Module 0.3.1

Getting started Windows (using cmd)

Satya Mallick, Ph.D.

June 19, 2017

Table of Contents

Installing OpenCV on Windows **3**

Step 1: Install Visual Studio 3

Step 2: Install CMake 5

Step 3: Install Anaconda (a python distribution) 7

Step 4: Download and extract opencv-3.2.0 and opencv_contrib-3.2.0 9

Step 5: Generate Visual Studio project using CMAKE 11

Step 5.1 : Additional changes to CMake config 14

Step 5.2 : Add Python paths for both Python2 and Python3 (optional) 15

Step 6: Compile OpenCV 17

Step 6.1 : Compile opencv in Release mode 17

Step 6.2 : Compile opencv in Debug mode 18

Step 7: Update System Environment Variables 18

Step 7.1 : Update environment variable - PATH 18

Step 7.2 : Update user environment variable - OPENCV_DIR 21

Step 8: Testing C++ code 23

Step 9: Testing Python code 28

Step 9.1 : Quick check 28

Step 9.2 : Testing redEyeRemover application 28

Installing Dlib on Windows **30**

Step 1: Install CMake 30

Step 2: Download Dlib 30

Step 3: Build and Install Dlib library 30

Step 4: Update user environment variable - dlib_DIR 31

Step 5: Build Dlib examples 32

Step 6: Test Dlib's C++ example 33

Step 7: Install Dlib's Python module (only for Anaconda 3) 33

Step 8: Test Dlib's Python example 33

Troubleshooting Dlib's C++ read/write image 33

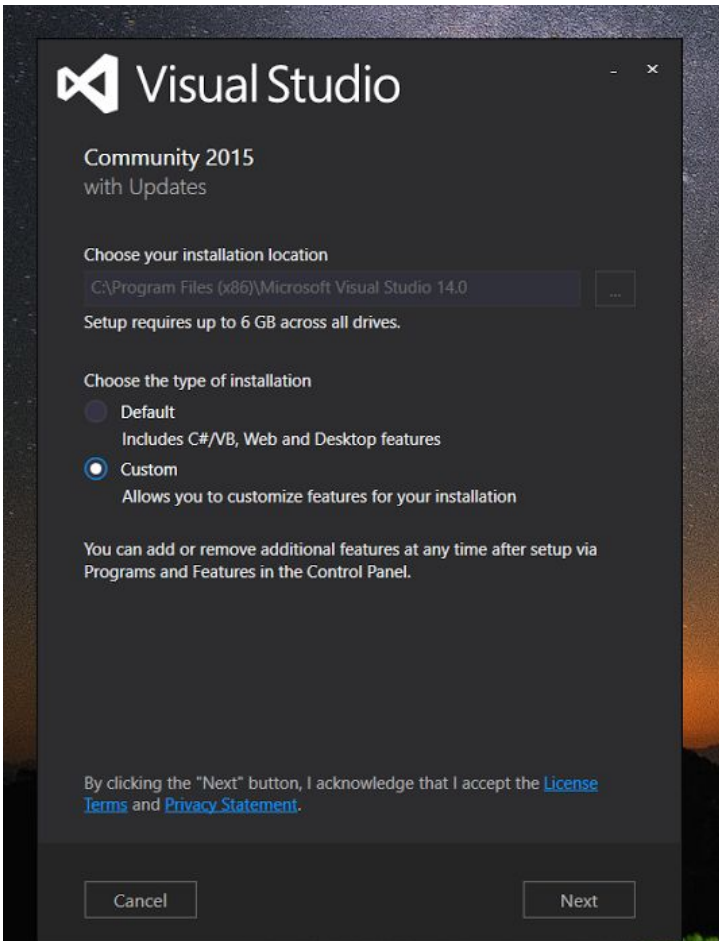
Installing OpenCV on Windows

In this tutorial, we will learn how to use CMAKE to build libraries and create projects for Visual Studios. If you would like to use Visual Studios GUI instead, please see Module 0.3.2.

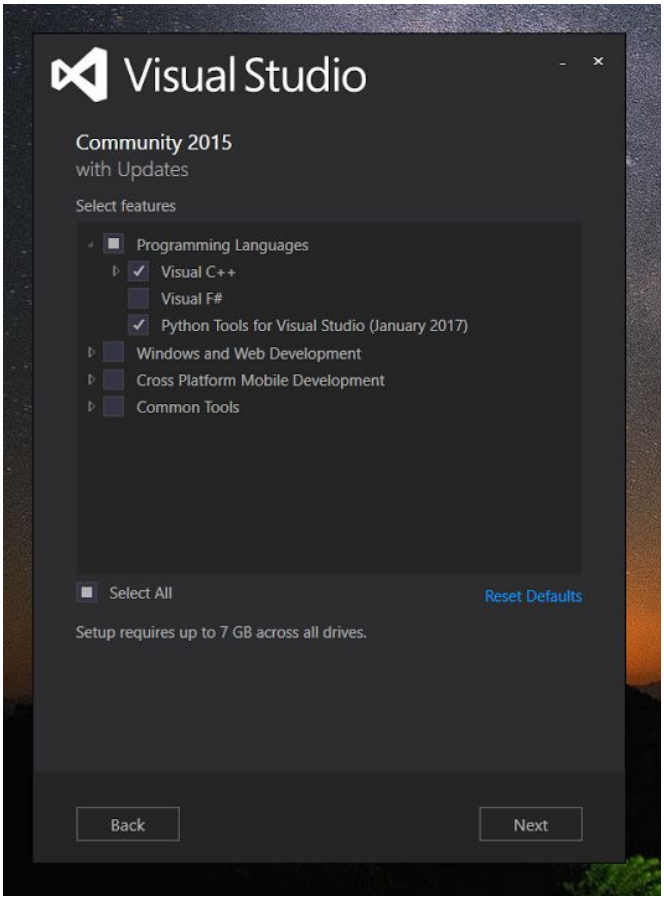
We have used Windows Power Shell to run commands. Alternatively, you can use the command prompt too.

Step 1: Install Visual Studio

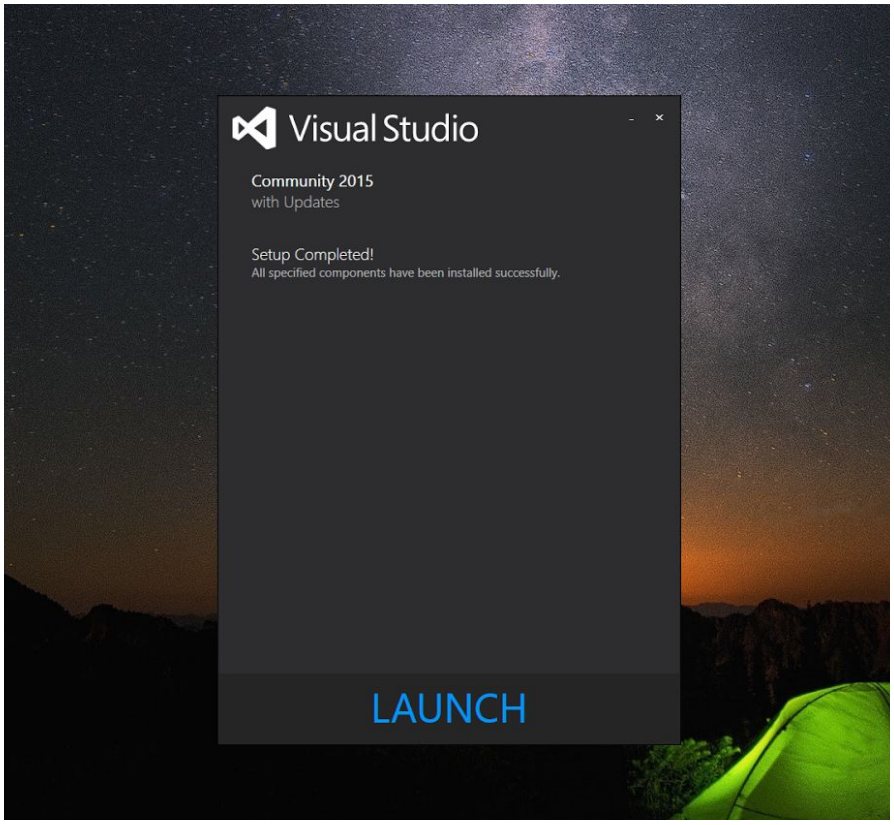
Download Visual Studio 2015 community edition from https://www.visualstudio.com/vs/older-downloads/. If you are finding it difficult to search for Visual Studio 2015, use this link. If you don't have a Visual Studio Dev Essentials account, create account and login. Run installer, select "Custom" in "type of installation".



In next screen within Programming Languages, select Visual C++ and Python tools for Visual Studio. Click next.



Now click next. It will take some time to complete installation.

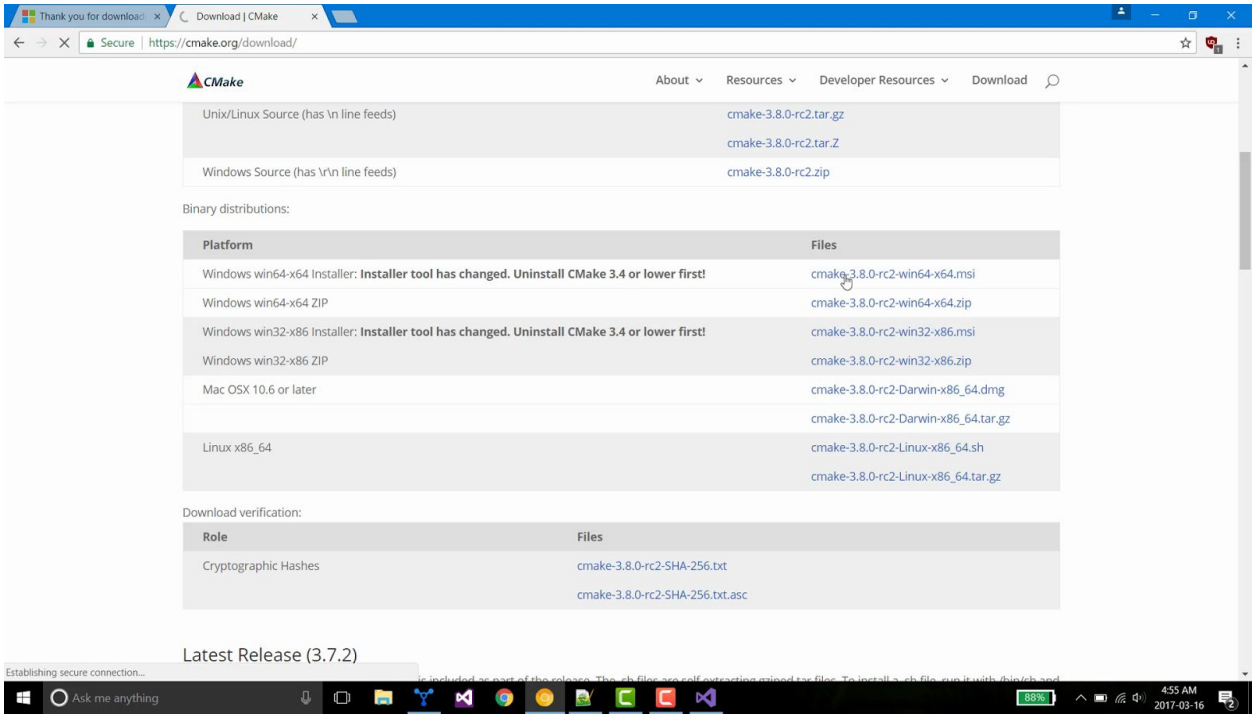


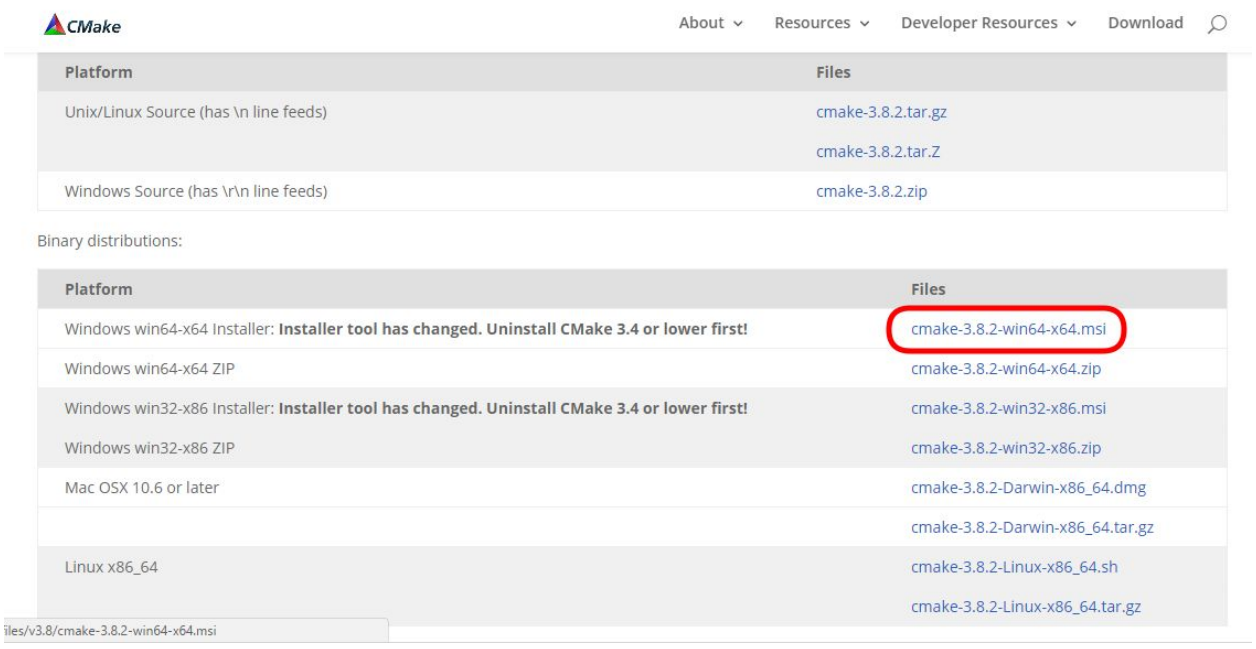
We have finished installation of Visual Studio 2015.

Note: Since Visual Studio 2017 fails to compile Dlib, we switched back to Visual Studio 2015.

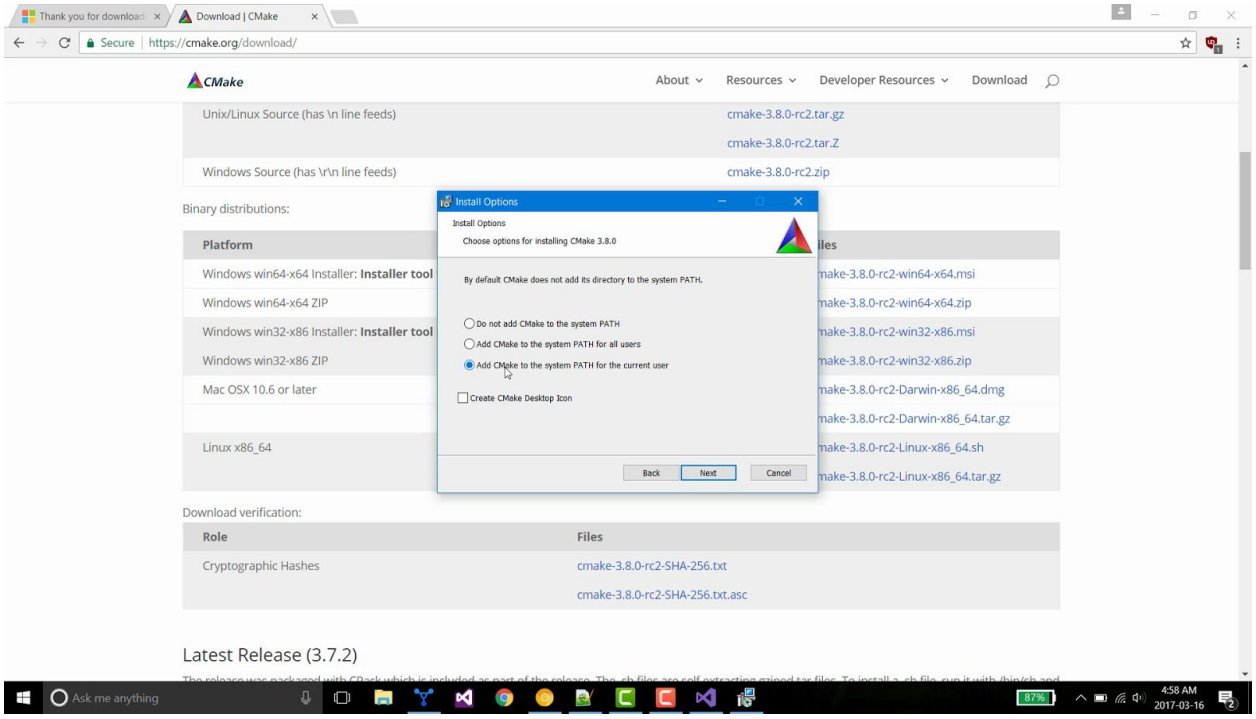
Step 2: Install CMake

Download and install CMake 3.8.2 from https://cmake.org/download/



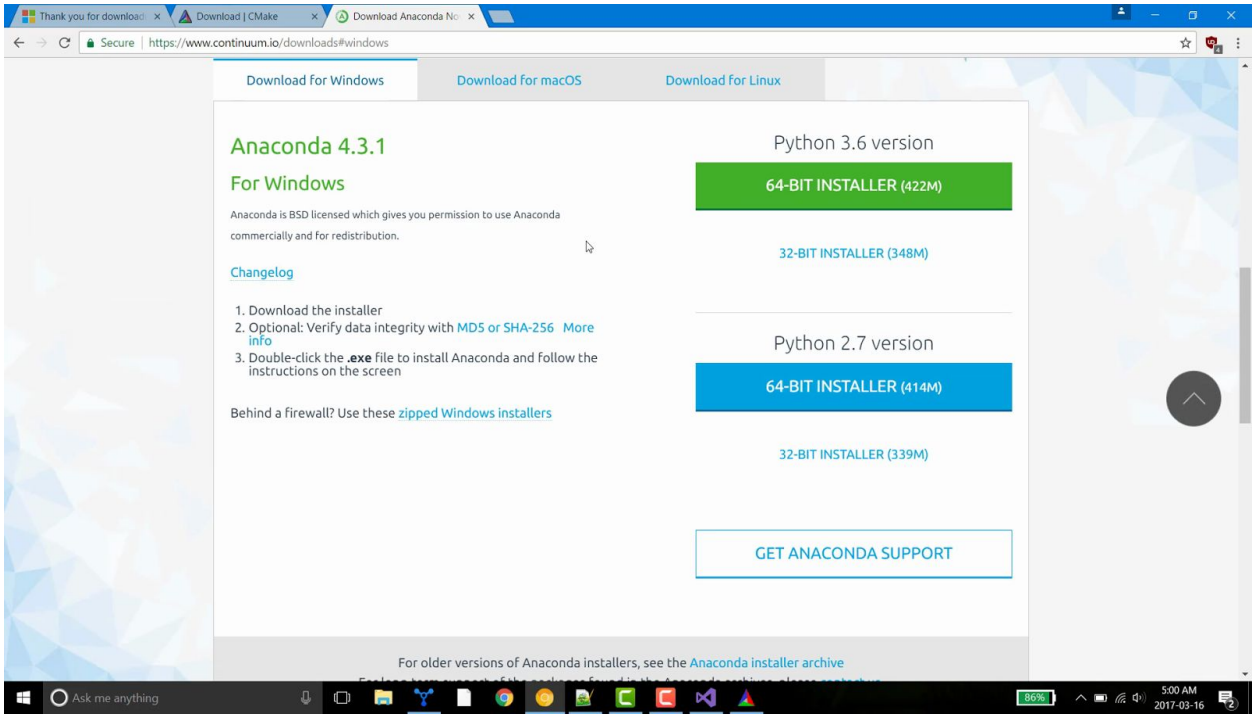


During installation select “Add CMake to system PATH”



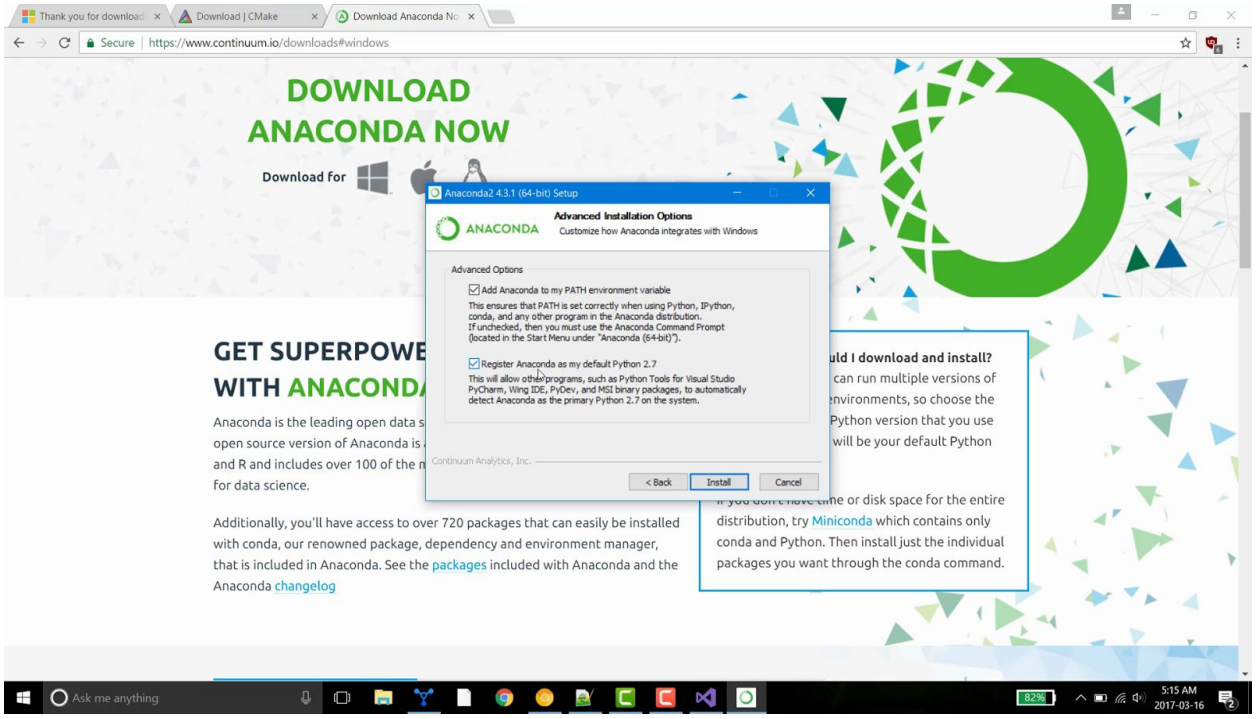
Step 3: Install Anaconda (a python distribution)

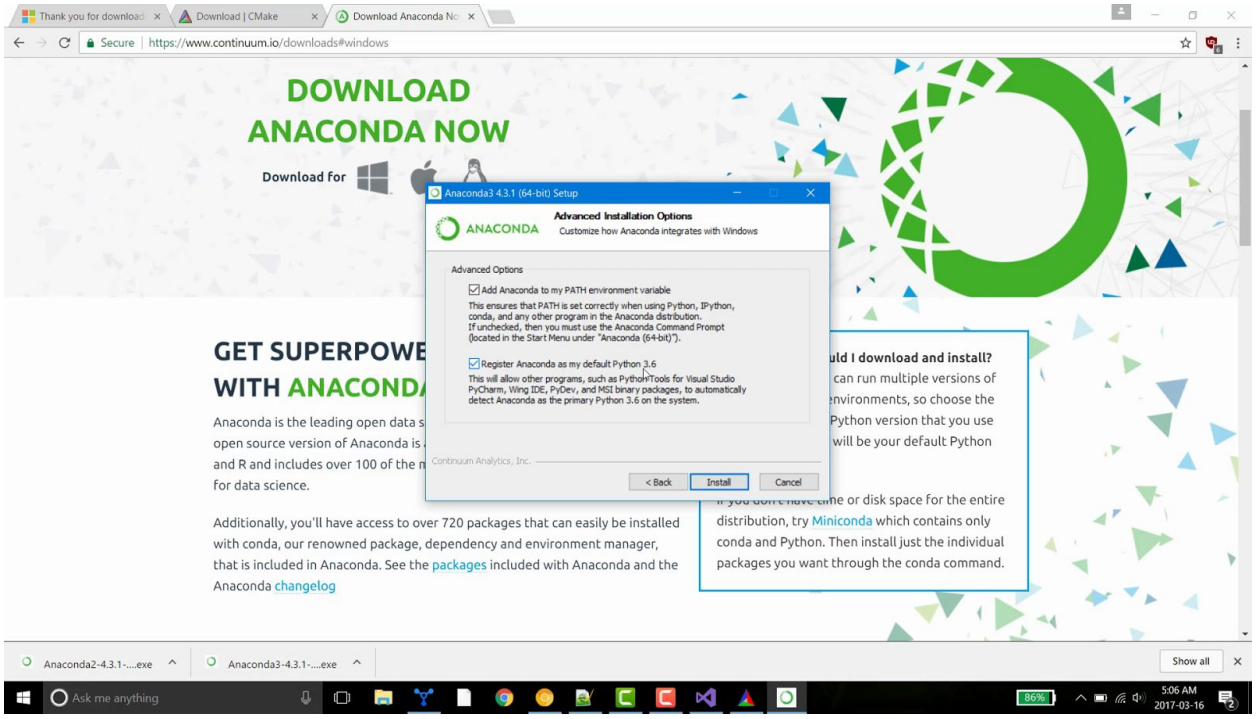
Download and install Anaconda 64-bit version from https://www.continuum.io/downloads. You can install Anaconda 2 or Anaconda 3, or both.



While installing Anaconda, make sure that you check both options:

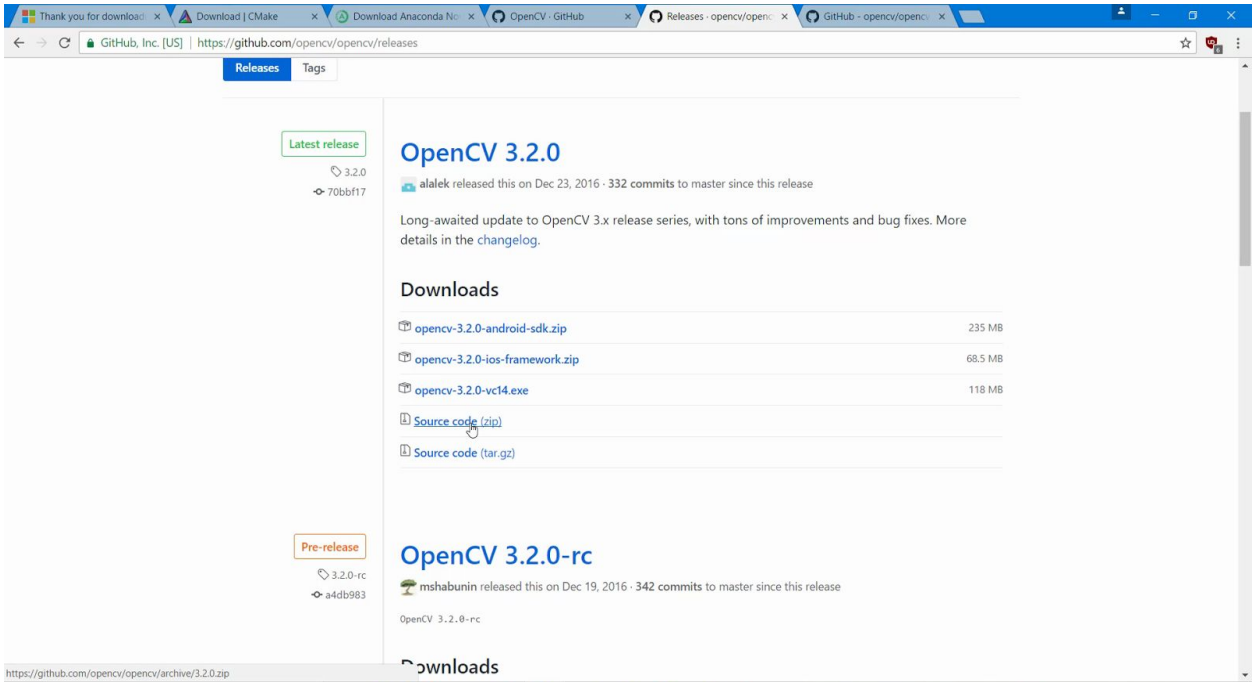
a) Add Anaconda to my PATH environment variable
b) Register Anaconda as my default Python



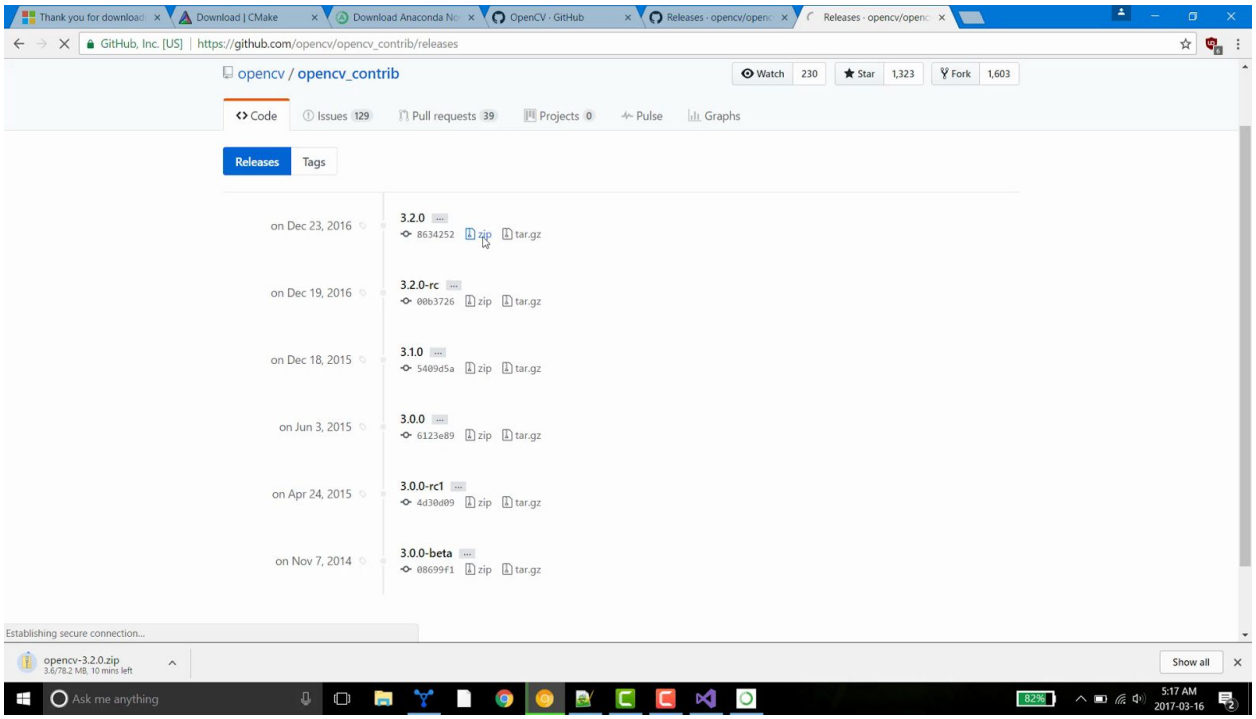


Step 4: Download and extract opencv-3.2.0 and opencv_contrib-3.2.0

Go to https://github.com/opencv/opencv/releases and download opencv-3.2.0 source code zip



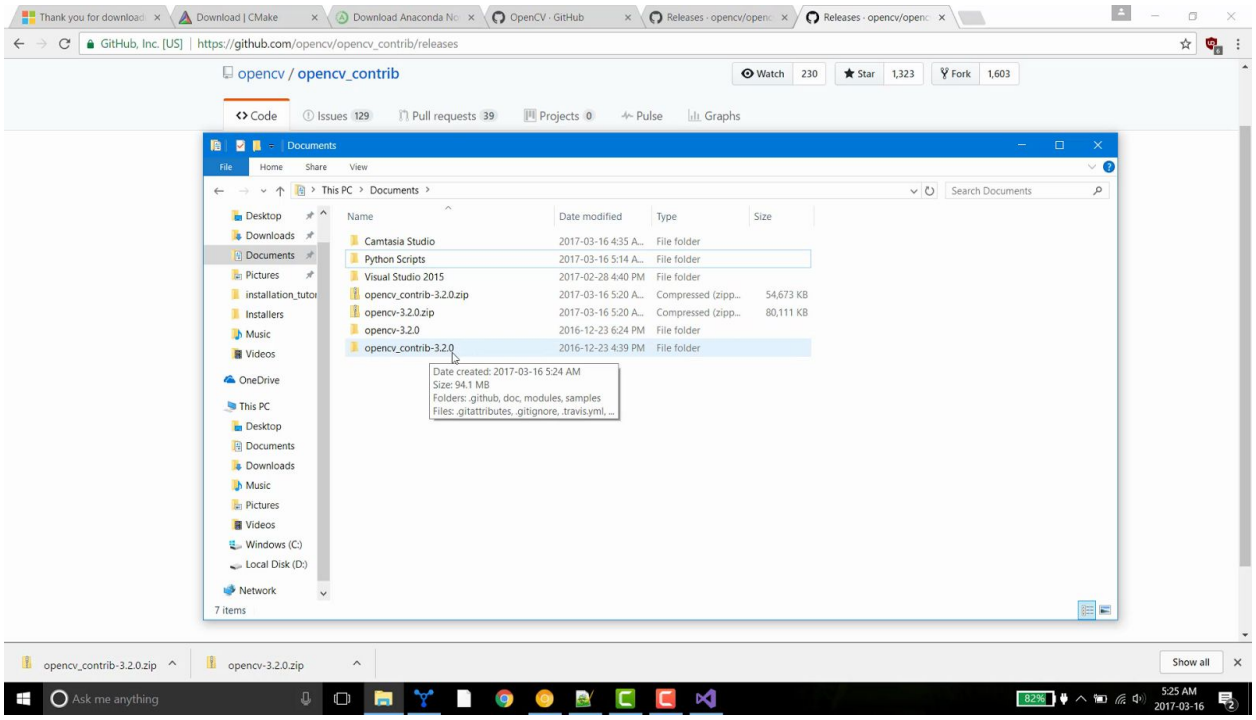
Go to https://github.com/opencv/opencv_contrib/releases and download opencv_contrib-3.2.0 source code zip.



Extract both zip files. Although you can keep opencv and opencv_contrib folders anywhere, I suggest that you should keep both in the same directory. I have placed these two folders in “My Documents” directory.

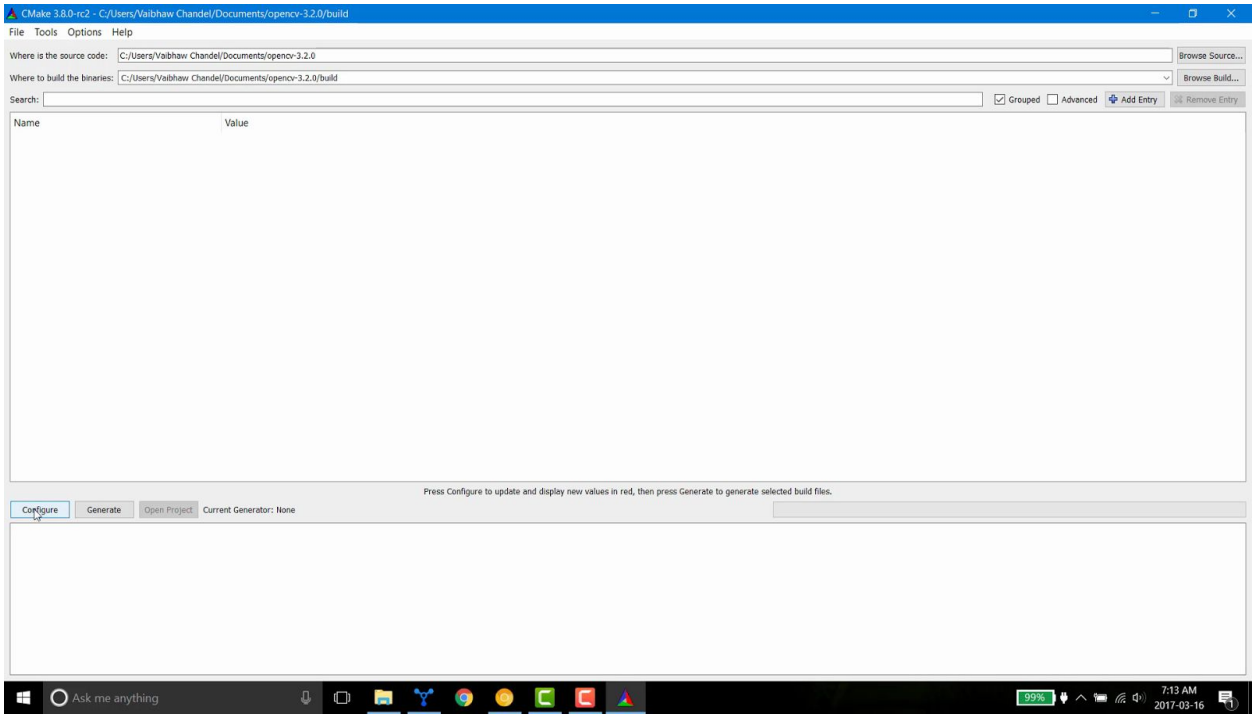
From hereon we will refer to the path to opencv-3.2.0 folder as **OPENCV_PATH**. In my case OPENCV_PATH is **C:/Users/Vaibhaw Chandel/Documents/opencv-3.2.0**

Depending on the location of your opencv-3.2.0 folder, this path would be different.



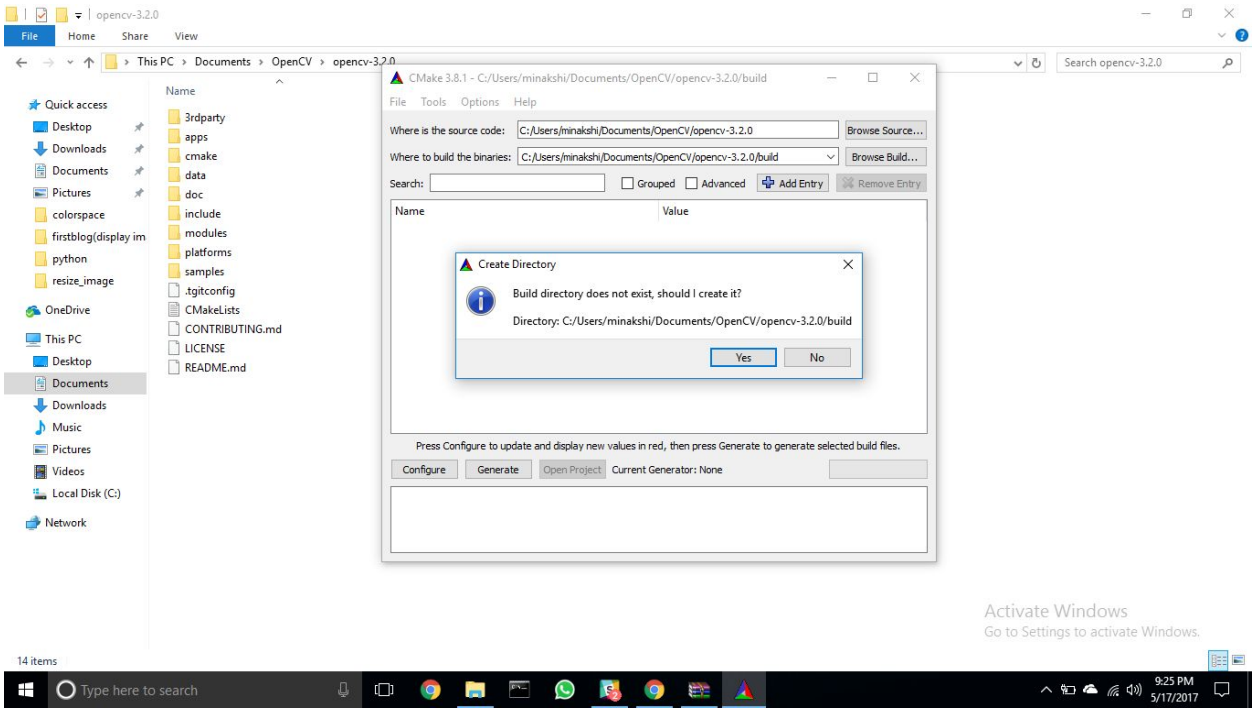
Step 5: Generate Visual Studio project using CMAKE

Run Cmake, in box “Where is the source code” write value of **OPENCV_PATH** (which is path to opencv-3.2.0 folder) and path to build directory. We will choose build directory as **OPENCV_PATH**/build

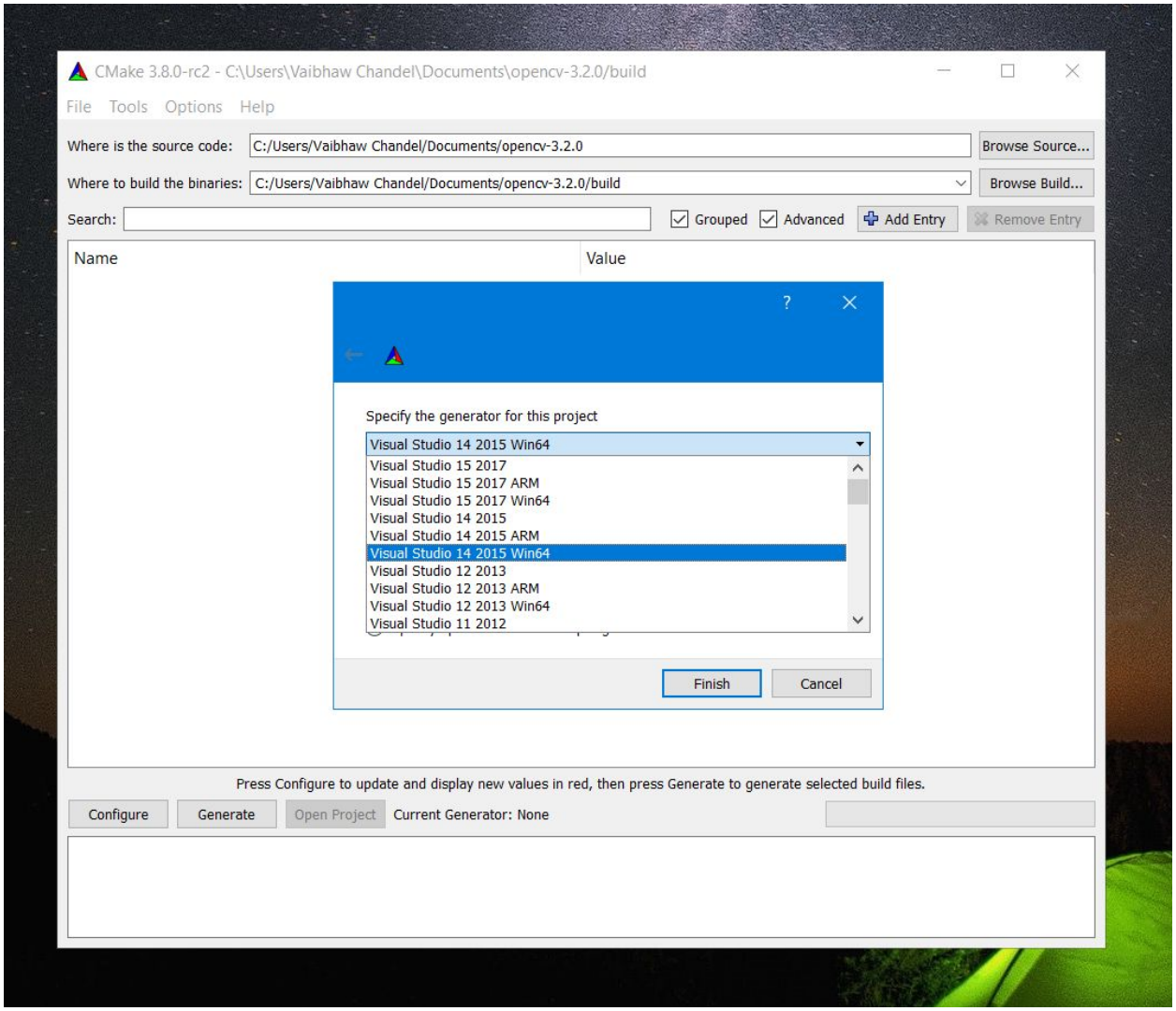


Now click configure.

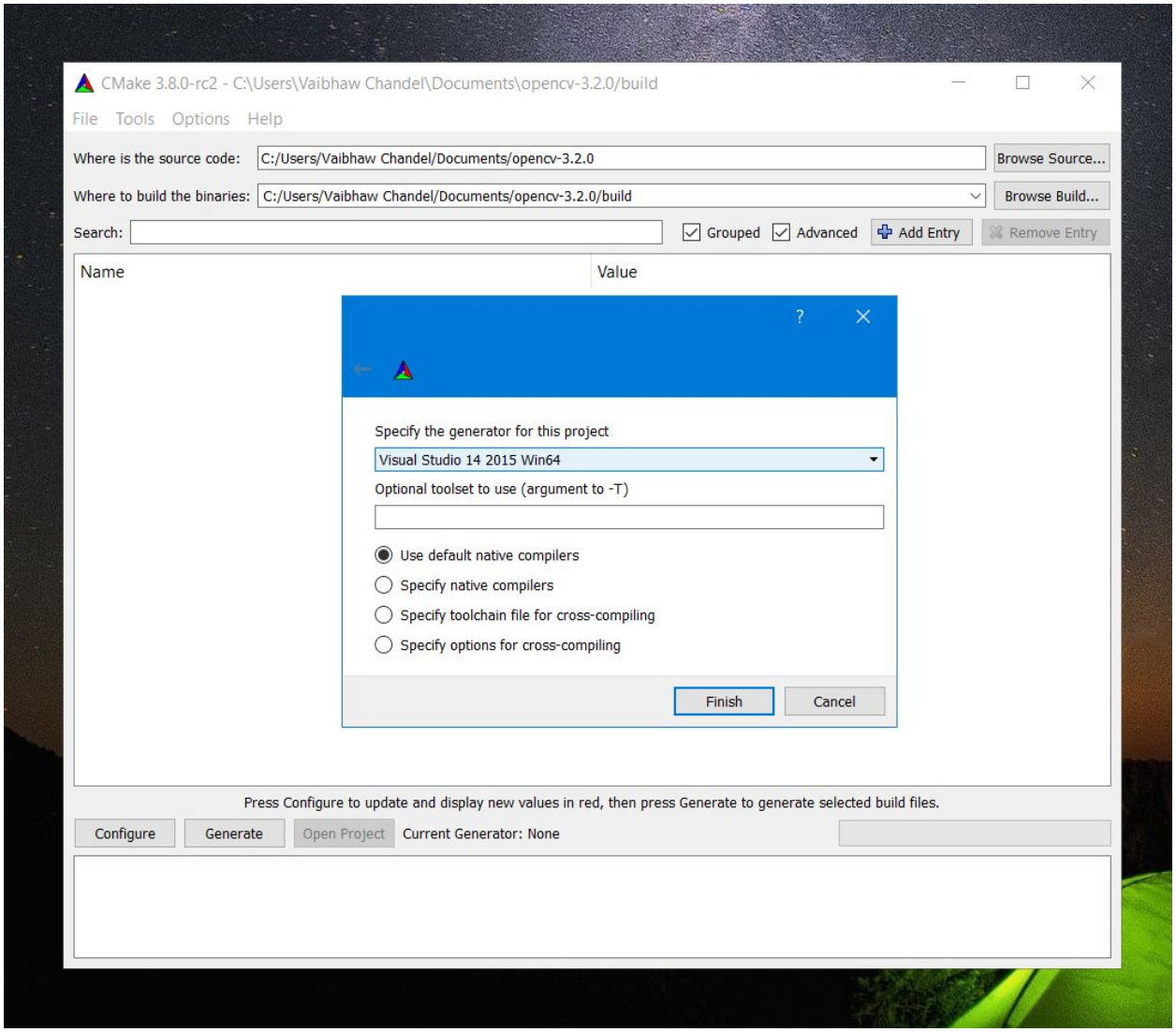
You will be asked for permission to create the build folder. Click Yes.



When prompted to select a compiler, select **Visual Studio 14 2015 Win64**.



Click finish and in the next window keep the default parameters checked.

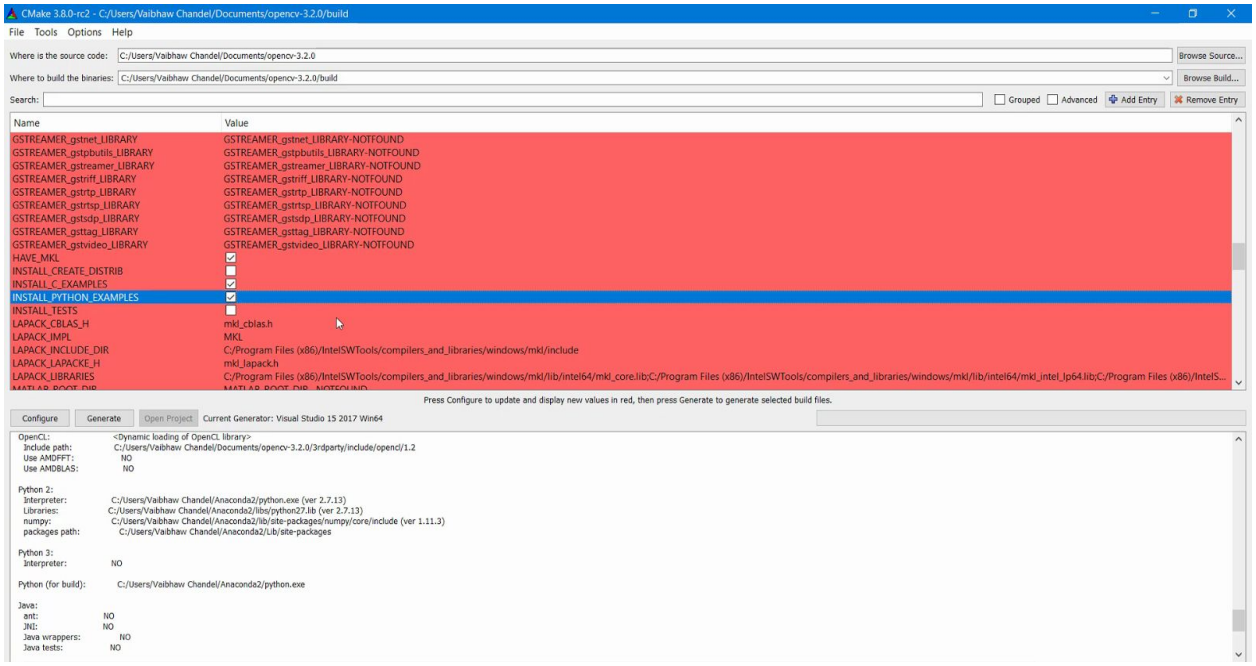


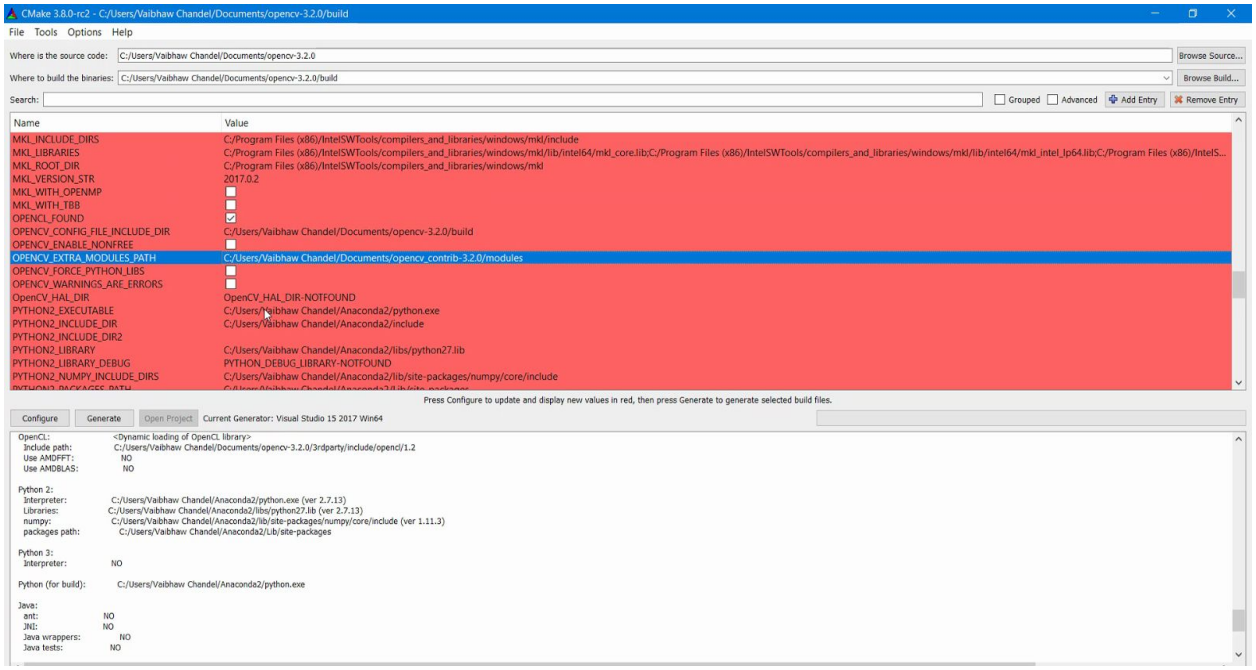
Click finish. Now CMake will look in the system directories and generate the makefiles.

Step 5.1 : Additional changes to CMake config

We will make a few changes in the default configuration generated by CMake.

a) Check “INSTALL_C_EXAMPLES” and “INSTALL_PYTHON_EXAMPLES”
b) In flag “OPENCV_EXTRA_MODULES_PATH”, give path of modules directory within opencv_contrib-3.2.0. In our case we have kept opencv_contrib-3.2.0 in Documents folder so path is “C:/Users/Vaibhaw Chandel/Documents/opencv_contrib-3.2.0/modules”



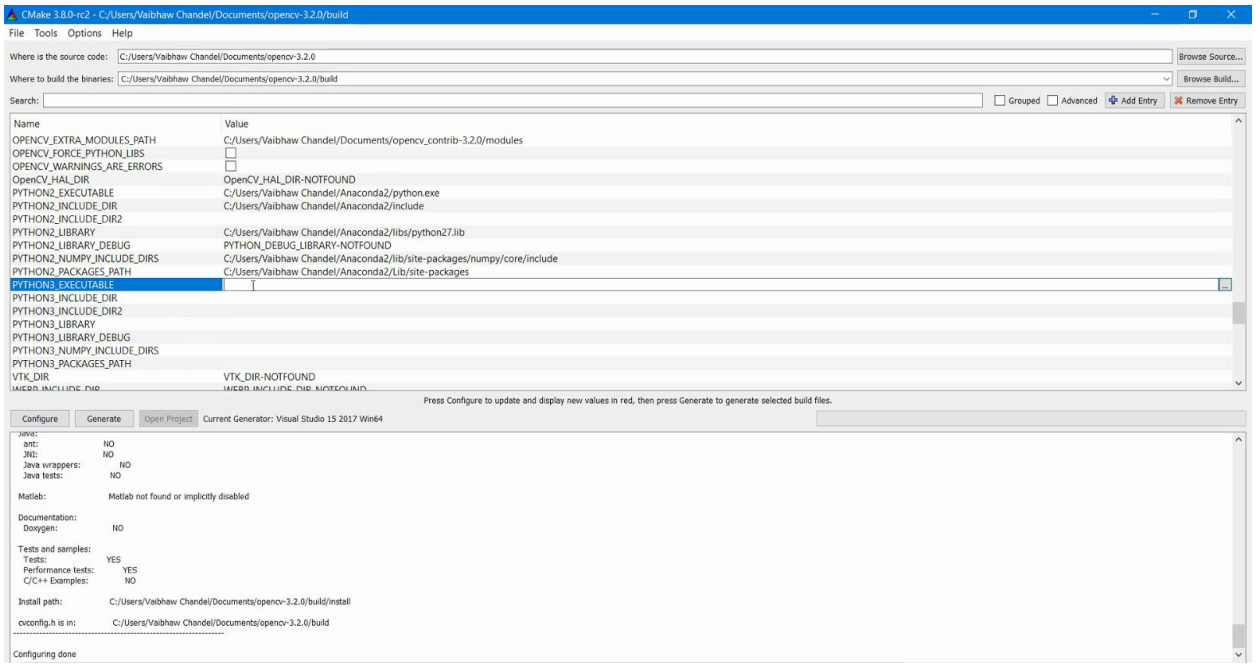


Now click on configure again.

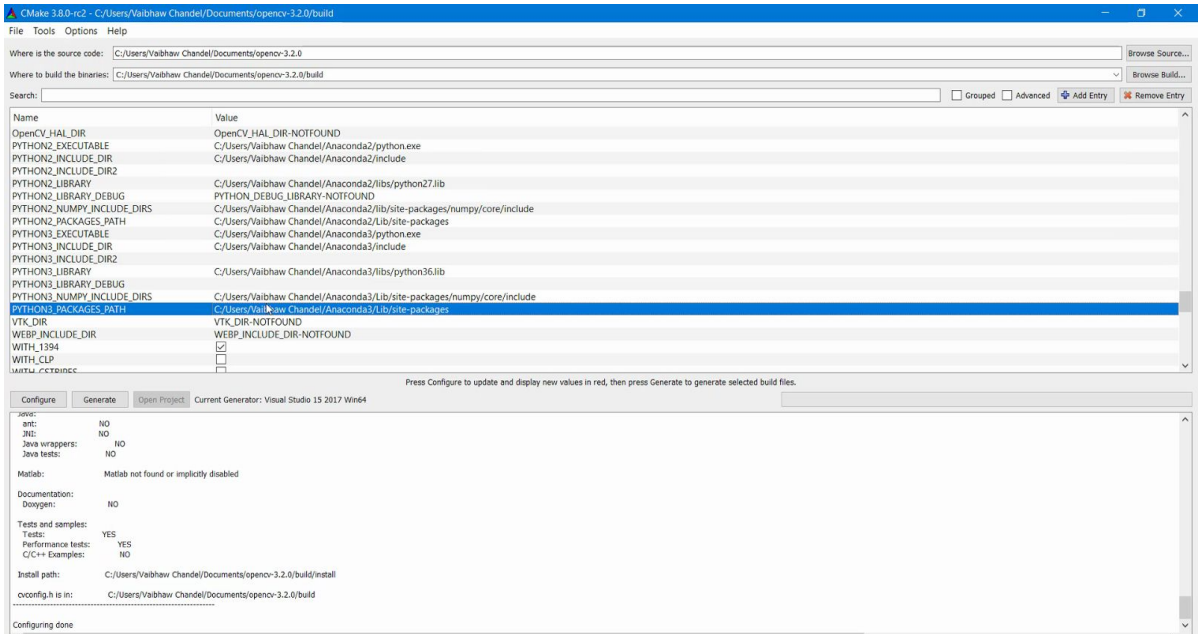
Step 5.2 : Add Python paths for both Python2 and Python3 (optional)

This section is only for people who want to generate OpenCV binary for both Python2 and Python 3. If you are going to use just one Python either 2 or 3, you should skip this section.

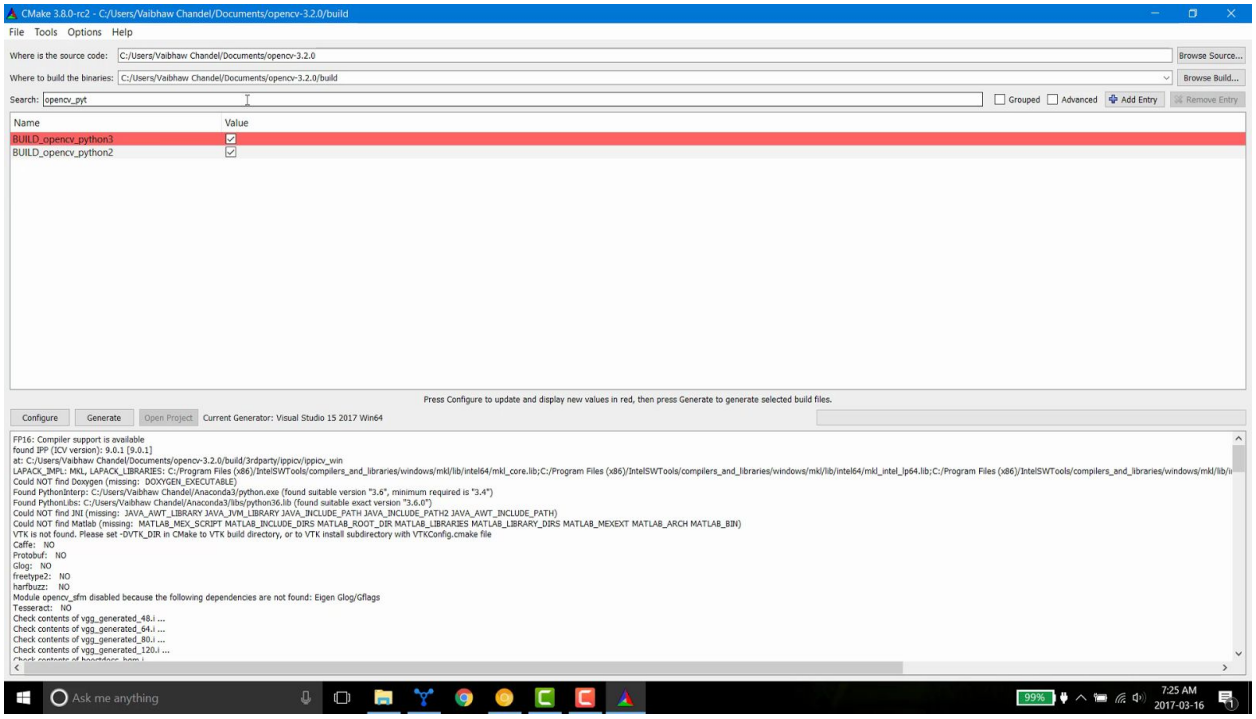
CMake was unable to find paths for my Python3 files.



So I manually added paths for Python3.



Now click configure again. After configuring is done, search **opencv_python** in search bar, both **BUILD_opencv_python2** and **BUILD_opencv_python3** should be checked. Now we are sure that OpenCV binaries for both Python2 and Python 3 will be generated after compilation.



Step 5.3 : Generate build files

If CMake is able to configure without any errors it should say “Configuring done”.

Click generate.

Note: Whenever you make any changes(check/uncheck boxes or change path) to configuration generated by CMake, always click configure and generate.

Step 6: Compile OpenCV

Step 6.1 : Compile opencv in Release mode

Open Windows Command Prompt (cmd).

Go to OPENCV_PATH/build directory and run this command

```
cmake.exe --build . --config Release --target INSTALL
```

Step 6.2 : Compile opencv in Debug mode

Open CMake GUI again as mentioned in Step 5.

1. Search “python” in search box
2. Uncheck INSTALL_PYTHON_EXAMPLES, BUILD_opencv_python3 and BUILD_opencv_python2
3. Click configure
4. Click generate

Now in windows command prompt

Go to OPENCV_PATH/build directory and run this command

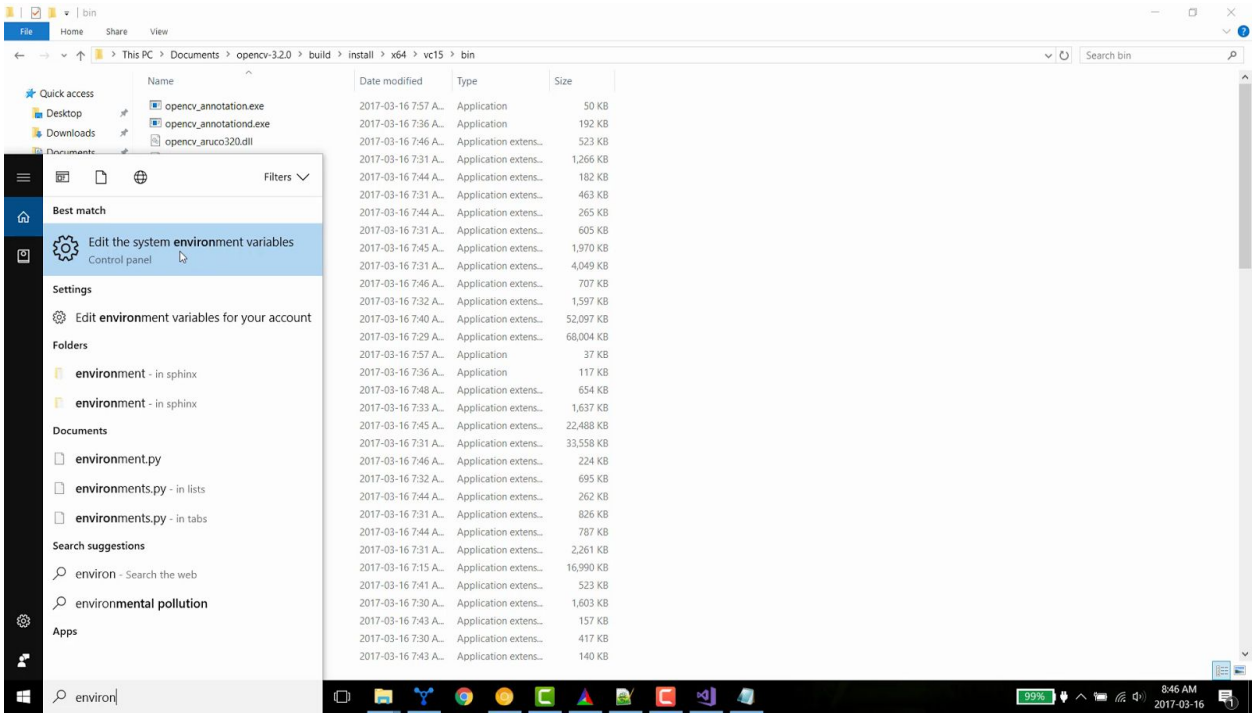
```
cmake.exe --build . --config Debug --target INSTALL
```

Now that we have compiled OpenCV we will find out how to test a OpenCV project using CMake.

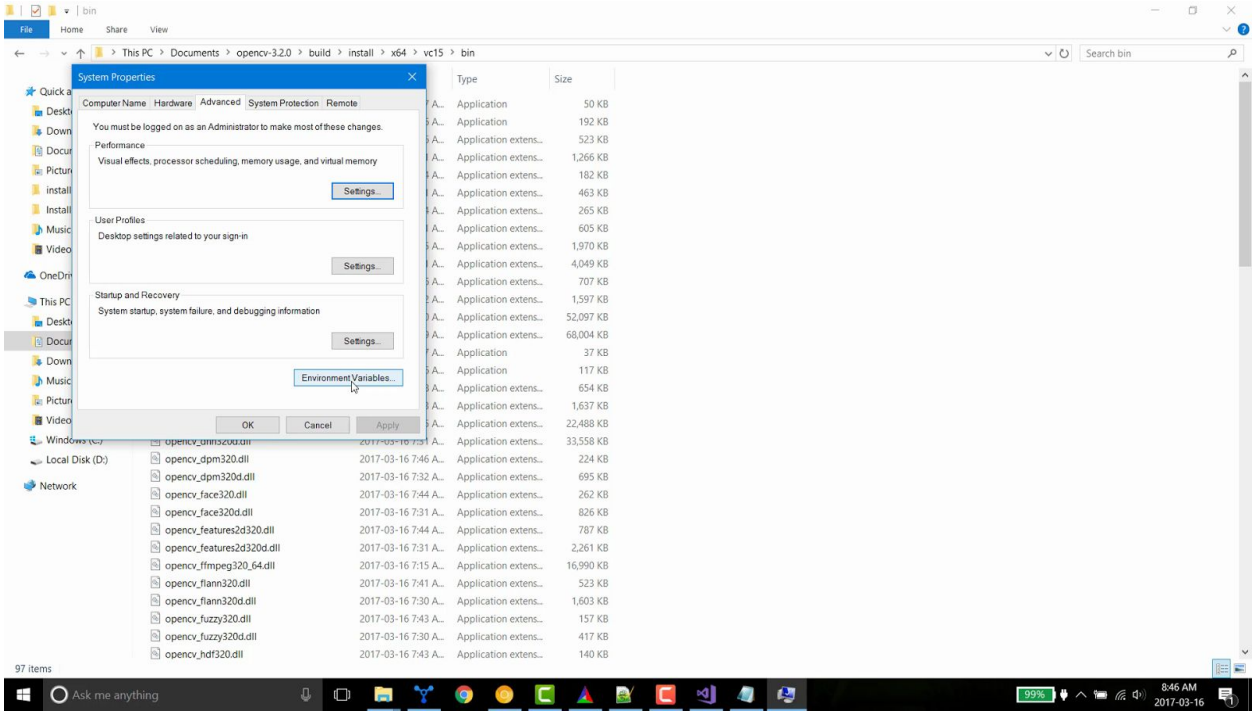
Step 7: Update System Environment Variables

Step 7.1 : Update environment variable - PATH

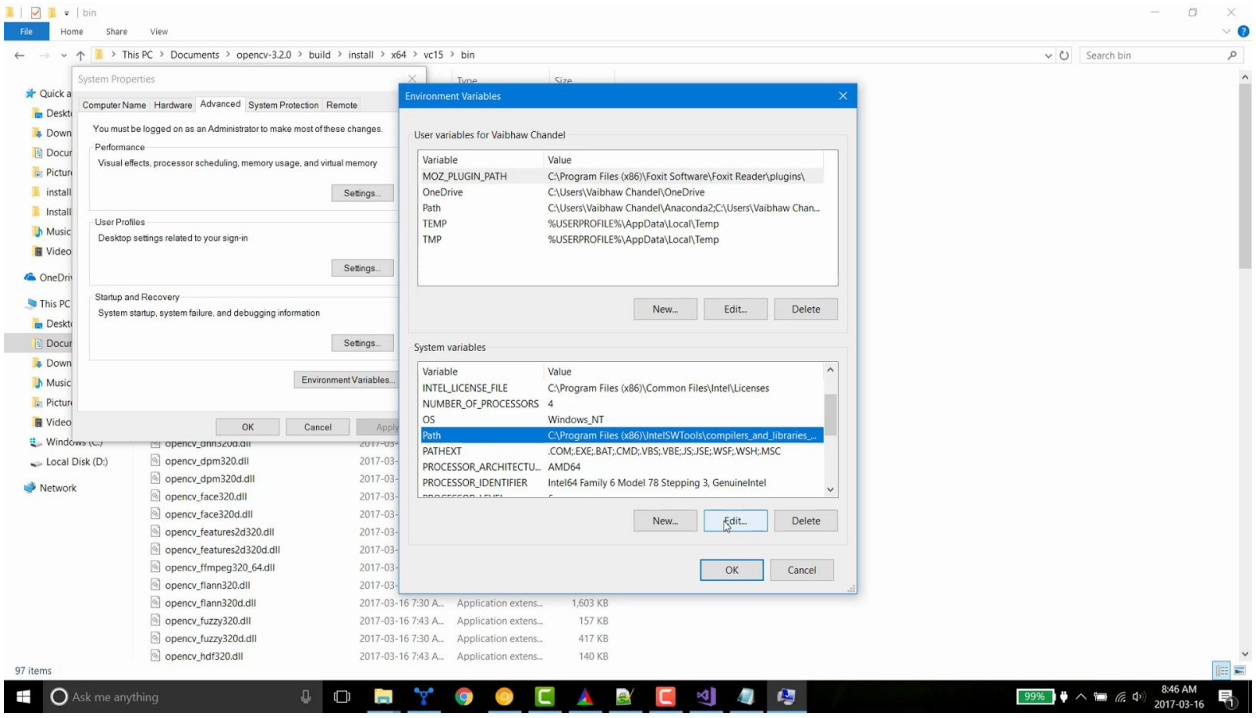
First of all we will add OpenCV dll files’ path to our system PATH. Press Windows Super key, search for “environment variables”



Click Environment Variables in System Properties window

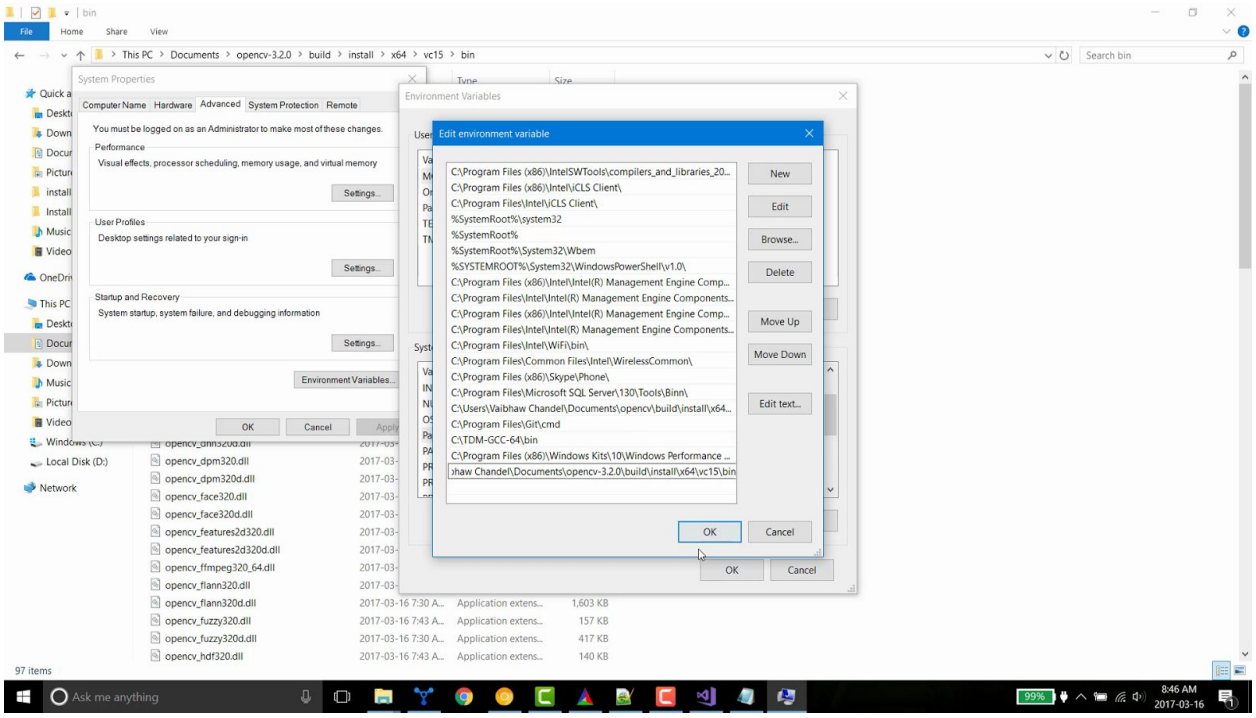


Under System Variables, Select Path and click edit



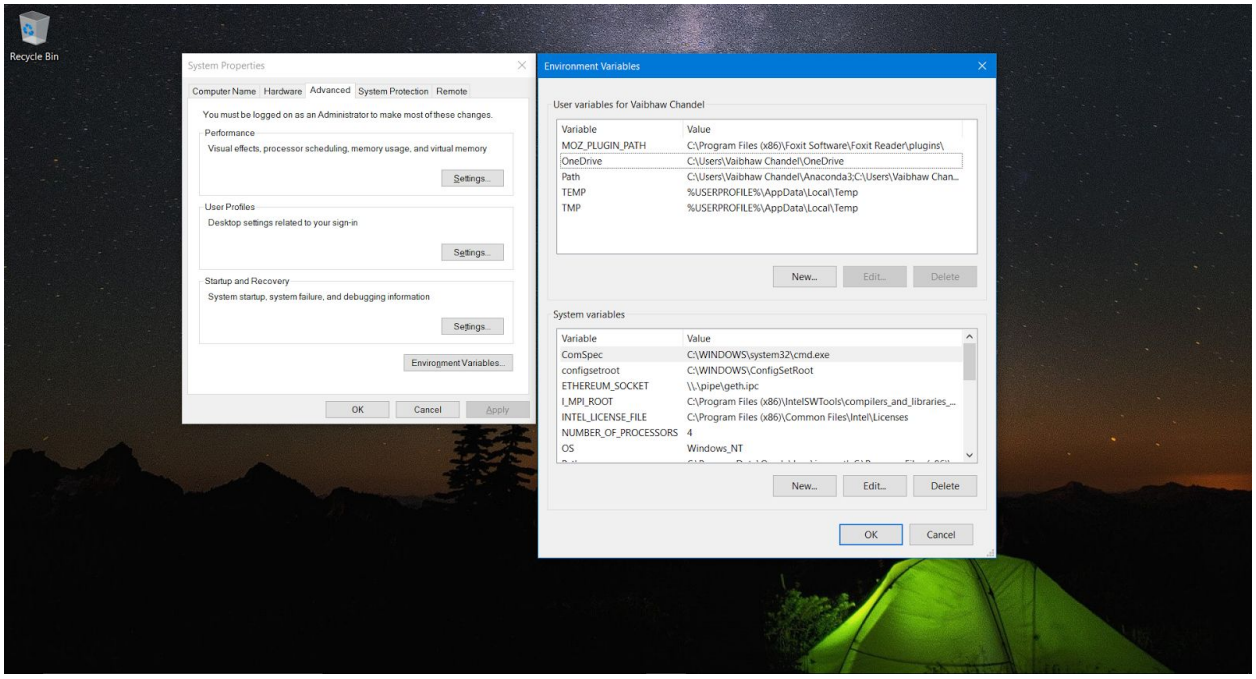
Click New, and give path to **OPENCV_PATH**\build\install\x64\vc14\bin and click Ok. Depending upon where you have kept opencv-3.2.0 folder and what version of Visual Studio you used to compile OpenCV, this path would be different. In my case full path is:

C:\Users\Vaibhaw Chandel\Documents\opencv-3.2.0\build\install\x64\vc14\bin



Now click Ok to save. Don't close the Environment Variables window yet. We will update OPENCV_DIR variable in next step.

Step 7.2 : Update user environment variable - OPENCV_DIR

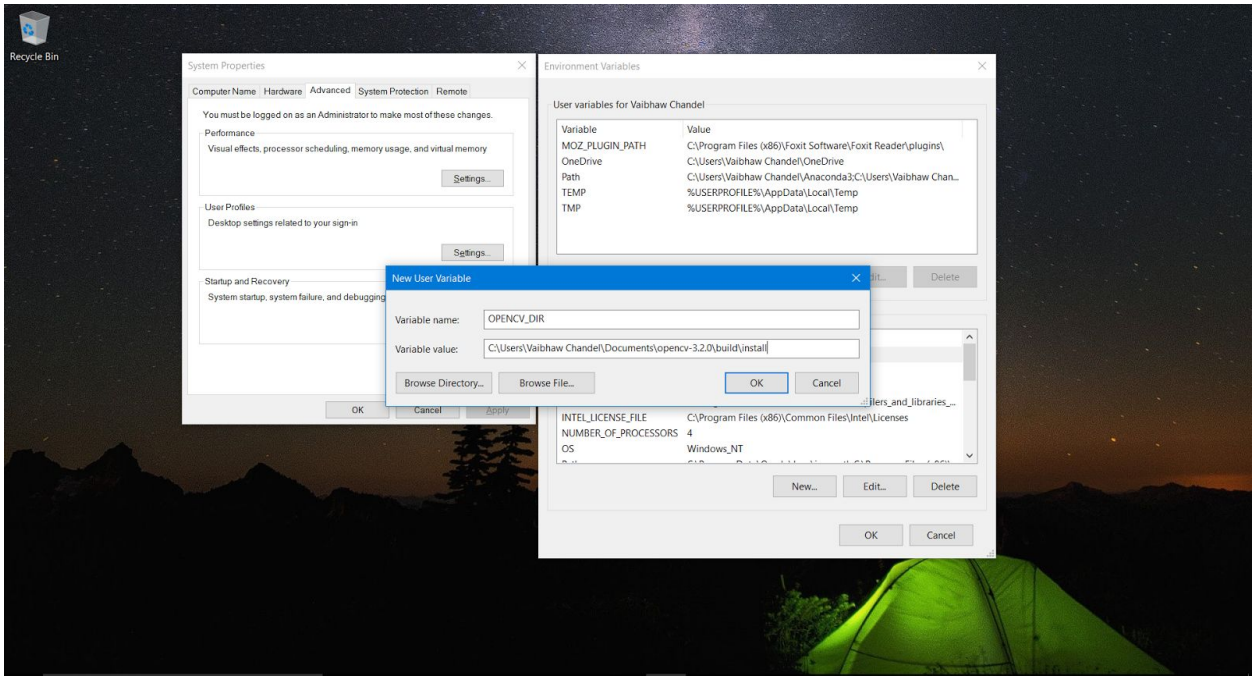


Click New in “User Variables” (upper half of right hand side window). Under variable name write OPENCV_DIR and under variable value write **OPENCV_PATH**\build\install.

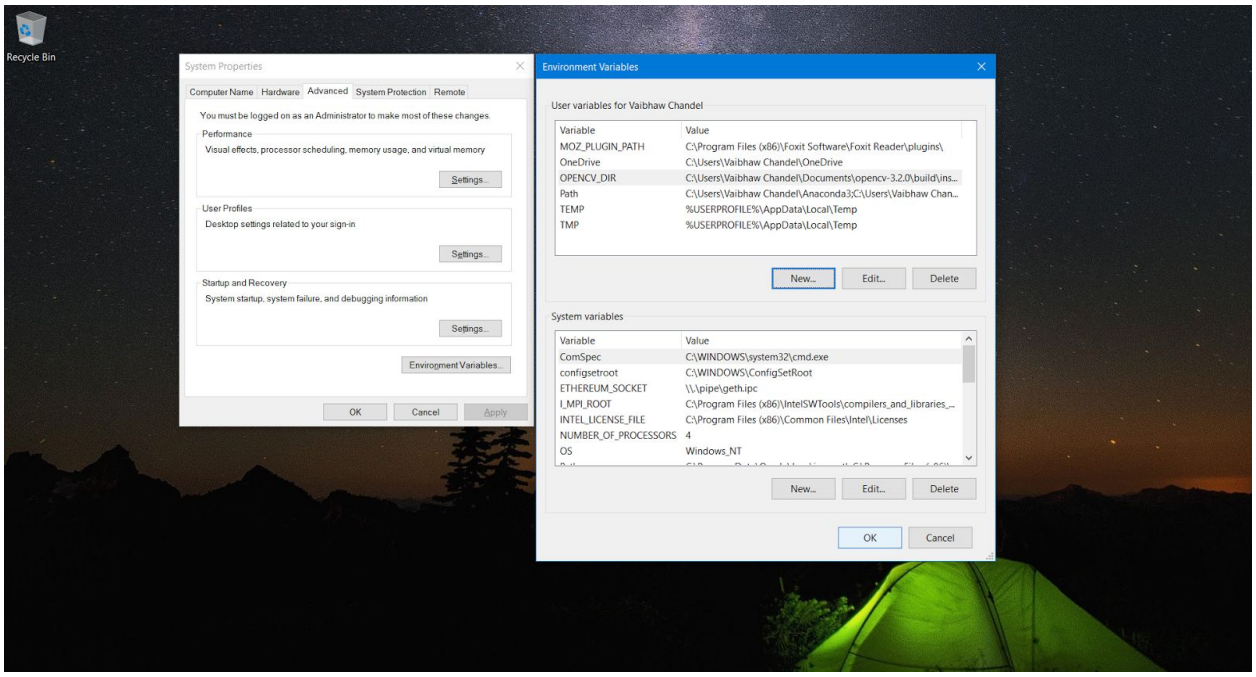
As you can see in my case variable value is:

C:\Users\Vaibhaw Chandel\Documents\opencv-3.2.0\build\install

This directory contains file “OpenCVConfig.cmake”. This is used by CMake to configure OpenCV_LIBS and OpenCV_INCLUDE_DIRS variables to generate project files.



Now click ok to save and close environment variables window.



Note: If you have an open Command Prompt/Power Shell window before these values were updated, you have to close and open a new Command Prompt/Power Shell window again.

Step 8: Testing C++ code

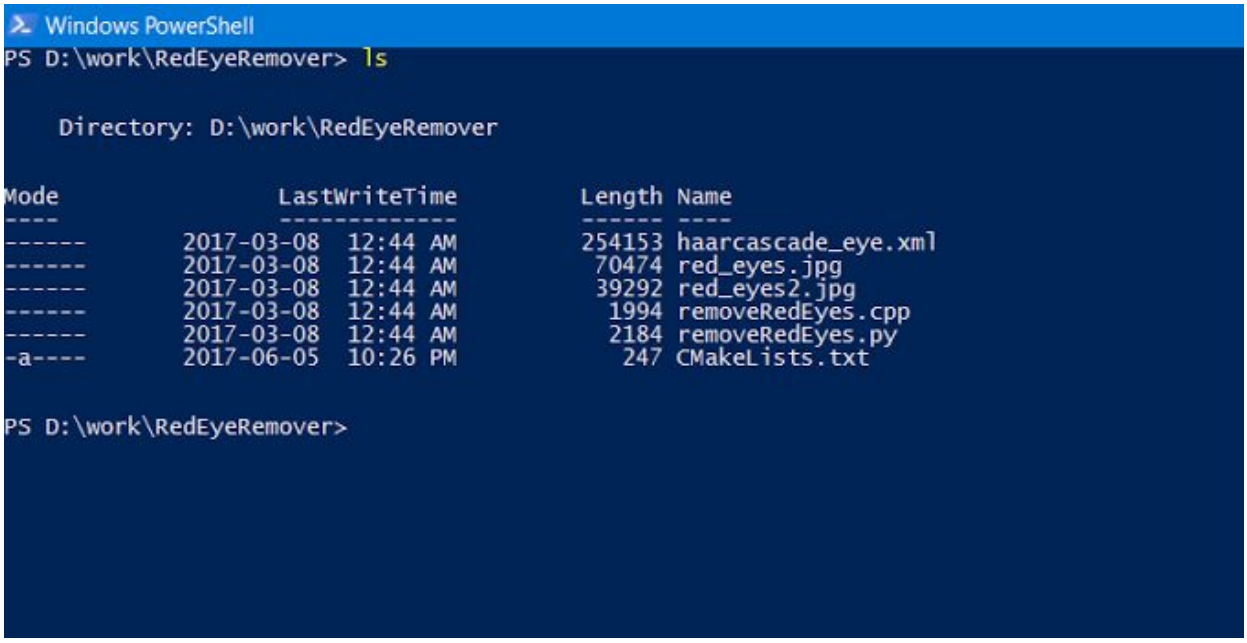
Download this redEyeRemover code and extract it into a folder.

Now open Windows Power Shell and navigate to this directory.

Create a file named CMakeLists.txt and put this code in this file.

```
cmake_minimum_required(VERSION 2.8)
project( redEyeRemover )
find_package( OpenCV REQUIRED )
include_directories( ${OpenCV_INCLUDE_DIRS} )
add_executable( removeRedEyes removeRedEyes.cpp )
target_link_libraries( removeRedEyes ${OpenCV_LIBS} )
```

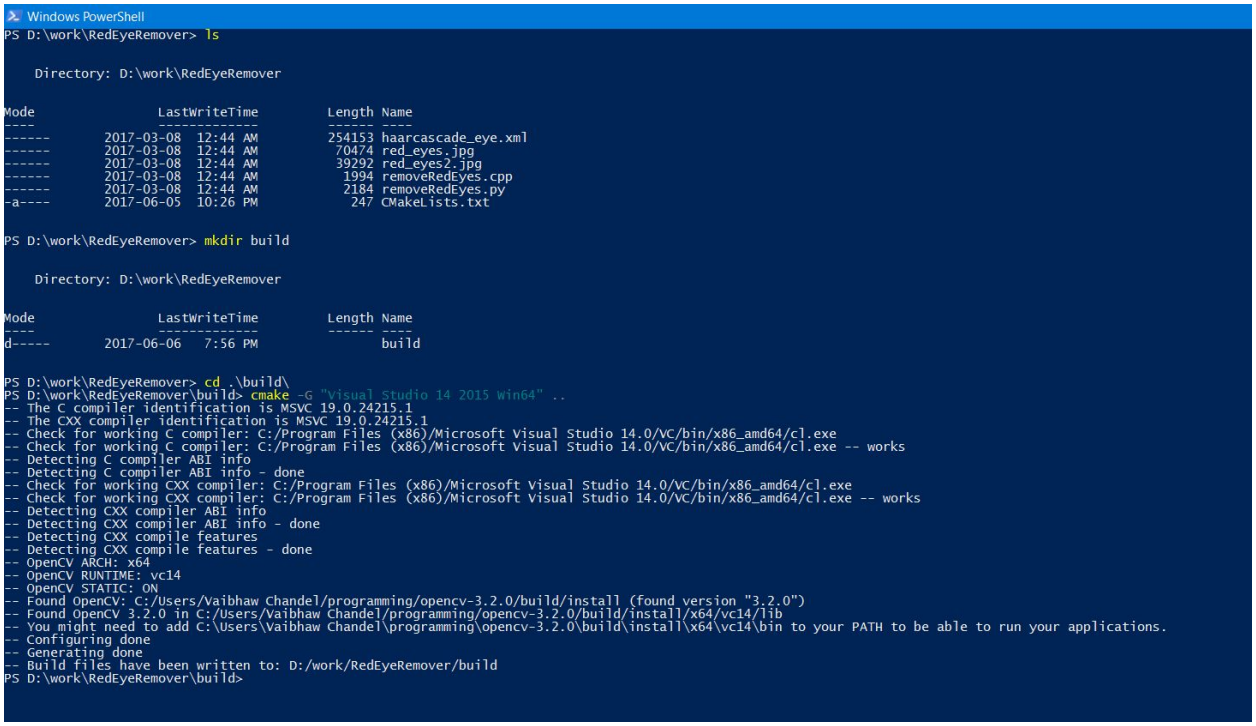
This file has information about opencv's include and library paths.



Now we will compile removeRedEyes.cpp and run it.

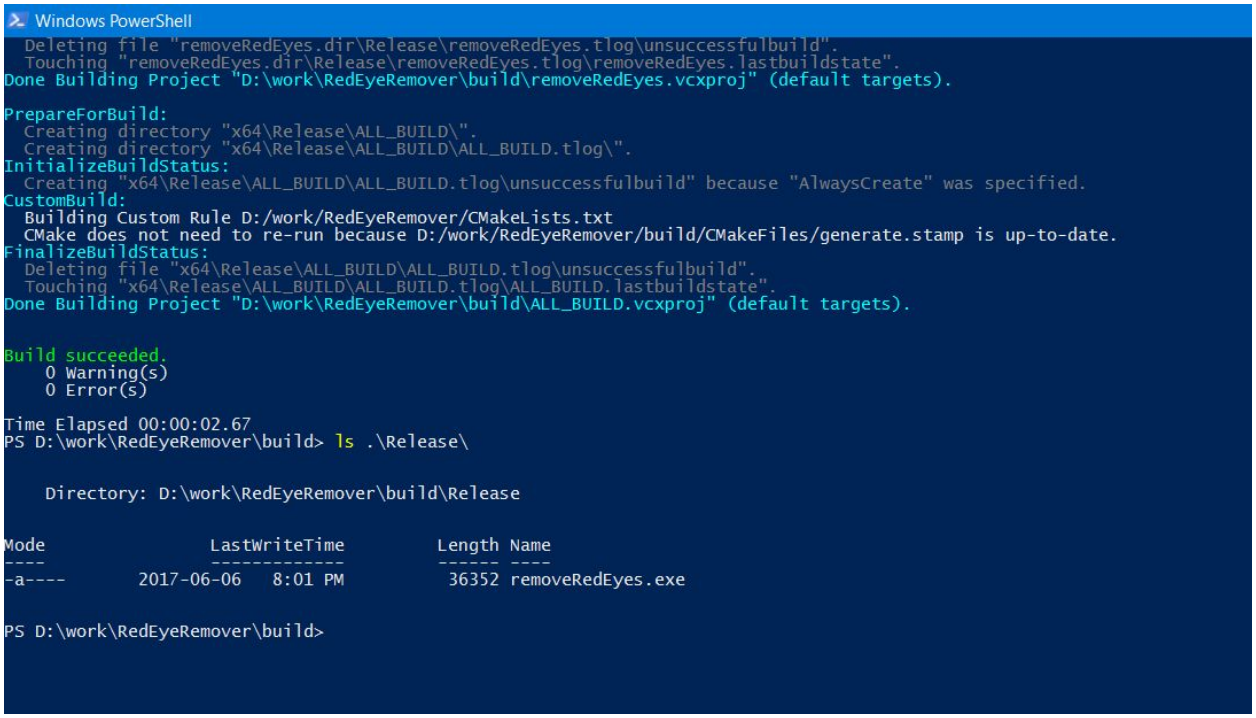
```
# create build directory
mkdir build
cd build
# create Visual Studio project files using cmake
cmake -G "Visual Studio 14 2015 Win64" ..
```

You can see in the screenshot below that CMake found the OpenCV on my machine.



Now we will build our application.

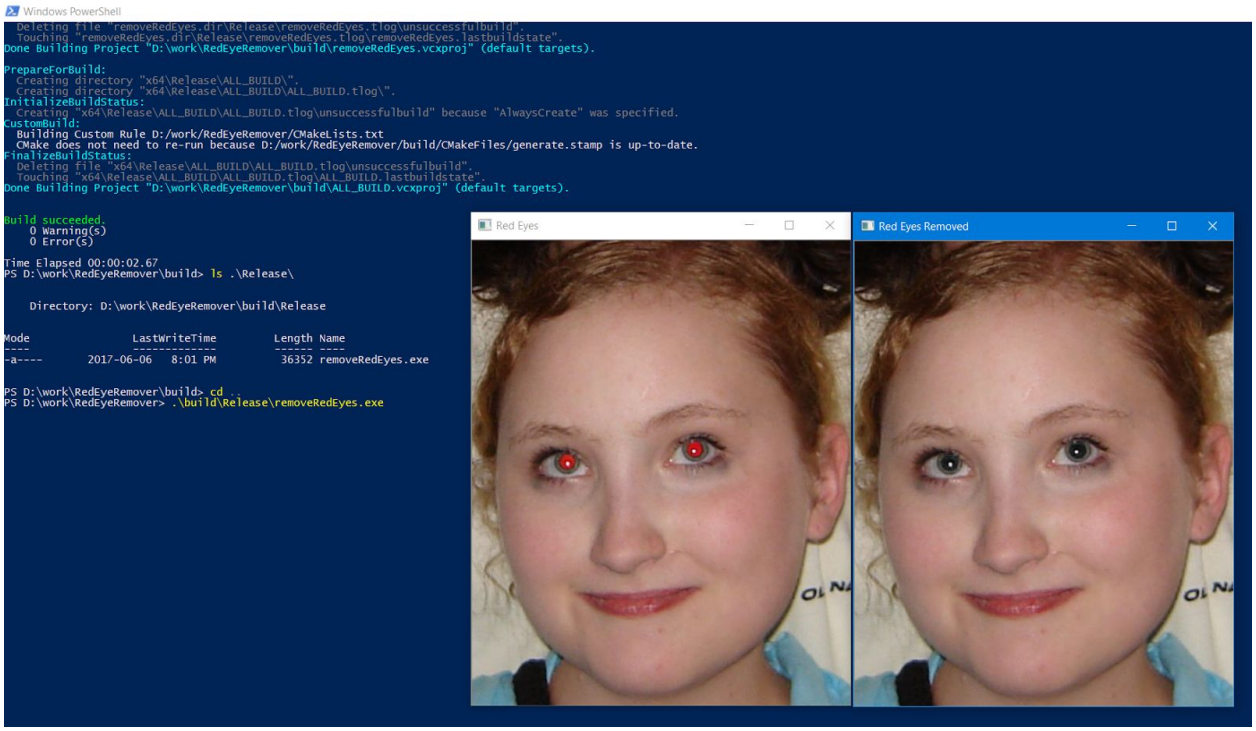
```
# build our application
cmake --build . --config Release
# once the build is complete, it will generate exe file in build\Release directory
```



Since our C++ code assumes that jpg files are in the current directory, we will move to directory RedEyeRemover and run removeRedEyes.exe file from there.

```
cd ..
 .\build\Release\removeRedEyes.exe
```

After running the application you will see two image windows, one with red eyes and another with black eyes.


Windows PowerShell
Deleting file "removeRedEyes.dir\Release\removeRedEyes.tlog\unsuccessfulbuild".
Touching "removeRedEyes.dir\Release\removeRedEyes.tlog\removeRedEyes.lastbuildstate".
Done Building Project "D:\work\RedEyeRemover\build\removeRedEyes.vcxproj" (default targets).
PrepareForBuild:
Creating directory "x64\Release\ALL_BUILD\".
Creating directory "x64\Release\ALL_BUILD\ALL_BUILD.tlog\".
InitializeBuildStatus:
Creating "x64\Release\ALL_BUILD\ALL_BUILD.tlog\unsuccessfulbuild" because "AlwaysCreate" was specified.
CustomBuild:
Building Custom Rule D:/work/RedEyeRemover/CMakeLists.txt
CMake does not need to re-run because D:/work/RedEyeRemover/build/CMakeFiles/generate.stamp is up-to-date.
FinalizeBuildStatus:
Deleting file "x64\Release\ALL_BUILD\ALL_BUILD.tlog\unsuccessfulbuild".
Touching "x64\Release\ALL_BUILD\ALL_BUILD.tlog\ALL_BUILD.lastbuildstate".
Done Building Project "D:\work\RedEyeRemover\build\ALL_BUILD.vcxproj" (default targets).
Build succeeded.
0 Warning(s)
0 Error(s)
Time Elapsed 00:00:02.67
PS D:\work\RedEyeRemover\build> ls .\Release\
Directory: D:\work\RedEyeRemover\build\Release
Mode LastWriteTime Length Name
-a---- 2017-06-06 8:01 PM 36352 removeRedEyes.exe
PS D:\work\RedEyeRemover\build> cd ..
PS D:\work\RedEyeRemover> .\build\Release\removeRedEyes.exe
Red Eyes
Red Eyes Removed

Step 9: Testing Python code

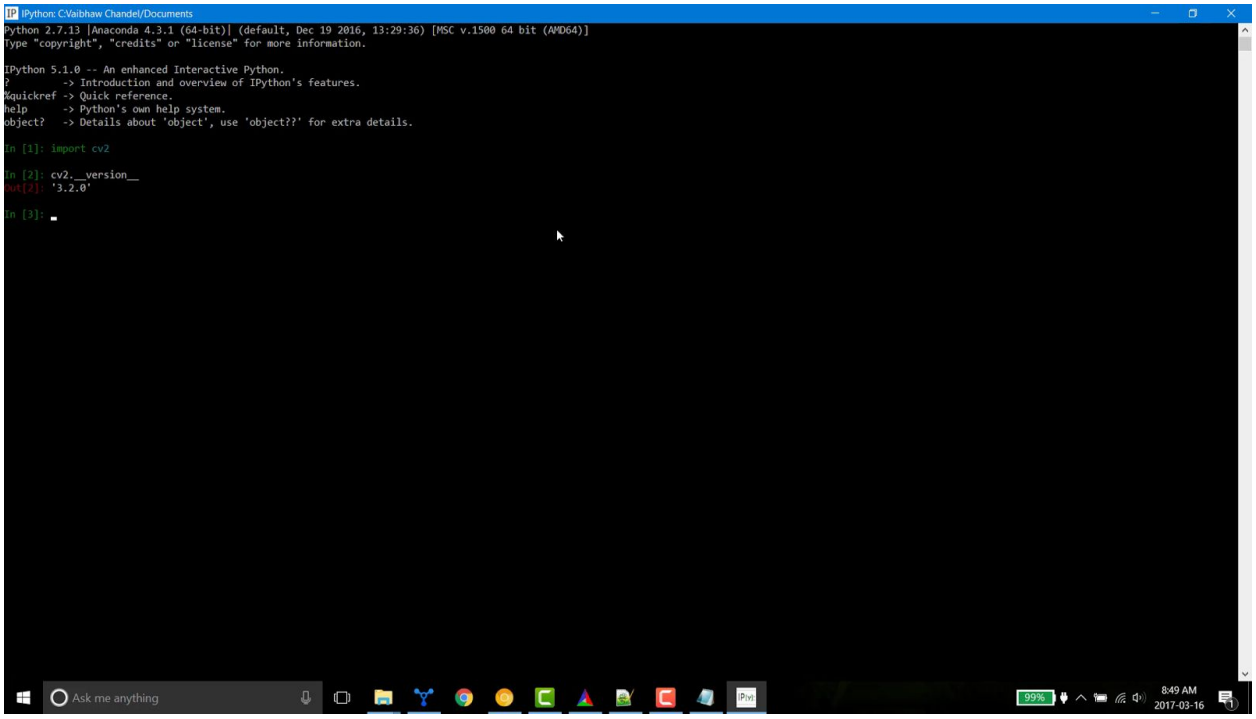
Step 9.1 : Quick check

Quick way to check whether OpenCV for Python is installed correctly or not is to import cv2 in python interpreter.

Open command prompt in Windows, run python command. This will open Python interpreter. Run these two commands

```
import cv2
print(cv2.__version__)
```

Anaconda comes with a feature-rich Python interpreter called IPython. I tested these commands in IPython.



If OpenCV for Python is installed correctly, running command “import cv2” will give no errors. If any error comes up it means installation failed.

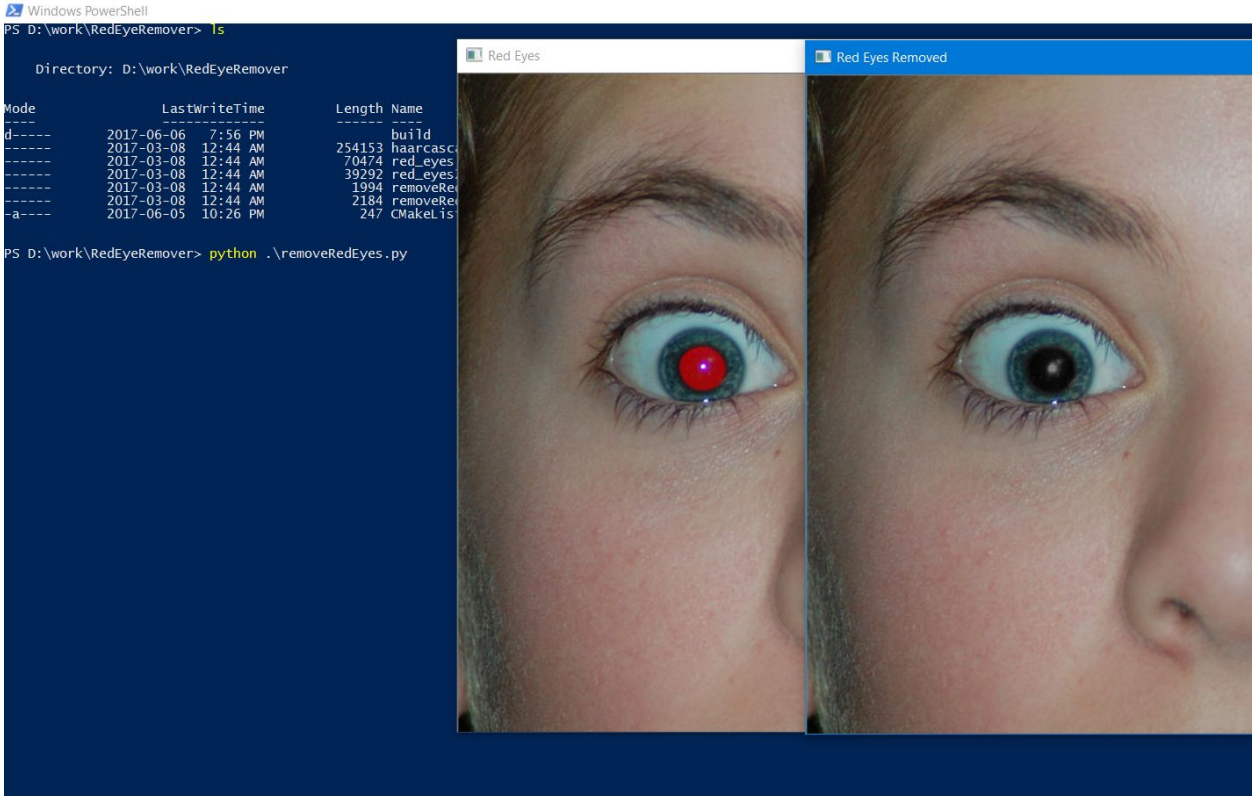
Step 9.2 : Testing redEyeRemover application

Open Windows Power Shell and navigate to directory where you have extracted RedEyeRemover.zip

Now run python code like this:

```
python .\removeRedEyes.py
```

If the program runs successfully, you will see two image windows one with red-eyes other with black eyes.



Installing Dlib on Windows

Step 1: Install CMake

We have already installed CMake while installing OpenCV.

Step 2: Download Dlib

Download Dlib v19.4 from http://dlib.net/files/dlib-19.4.zip

Step 3: Build and Install Dlib library

1. Extract the above compressed file.
2. Open Windows PowerShell.
3. Move to directory where you have extracted this file using cd command and follow the steps given below.

NOTE : If you are running these commands on Command Prompt replace ` (backtick) with ^ (caret).

```
cd dlib-19.4/
mkdir build
cd build
```

```
# This is a single command. Backticks are used for line continuation
cmake -G "Visual Studio 14 2015 Win64" `
-DJPEG_INCLUDE_DIR=..\dlib\external\libjpeg `
-DJPEG_LIBRARY=..\dlib\external\libjpeg `
-DPNG_PNG_INCLUDE_DIR=..\dlib\external\libpng `
-DPNG_LIBRARY_RELEASE=..\dlib\external\libpng `
-DZLIB_INCLUDE_DIR=..\dlib\external\zlib `
-DZLIB_LIBRARY_RELEASE=..\dlib\external\zlib `
-DCMAKE_INSTALL_PREFIX=install ..
```

```
cmake --build . --config Release --target INSTALL
cd ..
```

Dlib will be installed within **dlib-19.4\build\install** directory. We will use cmake to configure and build our projects. But if you want to configure your project in Visual Studio, use the path to inclue and library folders within this directory(dlib-19.4\build\install).

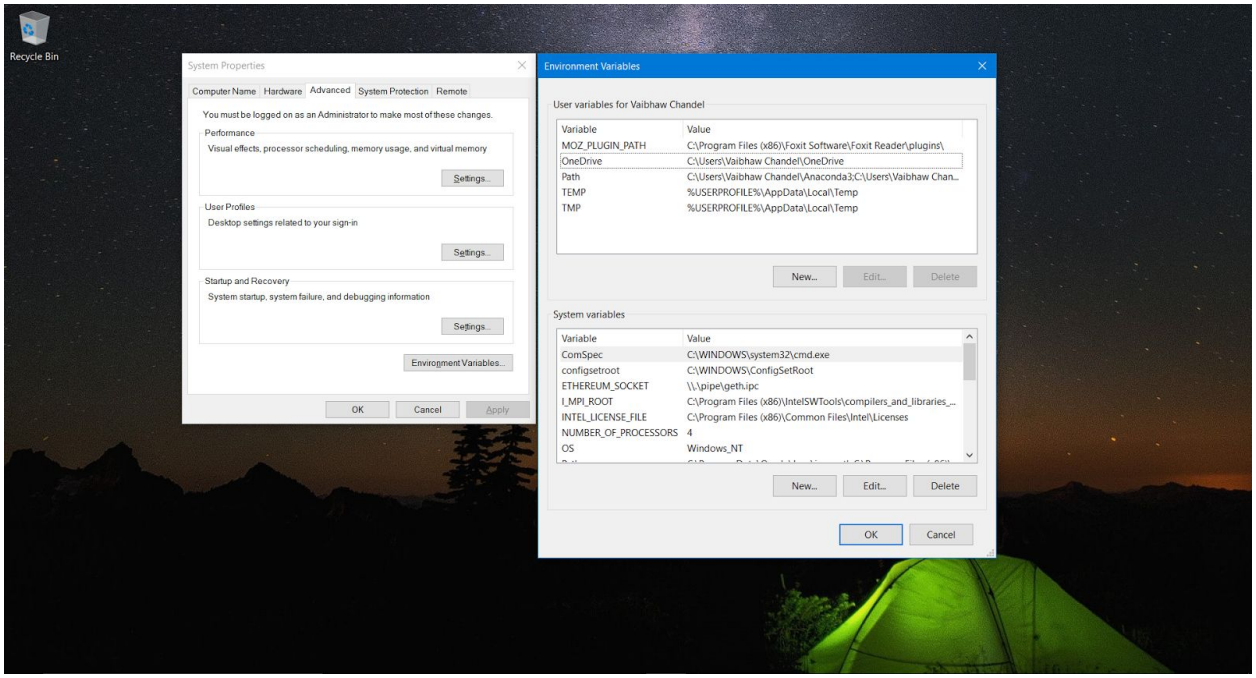
Step 4: Update user environment variable - dlib_DIR

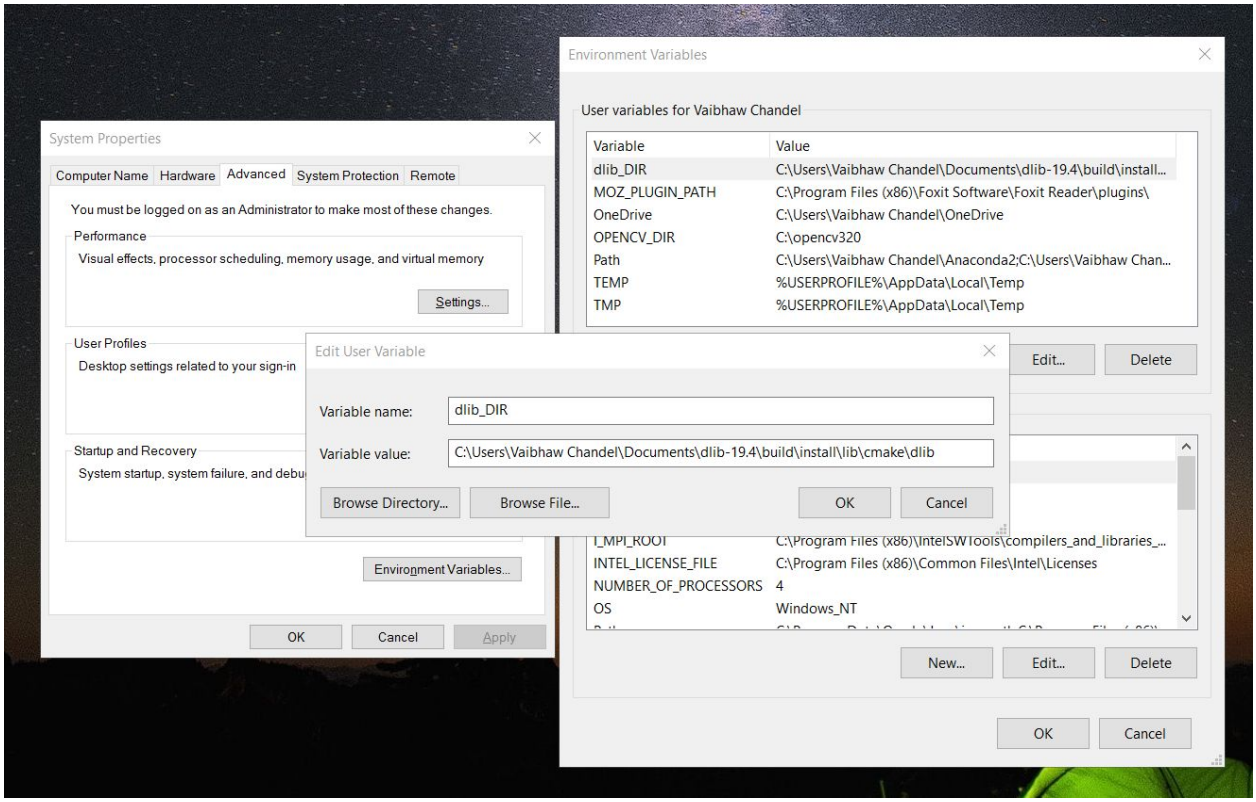
1. Open Environment Variables settings as we did earlier while updating environment variables for OpenCV.
2. Click New in “User Variables” (upper half of right hand side window).
3. Under variable name write dlib_DIR and under variable value write full path to directory **dlib-19.4\build\install\lib\cmake\dlib**

For example, since I kept dlib-19.4 directory in C:\Users\Vaibhaw Chandel\Documents, full path to above directory in my case is:

C:\Users\Vaibhaw Chandel\Documents\dlib-19.4\build\install\lib\cmake\dlib

Note: This directory contains file “dlibConfig.cmake”. This is used by CMake to configure dlib_LIBS and dlib_INCLUDE_DIRS variables to generate project files. Before assigning value to variable dlib_DIR make sure that the path has file dlibConfig.cmake.





Now click ok to save and close environment variables window.

Note: If you have an open Command Prompt/Power Shell window before these values were updated, you have to close and open a new Command Prompt/Power Shell window again.

Step 5: Build Dlib examples

We will use our CMakeLists.txt file instead of one which is shipped with Dlib.

Download CMakeLists.txt file attached in this module and put it in dlib-19.4\examples directory and replace the default one with this one. Then follow the steps given below

```
cd dlib-19.4/examples
mkdir build
cd build
```

```
cmake -G "Visual Studio 14 2015 Win64" ..
```

```
cmake --build . --config Release
cd ../..
```

Step 6: Test Dlib's C++ example

We will test Face Landmark Detection demo to check whether we have installed Dlib correctly.

Download trained model of facial landmarks(shape_predictor_68_face_landmarks.dat.bz2) from Dlib's website. Extract this file to Dlib's root directory (dlib-19.4/).

```
cd examples\build
```

```
# this is a single command
.\Release\face_landmark_detection_ex.exe
..\..\shape_predictor_68_face_landmarks.dat ..\faces\2008_001009.jpg
```

Step 7: Install Dlib's Python module (only for Anaconda 3)

Dlib v19.4 can only be installed for Anaconda 3. Anaconda 2 provides an older version of Dlib which is not useful in our case as we will use some features which were introduced in v19.4

```
conda install -c conda-forge dlib=19.4
```

Step 8: Test Dlib's Python example

Open Windows PowerShell and move to dlib-19.4 directory.

```
cd dlib-19.4/python_examples
```

```
python face_landmark_detection.py ..\shape_predictor_68_face_landmarks.dat
..\examples\faces\2008_001009.jpg
```

Troubleshooting Dlib's C++ read/write image

The method that we have followed to compile C++ example should definitely work. If it doesn't check dlib_DIR environment variable is defined and points to correct path. This path must have file dlibConfig.cmake.

Even after the configuration is correct, you still get an error like this:

```
exception thrown!
Unable to load image in file .\examples\faces\2007_007763.jpg.
You must #define DLIB_JPEG_SUPPORT and link to libjpeg to read JPEG files.
```

Do this by following the instructions at http://dlib.net/compile.html.

Note that you must cause DLIB_JPEG_SUPPORT to be defined for your entire project. So don't #define it in one file. Instead, add it to the C/C++->Preprocessor->Preprocessor Definitions field in Visual Studio's Property Pages window so it takes effect for your entire application.

It means Dlib's image read/write method is not working on your machine. We will follow an alternative approach.

Dlib supports OpenCV's Mat object. So we will use OpenCV for image read/write operations and pass this image object to Dlib for Dlib specific operations.

We will show one such example. We will change face_landmark_detection_ex.cpp and write a CMakeLists.txt using which we will compile this C++ example.

Download **examples_opencv.zip** file and extract it in **dlib-19.4**. This will create examples_opencv folder in dlib-19.4 folder. Now go to Power Shell window and run following commands:

```
cd dlib-19.4/examples_opencv
# this folder should have files,
# face_landmark_detection_opencv_ex.cpp and CMakeLists.txt
mkdir build
cd build
cmake -G "Visual Studio 14 2015 Win64" ..
cmake --build . --config Release
```

Once build is complete, it will generate a file in build\Release folder. Run the executable by running(this is a single command, not two):

```
.\Release\face_landmark_detection_ex.exe
..\..\shape_predictor_68_face_landmarks.dat ..\..\examples\faces\2008_001009.jpg
```