidth = videoSource.get(CV_CAP_PROP_FRAME_WIDTH);
    int cameraHeight = videoSource.get(CV_CAP_PROP_FRAME_HEIGHT);
    float cameraAspectRatio = cameraWidth / cameraHeight;

    std::cout <<"Camera resolution: " << cameraWidth<<"<< cameraHeight<<" aspect ratio: "<<cameraAspectRatio<< std::endl;

    while(true)
    {
        videoSource >> frame;
        if(frame.empty())
            break;
        //Resize frame
```

```

        cv::resize(frame, frame, cv::Size(320, 320 / cameraAspectRatio));
        imshow("frame", frame);
        waitKey(20);
    }
    waitKey(0);
    return 0;
}

```

Live Capture

Show how to use `cv::VideoCapture` with e.g. a webcam. Capturing frames from webcam and display it. Here is the example code:

```

#include <iostream>

#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/core/core.hpp"

using namespace cv;
VideoCapture videoSource;
Mat frame;

int main()
{
    if(!videoSource.open(0)) //if more cameras available use 1,2,...
        return 1;

    while(true)
    {
        videoSource >> frame;
        if(frame.empty())
            break;
        imshow("Webcam", frame); //or any kind of precessing
        if(waitKey(1)==27)
            break;//stop capturing is ESC pressed
    }

    return 0;
}

```

Saving Videos

Show how to use `cv::VideoWriter`.

```

#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    VideoCapture cap(0); // open the video camera no. 0

    if (!cap.isOpened()) // if not success, exit program

```

```

{
    cout << "ERROR: Cannot open the video file" << endl;
    return -1;
}

namedWindow("MyVideo", CV_WINDOW_AUTOSIZE); //create a window called "MyVideo"

double dWidth = cap.get(CV_CAP_PROP_FRAME_WIDTH); //get the width of frames of the video
double dHeight = cap.get(CV_CAP_PROP_FRAME_HEIGHT); //get the height of frames of the
video

cout << "Frame Size = " << dWidth << "x" << dHeight << endl;

Size frameSize(static_cast<int>(dWidth), static_cast<int>(dHeight));

VideoWriter oVideoWriter ("D:/MyVideo.avi", CV_FOURCC('P','I','M','l'), 20, frameSize,
true); //initialize the VideoWriter object

if ( !oVideoWriter.isOpened() ) //if not initialize the VideoWriter successfully, exit the
program
{
    cout << "ERROR: Failed to write the video" << endl;
    return -1;
}

while (1)
{

    Mat frame;

    bool bSuccess = cap.read(frame); // read a new frame from video

    if (!bSuccess) //if not success, break loop
    {
        cout << "ERROR: Cannot read a frame from video file" << endl;
        break;
    }

    oVideoWriter.write(frame); //writer the frame into the file

    imshow("MyVideo", frame); //show the frame in "MyVideo" window

    if (waitKey(10) == 27) //wait for 'esc' key press for 30ms. If 'esc' key is pressed, break
loop
    {
        cout << "esc key is pressed by user" << endl;
        break;
    }
}

return 0;

}

```

Saving Images

Actually, Live Capture example is good for capturing images, so I am using it to capture images and save them in a folder.

```

#include <fstream>
#include <string>

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

int main()
{
    std::stringstream file; // to write the file name

    cv::VideoCapture cap(0); // create a capture object

    int counter = 0; // Create counter

    while(true) // infinite loop
    {
        cv::Mat frame; // Create a object

        cap.read(frame); // read the frame

        file << "/home/user/path_to_your_folder/image" << counter << ".jpg"; // file name

        cv::imwrite(file.str(), frame);

        counter++; // increment the counter
    }

    return 0;
}

```

Read Loading and Saving Various Media Formats online:

<http://www.riptutorial.com/opencv/topic/6658/loading-and-saving-various-media-formats>

Chapter 15: Object Detection

Examples

Template Matching with Java

Java Source Code

```
import org.opencv.core.Core;
import org.opencv.core.Core.MinMaxLocResult;
import org.opencv.core.Mat;
import org.opencv.core.Point;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;

public class TemplateMatching {

    public static void main(String[] args) {

        System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
        Mat source=null;
        Mat template=null;
        String filePath="C:\\Users\\mesutpiskin\\Desktop\\Object Detection\\Template
Matching\\Sample Image\\";
        //Load image file
        source=Imgcodecs.imread(filePath+"kapadokya.jpg");
        template=Imgcodecs.imread(filePath+"balon.jpg");

        Mat outputImage=new Mat();
        int machMethod=Imgproc.TM_CCOEFF;
        //Template matching method
        Imgproc.matchTemplate(source, template, outputImage, machMethod);

        MinMaxLocResult mmr = Core.minMaxLoc(outputImage);
        Point matchLoc=mmr.maxLoc;
        //Draw rectangle on result image
        Imgproc.rectangle(source, matchLoc, new Point(matchLoc.x + template.cols(),
            matchLoc.y + template.rows()), new Scalar(255, 255, 255));

        Imgcodecs.imwrite(filePath+"sonuc.jpg", source);
        System.out.println("Complated.");
    }

}
```

RESULT



Resource Image



Template



Result Im

Read Object Detection online: <http://www.riptutorial.com/opencv/topic/6735/object-detection>

Chapter 16: OpenCV initialization in Android

Examples

Async Initialization

Using async initialization is a recommended way for application development. It uses the [OpenCV Manager](#) to access OpenCV libraries externally installed in the target system.

Code snippet implementing the async initialization:

```
public class MainActivity extends Activity implements CvCameraViewListener2 {

    private BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch(status) {
                case LoaderCallbackInterface.SUCCESS:
                    Log.i(TAG, "OpenCV Manager Connected");
                    //from now onwards, you can use OpenCV API
                    Mat m = new Mat(5, 10, CvType.CV_8UC1, new Scalar(0));
                    break;
                case LoaderCallbackInterface.INIT_FAILED:
                    Log.i(TAG, "Init Failed");
                    break;
                case LoaderCallbackInterface.INSTALL_CANCELED:
                    Log.i(TAG, "Install Cancelled");
                    break;
                case LoaderCallbackInterface.INCOMPATIBLE_MANAGER_VERSION:
                    Log.i(TAG, "Incompatible Version");
                    break;
                case LoaderCallbackInterface.MARKET_ERROR:
                    Log.i(TAG, "Market Error");
                    break;
                default:
                    Log.i(TAG, "OpenCV Manager Install");
                    super.onManagerConnected(status);
                    break;
            }
        }
    };

    @Override
    public void onResume() {
        super.onResume();
        OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_1_0, this, mLoaderCallback);
    }

    ...
}
```

In this case, our application works with OpenCV Manager in asynchronous fashion. `OnManagerConnected` callback will be called in UI thread, when initialization finishes.

Please note, that it is not allowed to use OpenCV calls or load OpenCV-dependent native libs before invoking this callback. Load your own native libraries that depend on OpenCV after the successful OpenCV initialization.

Default `BaseLoaderCallback` implementation treat application context as `Activity` and calls `Activity.finish()` method to exit in case of initialization failure. To override this behaviour you need to override `finish()` method of `BaseLoaderCallback` class and implement your own finalization method.

OpenCV Manager

OpenCV Manager is an Android service targeted to manage OpenCV library binaries on end users devices. It allows sharing the OpenCV dynamic libraries between applications on the same device.

The Manager provides the following benefits:

- Less memory usage (around 40MB). All apps use the same binaries from service and do not keep native libs inside themselves.
- Hardware specific optimizations for all supported platforms.
- Trusted OpenCV library source. All packages with OpenCV are published on Google Play market.
- Regular updates and bug fixes.

The only disadvantage is that the user is prompted to download and extra app, so the user experience slightly decreases.

More info: [Android OpenCV Manager](#)

Updated 18/10/16:

There is a bug in the OpenCV Manager version distributed on [Play Store](#) (updated 21/09/15).

It affects only OpenCV 3.1.0 version. When you run some OpenCV functions you get a `SIGSEGV` error. The version distributed with Android SDK works fine (`OpenCV-android-sdk/apk/OpenCV_3.1.0_Manager_3.10_{platform}.apk`). It can be downloaded from [OpenCV website](#).

More info: [Issue #6247](#).

Static Initialization

According to this approach all OpenCV binaries are included into your application package. It is designed mostly for development and debugging purposes. This approach is **deprecated** for the production code, async initialization is recommended.

If your application project doesn't have a JNI part, just copy the corresponding OpenCV native libs from `OpenCV-3.1.0-android-sdk/sdk/native/libs` to your project directory to folder `app/src/main/jniLibs`.

In case of the application project with a JNI part, instead of manual libraries copying you need to

modify your `Android.mk` file: add the following two code lines after the `"include $(CLEAR_VARS)"` and before `"include path_to_OpenCV-3.1.0-android-sdk/sdk/native/jni/OpenCV.mk"`:

```
OPENCV_CAMERA_MODULES:=on
OPENCV_INSTALL_MODULES:=on
```

The result should look like the following:

```
include $(CLEAR_VARS)
# OpenCV
OPENCV_CAMERA_MODULES:=on
OPENCV_INSTALL_MODULES:=on
include ../../sdk/native/jni/OpenCV.mk
```

After that the OpenCV libraries will be copied to your application `jniLibs` folder during the JNI build.

The last step of enabling OpenCV in your application is Java initialization code before calling OpenCV API. It can be done, for example, in the static section of the Activity class:

```
static {
    if (!OpenCVLoader.initDebug()) {
        // Handle initialization error
    }
}
```

If you application includes other OpenCV-dependent native libraries you should load them after OpenCV initialization:

```
static {
    if (!OpenCVLoader.initDebug()) {
        // Handle initialization error
    } else {
        System.loadLibrary("my_jni_lib1");
        System.loadLibrary("my_jni_lib2");
    }
}
```

Note: `initDebug()` method is deprecated for production code. It is designed for experimental and local development purposes only. If you want to publish your app use approach with async initialization.

Read OpenCV initialization in Android online:

<http://www.riptutorial.com/opencv/topic/7545/opencv-initialization-in-android>

Chapter 17: OpenCV Installation

Introduction

OpenCV Installation On Linux, Mac OS and Windows

Examples

OpenCV Installation on Ubuntu

Source Link

Open the Terminal and write the following commands.

1-Update and upgrade package your system ubuntu:

```
sudo su  
  
sudo apt-get -y update  
  
sudo apt-get -y upgrade  
  
sudo apt-get -y dist-upgrade  
  
sudo apt-get -y autoremove
```

2-Installing Dependences:

```
sudo apt-get install libopencv-dev
```

3-Build Tools for OpenCV Source code:

```
sudo apt-get install build-essential checkinstall cmake pkg-config
```

4-Image I/O Libraries for OpenCV:

```
sudo apt-get install libtiff5-dev libjpeg-dev libjasper-dev libpng12-dev zlib1g-dev  
libopenexr-dev libgdal-dev
```

5-Video I/O Libraries for OpenCV:

```
sudo apt-get install libavcodec-dev libavformat-dev libmp3lame-dev libswscale-dev  
libdc1394-22-dev libxine2-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-  
dev v4l-utils libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-  
dev libxvidcore-dev libx264-dev x264 yasm
```

6-Parallelism and linear algebra libraries:

```
sudo apt-get install libtbb-dev libeigen3-dev
```

7-Graphic User Interface Libraries:

```
sudo apt-get install libqt4-dev libgtk2.0-dev qt5-default
```

```
sudo apt-get install libvtk6-dev
```

8-Java Installation:

```
sudo apt-get install ant default-jdk
```

9-Python Installation:

```
sudo apt-get install python-dev python-tk python-numpy python3-dev python3-tk python3-numpy  
python-matplotlib
```

```
sudo apt-get install python-opencv
```

```
sudo apt-get install doxygen
```

10-Download OPENCV source code from Github:

```
wget https://github.com/opencv/opencv/archive/3.2.0.zip
```

11-Decompress OPENCV Zip file:

```
unzip 3.2.0.zip
```

12-Remove OPENCV Zip file:

```
rm 3.2.0.zip
```

13-Build OPENCV:

```
mv opencv-3.2.0 opencv
```

```
cd opencv
```

```
mkdir build
```

```
cd build
```

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D  
BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D  
INSTALL_PYTHON_EXAMPLES=ON -D BUILD_DOC=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON  
-D WITH_EIGEN=ON -D FORCE_VTK=TRUE -D WITH_VTK=ON ..
```

```
make -j4
```

sudo make install

```
sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
```

14-Finished Check your OpenCV Version Number:

```
pkg-config --modversion opencv
pkg-config --cflags opencv
```

Read OpenCV Installation online: <http://www.riptutorial.com/opencv/topic/8934/opencv-installation>

Chapter 18: Pixel Access

Remarks

Be careful to be aware of the type of `cv::Mat` you are dealing with. For example, if you have a `cv::Mat` of type `CV_8UC3`, but access it with `image.at<uchar>(r,c)` no error will occur, but your program will have some unexpected behavior.

Examples

Setting and getting pixel values of a Gray image in C++

```
// PixelAccessTutorial.cpp : Defines the entry point for the console
// Environment: Visual studio 2015, Windows 10
// Assumptions: Opecv is installed configured in the visual studio project
// Opencv version: OpenCV 3.1

#include "stdafx.h"
#include<opencv2/core/core.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<string>
#include<iostream>

int main()
{
    cv::Mat imgOriginal;           // input image
    cv::Mat imgGrayscale;         // grayscale of input image

    std::cout << "Please enter an image filename : ";
    std::string img_addr;
    std::cin >> img_addr;

    std::cout << "Searching for " + img_addr << std::endl;

    imgOriginal = cv::imread(img_addr);           // open image

    if (imgOriginal.empty()) {                    // if unable to open
        std::cout << "error: image not read from file\n\n";           // show error message
        return(0);                               // and exit program
    }

    cv::cvtColor(imgOriginal, imgGrayscale, CV_BGR2GRAY);           // convert to grayscale

    const int channels = imgGrayscale.channels();
    printf("Number of channels = %d", channels);

    cv::Mat output ;
    imgGrayscale.copyTo(output); // Just to make sure the Mat objects are of the same size.

    //Set the threshold to your desired value
```



```

uchar threshold = 127;

if (channels == 1)
{
    for (int x = 0; x<imgGrayscale.rows; x++) {
        for (int y = 0; y<imgGrayscale.cols; y++) {
            // Accesssing values of each pixel
            if (imgGrayscale.at<uchar>(x, y) >= threshold) {
                // Setting the pixel values to 255 if it's above the threshold
                output.at<uchar>(x, y) = 254;
            }
            else if (imgGrayscale.at<uchar>(x, y) < threshold) {
                // Setting the pixel values to 255 if it's below the threshold
                output.at<uchar>(x, y) = 0;
            }
            else {
                // Just in case
                printf("The value at (%d, %d) are not right. Value: %d\n", x, y,
imgGrayscale.at<uchar>(x, y));
            }
        }
    }
}
else if (channels == 3)
{
    // This is only for gray scale images
    printf("\tThe image has 3 channels. The function does not support images with 3
channels.\n");
}

//Create windows to show image
cv::namedWindow("Gray scale", CV_WINDOW_AUTOSIZE);
cv::namedWindow("Binary", CV_WINDOW_AUTOSIZE);

cv::imshow("Gray scale", imgGrayscale);
cv::imshow("Binary", output);

cv::waitKey(0); // hold windows open until user presses a key

return 0;
}

```

Alternative pixel access with Matiterator

It's not the best way of iterating through the pixels; however, it's better than `cv::Mat::at<T>`.

Let's assume you have a color image in your folder and you want to iterate each pixels of this image and erase green and red channels(Note that this is an example, you can do this in more optimized ways);

```

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

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

    // Create a container

```

```

cv::Mat im;

//Create a vector
cv::Vec3b *vec;

// Create an mat iterator
cv::MatIterator_<cv::Vec3b> it;

// Read the image in color format
im = cv::imread("orig1.jpg", 1);

// iterate through each pixel
for(it = im.begin<cv::Vec3b>(); it != im.end<cv::Vec3b>(); ++it)
{
    // Erase the green and red channels
    (*it)[1] = 0;
    (*it)[2] = 0;
}

// Create a new window
cv::namedWindow("Resulting Image");

// Show the image
cv::imshow("Resulting Image", im);

// Wait for a key
cv::waitKey(0);

return 0;
}

```

To compile this with Cmake:

```

cmake_minimum_required(VERSION 2.8)
project(Main)
find_package(OpenCV REQUIRED)
add_executable(Main main.cpp)
target_link_libraries(Main ${OpenCV_LIBS})

```

The original image:



The processed image:



Note that we don't touch only the Blue channel.

For more information: http://docs.opencv.org/2.4/opencv_tutorials.pdf Page:145

Efficient pixel access using `cv::Mat::ptr` pointer

If efficiency is important, a fast way to iterate over pixels in a `cv::Mat` object is to use its `ptr<T>(int r)` method to obtain a pointer to the beginning of row `r` (0-based index).

According to the matrix type, the pointer will have a different template.

- For `CV_8UC1`: `uchar* ptr = image.ptr<uchar>(r);`
- For `CV_8UC3`: `cv::Vec3b* ptr = image.ptr<cv::Vec3b>(r);`
- For `CV_32FC1`: `float* ptr = image.ptr<float>(r);`
- For `CV_32FC3`: `cv::Vec3f* ptr = image.ptr<cv::Vec3f>(r);`

This `ptr` object can then be used to access the pixel value on row `r` and column `c` by calling `ptr[c]`.

To illustrate this, here is an example where we load an image from disk and invert its Blue and Red channels, operating pixel by pixel:

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

```

int main(int argc, char** argv) {
    cv::Mat image = cv::imread("image.jpg", CV_LOAD_IMAGE_COLOR);

    if(!image.data) {
        std::cout << "Error: the image wasn't correctly loaded." << std::endl;
        return -1;
    }

    // We iterate over all pixels of the image
    for(int r = 0; r < image.rows; r++) {
        // We obtain a pointer to the beginning of row r
        cv::Vec3b* ptr = image.ptr<cv::Vec3b>(r);

        for(int c = 0; c < image.cols; c++) {
            // We invert the blue and red values of the pixel
            ptr[c] = cv::Vec3b(ptr[c][2], ptr[c][1], ptr[c][0]);
        }
    }

    cv::imshow("Inverted Image", image);
    cv::waitKey();

    return 0;
}

```

Access individual pixel values with `cv::Mat::at()`

To access pixel values in an OpenCV `cv::Mat` object, you first have to know the *type* of your matrix.

The most common types are:

- `CV_8UC1` for 8-bit 1-channel grayscale images;
- `CV_32FC1` for 32-bit floating point 1-channel grayscale images;
- `CV_8UC3` for 8-bit 3-channel color images; and
- `CV_32FC3` for 32-bit floating point 3-channel color images.

The default setting with `cv::imread` will create a `CV_8UC3` matrix.

To access individual pixels, the safest way, though not the most efficient, is to use `cv::Mat::at<T>(r,c)` method where *r* is the *row* of the matrix and *c* is the *column*. The template argument depends on the type of the matrix.

Let us say you have a `cv::Mat image`. According to its type, the access method and the pixel color type will be different.

- For `CV_8UC1`: `uchar pixelGrayValue = image.at<uchar>(r,c).`
- For `CV_8UC3`: `cv::Vec3b pixelColor = image.at<cv::Vec3b>(r,c).` The `cv::Vec3b` object represents a triplet of `uchar` values (integers between 0 and 255).
- For `CV_32FC1`: `float pixelGrayValue = image.at<float>(r,c).`
- For `CV_32FC3`: `cv::Vec3f pixelColor = image.at<cv::Vec3f>(r,c).` The `cv::Vec3f` object represents a triplet of `float` values.

Note that OpenCV represents images in **row-major** order, like, e.g. Matlab or as the convention in Algebra. Thus, if your pixel coordinates are (x, y) , then you will access the pixel using

```
image.at<.>(y, x).
```

Alternatively, `at<>` also support access via a single `cv::Point` argument.

In this case, the access is done in *column-major*.

```
image.at<.>(cv::Point(x, y));
```

Take a look at [OpenCV documentation](#) for more details on this method.

Pixel Access in Mat

Individual pixel access in OpenCV Mat structure can be achieved in multiple ways. To understand how to access, it is better to learn the data types first.

[Basic Structures](#) explains the basic datatypes. Shortly, `CV_<bit-depth>{U|S|F}C(<number_of_channels>)` is the basic structure of a type. Along with that, it is important to understand `Vec` structures.

```
typedef Vec<type, channels> Vec< channels>< one char for the type>
```

where `type` is one of `uchar`, `short`, `int`, `float`, `double` and the characters for each type are `b`, `s`, `i`, `f`, `d`, respectively.

For example, `Vec2b` indicates an unsigned char vector of 2 channels.

Consider `Mat mat(R, C, T)` where `R` is #rows, `C` is #cols and `T` is type. Some examples for accessing the (i, j) coordinate of `mat` are:

2D:

```
If the type is CV_8U or CV_8UC1 ---- //they are alias
mat.at<uchar>(i, j) // --> This will give char value of index (i, j)
//If you want to obtain int value of it
(int)mat.at<uchar>(i, j)
```

```
If the type is CV_32F or CV_32FC1 ---- //they are alias
mat.at<float>(i, j) // --> This will give float value of index (i, j)
```

3D:

```
If the type is CV_8UC2 or CV_8UC3 or more channels
mat.at<Vec2b/Vec3b>(i, j)[k] // note that (k < #channels)
//If you want to obtain int value of it
(int)mat.at<uchar>(i, j)[k]
```

```
If the type is CV_64FC2 or CV_64FC3
mat.at<Vec2d/Vec3d>(i, j)[k] // note that k < #channels
```

Note that, it is very crucial to enter correct type in `<...>`, otherwise, you can have runtime error or

unwanted results.

Read Pixel Access online: <http://www.riptutorial.com/opencv/topic/1957/pixel-access>

Chapter 19: Using Cascade Classifiers In Java

Syntax

- `CascadeClassifier cascade = new CascadeClassifier("cascade.xml");` // Creates a cascade classifier from cascade.xml
- `Mat image = Imgcodecs.imread("image.png");` // Converts image.png into a Mat (Matrix) object
- `MatOfRect detections = new MatOfRect();` // Creates an empty MatOfRect (Matrix of Rectangles) file, used as output for our detection classes
- `detections.toArray();` // Returns an array of Rect objects that can be iterated over
- `Imgproc.rectangle(image, new Point(rect.x, rect.y), new Point(rect.x + rect.width, rect.y + rect.height), new Scalar(0, 255, 0));` // Draws an green outlined rectangle from the first Point's x and y locations to the second Point's x and y location onto the Mat object "image". "rect" is a Rect object, usually provided by detections.toArray(). Uses OpenCV's Point class.
- `Imgcodecs.imwrite("output.png", image);` // Writes the modified Mat object "image" to the "output.png"
- `CascadeClassifier.detectMultiScale(image, detections);` // Detects any object in the Mat object "image" and outputs the detections in the MatOfRect object "detections"
- `CascadeClassifier.detectMultiScale(image, detections, scaleFactor, minNeighbors, flags, minSize, maxSize);` // Performs a detection with additional parameters. See details below.
- `Imgproc.ellipse(image, center, axes, 0, 0, 360, new Scalar(255, 0, 255), thickness, lineType, 0);` // Draws an ellipse onto the image at the point `center`. Uses OpenCV's Point class.

Parameters

Parameter	Details
scaleFactor	How much the image size is reduced at each image scale. Default = 1.1
minNeighbors	How many neighbors a candidate rectangle should have before selecting it as an detected object. Default = 4
flags	Legacy flags. In most cases, this should be set to 0. Default = 0
minSize	Minimum size a candidate rectangle can be. This uses OpenCV's <code>Size</code> class. Can be used to decrease detection time and CPU usage, as well as to reduce false positives.
maxSize	Maximum size a candidate rectangle can be. This uses OpenCV's <code>Size</code> class. Can be used to decrease detection time and CPU usage, as well as to reduce false positives.

Parameter	Details
axes	Uses OpenCV's Size class. Defines the width and height of the ellipse.
thickness	Thickness of the line, in pixels.
lineType	Has various parameters. 0 is the solid line, 8 is for a 8-connected line, 4 is for a 4-connected line, and CV_AA is for an antialiased line. Default = 8

Examples

Getting a static image, detecting items on it, and outputting the results.

Please note that this example uses OpenCV 3.1.

```
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;

public class Classifier {
    private CascadeClassifier diceCascade = new
        CascadeClassifier("res/newMethod/diceCascade.xml");
    private Mat image;
    private String loc = "path/to/image.png";
    private String output = "path/to/output.png";

    public void detImg() {

        Mat image = Imgcodecs.imread(loc); // Reads the image

        MatOfRect diceDetections = new MatOfRect(); // Output container
        diceCascade.detectMultiScale(image, diceDetections); // Performs the detection

        // Draw a bounding box around each detection.
        for (Rect rect : diceDetections.toArray()) {
            Imgproc.rectangle(image, new Point(rect.x, rect.y),
                new Point(rect.x + rect.width, rect.y + rect.height),
                new Scalar(0, 255, 0));
        }

        // Save the visualized detection.
        Imgcodecs.imwrite(output, image);
    }
}
```

The `Rect[]` returned by `diceDetections.toArray()` can be iterated over. Each `Rect` inside the array will have four main properties: `x`, `y`, `width`, and `height`. `x` and `y` defines the rectangle's top-left position, and `width` and `height` returns an `int` of the width and height of the rectangle. This is used when drawing rectangles onto images. The `Imgproc.rectangle` function's minimal required

parameters are as follows:

```
Imgproc.rectangle(Mat image, Point start, Point end, Scalar color);
```

Both `Point` are used for the positions of the top-left corner and the lower-right corner. These positions are *both absolute* to the image provided as the first parameter, not to each other. Thus, you must add both the rectangle's `x` or `y` position in addition to the `width` or `height` to properly define the `end Point`.

Note that the `Point` class used in these parameters are **not** Java's standard library's `Point` class. You must import OpenCV's `Point` class instead!

Detecting images from a video device

This example introduces the `VideoCapture` class, where we use it to take an image from a webcam and save it to an image.

```
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;
import org.opencv.videoio.VideoCapture;

public class Classifier {
    private CascadeClassifier diceCascade = new
        CascadeClassifier("res/newMethod/diceCascade.xml");
    private Mat image;
    private String loc = "path/to/image.png";
    private String output = "path/to/output.png";
    private VideoCapture vc = new VideoCapture();

    public void detImg() {
        vc.open(0); // Opens the video stream

        Mat image = new Mat(); // Creates an empty matrix
        vc.read(image); // Reads the image from the video stream and
            writes it to the image matrix.

        MatOfRect diceDetections = new MatOfRect(); // Output container
        diceCascade.detectMultiScale(image, diceDetections); // Performs the detection

        // Draw a bounding box around each detection.
        for (Rect rect : diceDetections.toArray()) {
            Imgproc.rectangle(image, new Point(rect.x, rect.y),
                new Point(rect.x + rect.width, rect.y + rect.height),
                new Scalar(0, 255, 0));
        }

        // Save the visualized detection.
        Imgcodecs.imwrite(output, image);

        vc.release(); // Closes the stream.
    }
}
```

```
}  
}
```

Converting an Mat object to an BufferedImage object

This example by Daniel Baggio was taken directly from [this StackExchange answer](#), but has been reposted for visibility.

This class takes an Mat object and returns the BufferedImage object used by the `javax.swing` libraries. This can be used by a `Graphics` object to draw the image.

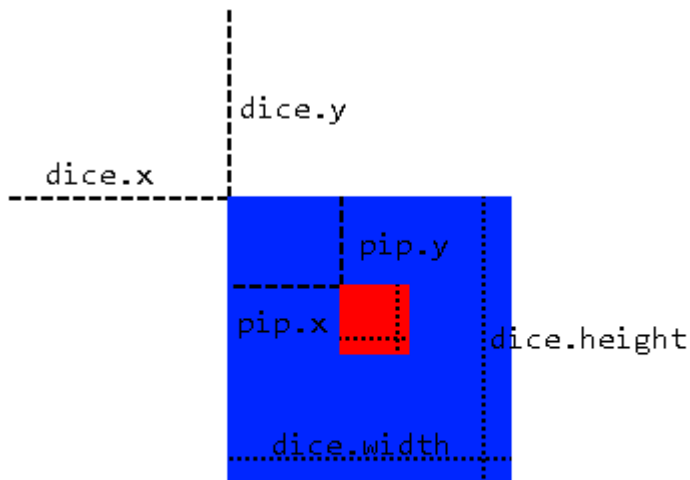
```
private BufferedImage toBufferedImage(Mat m) {  
    if (!m.empty()) {  
        int type = BufferedImage.TYPE_BYTE_GRAY;  
        if (m.channels() > 1) {  
            type = BufferedImage.TYPE_3BYTE_BGR;  
        }  
        int bufferSize = m.channels() * m.cols() * m.rows();  
        byte[] b = new byte[bufferSize];  
        m.get(0, 0, b); // get all the pixels  
        BufferedImage image = new BufferedImage(m.cols(), m.rows(), type);  
        final byte[] targetPixels = ((DataBufferByte)  
image.getRaster().getDataBuffer()).getData();  
        System.arraycopy(b, 0, targetPixels, 0, b.length);  
        return image;  
    }  
  
    return null;  
}
```

Detections within Detections

This example uses Dice and the black spots on the dice (the pips) as our object. As the example is quite long, first explaining some key concepts is critical to understanding the example.

Understanding the first example, "Getting a static image, detecting items on it, and outputting the results." is critical to understanding this example, especially how OpenCV draws rectangles.

Take a look at the following image:



We will be using the subimaging method, where we use a detected area as our base for applying more detections. This is only possible if an object will always be within another object that we can detect, such as our pips on our dice. This method has several benefits:

- Instead of scanning the entire image, we only need to scan the area where we know the object will be in.
- Removes any chance of false positives outside the detection area.

We do this by first applying one cascade classifier scan over the entire image to give us an `MatOfRect` object containing our large objects (dice, in this case). We then iterate over the `Rect[]` array given by the `toArray()` function from the `MatOfRect` object. This `Rect` object is used in creating a temporary `Mat` object that is "cropped" to the `Rect` object's properties (`x`, `y`, `width`, `height`) from the original image, where we can then perform detections on the temporary `Mat` object. In other words, we tell the classifier to only perform detections on the dice parts of the image instead, and we specify the position of each dice by using the `Rect` objects that we got from performing a detection on the entire image.

However, the `Rect` objects (pips) have their properties relative to their dice, and not the image itself. To solve this issue, when we want to draw rectangles to the actual image showing the pips' locations, we add both `dice.x` and `dice.y` to the start `Point`.

```
import org.opencv.core.Mat;
import org.opencv.core.MatOfRect;
import org.opencv.core.Point;
import org.opencv.core.Rect;
import org.opencv.core.Scalar;
import org.opencv.core.Size;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import org.opencv.objdetect.CascadeClassifier;
import org.opencv.videoio.VideoCapture;

public class Classifier {

    private CascadeClassifier diceCascade =
        new CascadeClassifier("res/newMethod/diceCascade.xml");
```

```

private CascadeClassifier pipCascade =
    new CascadeClassifier("res/newMethod/pipCascade6.xml");
private VideoCapture vc = new VideoCapture();
private Mat image;

public void openVC(int index) {
    vc.open(index);
}

public void closeVC() {
    vc.close();
}

public Mat getNextImage() {
    image = new Mat();
    vc.read(image); // Sets the matrix to the current livestream frame.

    MatOfRect diceDetections = new MatOfRect(); // Output container

    // See syntax for explanations on addition parameters
    diceCascade.detectMultiScale(image, diceDetections, 1.1, 4, 0, new Size(20, 20),
        new Size(38, 38));

    // Iterates for every Dice ROI
    for (int i = 0; i < diceDetections.toArray().length; i++) {
        Rect diceRect = diceDetections.toArray()[i];

        // Draws rectangles around our detected ROI
        Point startingPoint = new Point(diceRect.x, diceRect.y);
        Point endingPoint = new Point(diceRect.x + diceRect.width,
            diceRect.y + diceRect.height);
        Imgproc.rectangle(image, startingPoint, endingPoint, new Scalar(255, 255, 0));

        MatOfRect pipDetections = new MatOfRect();

        pipCascade.detectMultiScale(image.submat(diceRect), pipDetections, 1.01, 4, 0,
            new Size(2, 2), new Size(10, 10));

        // Gets the number of detected pips and draws a cricle around the ROI
        for (int y = 0; y < pipDetections.toArray().length; y++) {
            // Provides the relative position of the pips to the dice ROI
            Rect pipRect = pipDetections.toArray()[y];

            // See syntax explanation
            // Draws a circle around our pips
            Point center = new Point(diceRect.x + pipRect.x + pipRect.width / 2,
                diceRect.y + pipRect.y + pipRect.height / 2);
            Imgproc.ellipse(image, center, new Size(pipRect.width / 2, pipRect.height /
2),
                0, 0, 360, new Scalar(255, 0, 255), 1, 0, 0);
        }
    }

    return image;
}
}

```

The `getNextImage()` function returns a `Mat` object, which with in conjunction with the other examples posted, can be called constantly and can be converted to a `BufferImage`, to provide a livestream displaying detections.

Read Using Cascade Classifiers In Java online:

<http://www.riptutorial.com/opencv/topic/6377/using-cascade-classifiers-in-java>

Chapter 20: Using VideoCapture With OpenCV Python

Examples

Reading frames from a pre-captured video



```
import numpy as np
import cv2

#access a video from your disk
#to use the GIF in this example, convert to avi!
cap = cv2.VideoCapture('eg_videoRead.avi')

#we are going to read 10 frames
#we store the frames in a numpy structure
#then we'll generate a minimum projection of those frames

frameStack=[]
numFrames=10

for fr in range(numFrames):
    cap.set(cv2.CAP_PROP_POS_FRAMES,fr) #specifies which frame to read next
    frame=cap.read() #read the frame
    #gray = cv2.cvtColor(frame[1], cv2.COLOR_BGR2GRAY) #convert to gray scale
    frameStack.append(frame[1]) #add current frame to our frame Stack

minProjection=np.min(frameStack,axis=0) #find the minimum across frames
cv2.imshow("projection", minProjection) #show the result
```

Using VideoCapture With OpenCV Java

There is no imshow in the java, you need to write a method for this. This method is a `Mat2bufferedImage`. Takes mat object as parameter and returns image.

```
public static void main(String[] args) {
    Mat frame = new Mat();
    //0; default video device id
    VideoCapture camera = new VideoCapture(0);
    JFrame jframe = new JFrame("Title");
    jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
JLabel vidpanel = new JLabel();
jframe.setContentPane(vidpanel);
jframe.setVisible(true);

while (true) {
    if (camera.read(frame)) {

        ImageIcon image = new ImageIcon(Mat2bufferedImage(frame));
        vidpanel.setIcon(image);
        vidpanel.repaint();

    }
}
}
```

Read Using VideoCapture With OpenCV Python online:

<http://www.riptutorial.com/opencv/topic/6803/using-videocapture-with-opencv-python>

Credits

S. No	Chapters	Contributors
1	Getting started with opencv	Arijit , bburns.km , Berriel , Community , Elizabeth , hackhisass , jlarsch , John Hany , K D , MD. Nazmul Kibria , mesutpiskin , snb , StephenG , Sunreef , winseybash , Yassie , zeeshan khan
2	Basic Structures	smttsp
3	Blob Detection	MD. Nazmul Kibria , Sebastian
4	Build and Compile opencv 3.1.0-dev for Python2 on Windows using CMake and Visual Studio	Tes3awy
5	Cascade Classifiers	Arijit , MD. Nazmul Kibria , mesutpiskin
6	Contrast and Brightness in C++	Ehsan Ab , MD. Nazmul Kibria
7	Creating a Video	mesutpiskin
8	Display Image OpenCV	Aleksandar , Elizabeth , jlarsch , mesutpiskin , smttsp
9	Drawing Functions in Java	mesutpiskin
10	Drawing Shapes (Line, Circle, ..., etc) in C++	CroCo
11	Edge detection	cmastudios , Ehsan Ab , K D , m3h0w , Sounak
12	Image Content Modification	Adi Shavit , DivyaMaheswaran , smttsp
13	Image Processing	cagatayodabasi , Dan Mašek , Elizabeth , jlarsch , Shubham Batra , Sunreef , Utkarsh Sinha
14	Loading and Saving Various Media Formats	Adi Shavit , cagatayodabasi , Jav_Rock , Lakshya Kejriwal , MD. Nazmul Kibria

15	Object Detection	K D , mesutpiskin
16	OpenCV initialization in Android	David Miguel
17	OpenCV Installation	amorenew
18	Pixel Access	Adi Shavit , brian , cagatayodabasi , Ehsan Ab , Elizabeth , smttsp , Sunreef
19	Using Cascade Classifiers In Java	Edward Shen
20	Using VideoCapture With OpenCV Python	jlarsch , mesutpiskin