

Module 0.2

# Getting started Ubuntu

---

**Satya Mallick, Ph.D.**

June 19, 2017

# Table of Contents

<b>Installing OpenCV on Ubuntu</b>	<b>2</b>
Step 1: Update packages	2
Step 2: Install OS libraries	2
Step 3: Install Python libraries	2
Step 4: Download opencv and opencv_contrib	3
Step 4.1: Download opencv from Github	3
Step 4.2: Download opencv_contrib from Github	4
Step 5: Compile and install OpenCV with contrib modules	4
Step 5.1 : Create a build directory	4
Step 5.2 : Run CMake	4
Step 5.3 : Compile and Install	4
Step 5.4 : Create symlink in virtual environment	5
Step 6: Test OpenCV 3	5
<b>Installing Dlib on Ubuntu</b>	<b>6</b>
Step 1: Install OS libraries	6
Step 2: Install Python libraries	6
Step 3: Compile DLib	6
Step 3.1: Compile C++ binary	6
Step 3.2: Compile Python module	7

# Installing OpenCV on Ubuntu

## Step 1: Update packages

```
sudo apt-get update
sudo apt-get upgrade
```

## Step 2: Install OS libraries

```
# remove any previous installations of x264
sudo apt-get remove x264 libx264-dev

# we will install dependencies now
sudo apt-get install build-essential checkinstall cmake pkg-config yasm
sudo apt-get install git gfortran
sudo apt-get install libjpeg8-dev libjasper-dev libpng12-dev

# If you are using Ubuntu 14.04
sudo apt-get install libtiff4-dev
# If you are using Ubuntu 16.04
sudo apt-get install libtiff5-dev

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev
sudo apt-get install libxine2-dev libv4l-dev
sudo apt-get install libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev
sudo apt-get install libqt4-dev libgtk2.0-dev libtbb-dev
sudo apt-get install libatlas-base-dev
sudo apt-get install libfaac-dev libmp3lame-dev libtheora-dev
sudo apt-get install libvorbis-dev libxvidcore-dev
sudo apt-get install libopencore-amrnb-dev libopencore-amrwb-dev
sudo apt-get install x264 v4l-utils

# Optional dependencies
sudo apt-get install libprotobuf-dev protobuf-compiler
sudo apt-get install libgoogle-glog-dev libgflags-dev
sudo apt-get install libgphoto2-dev libeigen3-dev libhdf5-dev doxygen
```

## Step 3: Install Python libraries

```
sudo apt-get install python-dev python-pip python3-dev python3-pip
sudo -H pip2 install -U pip numpy
sudo -H pip3 install -U pip numpy
```

We will use Virtual Environment to install Python libraries. It is generally a good practice in order to separate your project environment and global environment.

```

# Install virtual environment
sudo pip2 install virtualenv virtualenvwrapper
sudo pip3 install virtualenv virtualenvwrapper
echo "# Virtual Environment Wrapper" >> ~/.bashrc
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
source ~/.bashrc

##### For Python 2 #####
# create virtual environment
mkvirtualenv facecourse-py2 -p python2
workon facecourse-py2

# now install python libraries within this virtual environment
pip install numpy scipy matplotlib scikit-image scikit-learn ipython

# quit virtual environment
deactivate
#####

##### For Python 3 #####
# create virtual environment
mkvirtualenv facecourse-py3 -p python3
workon facecourse-py3

# now install python libraries within this virtual environment
pip install numpy scipy matplotlib scikit-image scikit-learn ipython

# quit virtual environment
deactivate
#####

```

## Step 4: Download opencv and opencv\_contrib

We will download opencv and opencv\_contrib packages from their github repositories.

### Step 4.1: Download opencv from Github

If you have OpenBlas installed on your machine, OpenCV 3.2.0 fails to compile. We will use the commit where this bug has been patched. So this is OpenCV 3.2.0 with few bugs patched.

```

git clone https://github.com/opencv/opencv.git
cd opencv
git checkout 2b44c0b6493726c465152e1db82cd8e65944d0db
cd ..

```

## Step 4.2: Download opencv\_contrib from Github

If we use v3.2.0 python module of opencv (cv2) fails to import due to a bug in opencv\_contrib. Here too we will use a commit where this bug has been patched.

```
git clone https://github.com/opencv/opencv_contrib.git
cd opencv_contrib
git checkout abf44fcccfe2f281b7442dac243e37b7f436d961
cd ..
```

## Step 5: Compile and install OpenCV with contrib modules

### Step 5.1 : Create a build directory

```
cd opencv
mkdir build
cd build
```

### Step 5.2 : Run CMake

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_C_EXAMPLES=ON \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D WITH_TBB=ON \
-D WITH_V4L=ON \
-D WITH_QT=ON \
-D WITH_OPENGL=ON \
-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib/modules \
-D BUILD_EXAMPLES=ON ..
```

### Step 5.3 : Compile and Install

```
# find out number of CPU cores in your machine
nproc
# substitute 4 by output of nproc
make -j4
sudo make install
sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
sudo ldconfig
```

## Step 5.4 : Create symlink in virtual environment

Depending upon Python version you have, paths would be different. OpenCV's Python binary (cv2.so) can be installed either in directory site-packages or dist-packages. Use following command to find out the correct location on your machine.

```
find /usr/local/lib/ -type f -name "cv2*.so"
```

It should output paths similar to one of these (or two in case OpenCV was compiled for both Python2 and Python3:

```
##### For Python 2 #####
## binary installed in dist-packages
/usr/local/lib/python2.6/dist-packages/cv2.so
/usr/local/lib/python2.7/dist-packages/cv2.so
## binary installed in site-packages
/usr/local/lib/python2.6/site-packages/cv2.so
/usr/local/lib/python2.7/site-packages/cv2.so

##### For Python 3 #####
## binary installed in dist-packages
/usr/local/lib/python3.5/dist-packages/cv2.cpython-35m-x86_64-linux-gnu.so
/usr/local/lib/python3.6/dist-packages/cv2.cpython-36m-x86_64-linux-gnu.so
## binary installed in site-packages
/usr/local/lib/python3.5/site-packages/cv2.cpython-35m-x86_64-linux-gnu.so
/usr/local/lib/python3.6/site-packages/cv2.cpython-36m-x86_64-linux-gnu.so
```

Double check the exact path on your machine before running the following commands.

```
##### For Python 2 #####
cd ~/.virtualenvs/facecourse-py2/lib/python2.7/site-packages
ln -s /usr/local/lib/python2.7/dist-packages/cv2.so cv2.so

##### For Python 3 #####
cd ~/.virtualenvs/facecourse-py3/lib/python3.6/site-packages
ln -s /usr/local/lib/python3.6/dist-packages/cv2.cpython-36m-x86_64-linux-gnu.so
cv2.so
```

## Step 6: Test OpenCV 3

Activate virtual environment

```
##### For Python 2 #####  
workon facecourse-py2  
  
##### For Python 3 #####  
workon facecourse-py3
```

Open IPython and check OpenCV version

```
# open ipython (run this command on terminal)  
ipython  
# import cv2 and print version (run following commands in ipython)  
import cv2  
print(cv2.__version__)  
# If OpenCV3 is installed correctly, the above command should give output 3.2.0 or  
# 3.2.0-dev  
# Press CTRL+D to exit ipython
```

Now you can exit from Python virtual environment.

```
deactivate
```

We will always use virtual environment to run our Python scripts.

## Installing Dlib on Ubuntu

### Step 1: Install OS libraries

```
sudo apt-get install build-essential cmake pkg-config  
sudo apt-get install libx11-dev libatlas-base-dev  
sudo apt-get install libgtk-3-dev libboost-python-dev
```

### Step 2: Install Python libraries

We have already completed this step while installing OpenCV in Step 3.

## Step 3: Compile DLib

### Step 3.1: Compile C++ binary

Davis King, creator of Dlib, recommends using CMake for using Dlib in your code.

But if you want to use Dlib as a library follow these steps:

```
wget http://dlib.net/files/dlib-19.4.tar.bz2
tar xvf dlib-19.4.tar.bz2
cd dlib-19.4/
mkdir build
cd build
cmake ..
cmake --build . --config Release
sudo make install
sudo ldconfig
cd ..
```

Now you can use pkg-config to provide path to Dlib's include directory and link Dlib library file.

```
pkg-config --libs --cflags dlib-1
```

### Step 3.2: Compile Python module

Now we will activate virtual environment, compile and install Dlib.


```
##### For Python 2 #####
workon facecourse-py2
# move to dlib root directory
cd dlib-19.4/

# build and install Python module
python setup.py install

# clean up. Remove any local copies of dlib.so
rm -rf dist
rm -rf tools/python/build
rm python_examples/dlib.so

# exit from virtual environment
deactivate
```





```
##### For Python 3 #####
workon facecourse-py3
# move to dlib root directory
cd dlib-19.4/

# build and install Python module
python setup.py install

# clean up. Remove any local copies of dlib.so
rm -rf dist
rm -rf tools/python/build
rm python_examples/dlib.so

# exit from virtual environment
deactivate
```

For consistency we have installed Python and C++ binaries of Dlib using same source code.

You can also install python bindings for Dlib using pip. If you are only going to use Python (and not C++), installing Dlib using pip is a good idea. It will install the latest version.

```
##### For Python 2 #####
workon facecourse-py2
pip install dlib
deactivate
```

```
##### For Python 3 #####
workon facecourse-py3
pip install dlib
deactivate
```

Now whenever you are going to run Python scripts which use Dlib and/or OpenCV you have to activate the virtual environment using workon command.

