**Installing OpenCV and Python on Raspberry PI**

**Step 1:**

Download Raspbian Jessie from [www.raspberrypi.org](http://www.raspberrypi.org) and write it into SD card using Win32 disk image writer.

**Step 2:**

Insert the SD card into the raspberry pi and boot it by applying the power

To enter the GUI interface enter the following command in the terminal

1. ***sudo startx***

**Step 3:**

To expand the file system:

Open the LXTerminal and type the following command:

1. ***sudo raspi-config***

Once prompted, you should select the first option, ***“1. Expand File System”***, ***hit Enter*** on your keyboard, arrow down to the ***“<Finish>”*** button, and then reboot your Pi:

1. ***sudo reboot***

**Step 4:**

To enable the WiFi connection type in the terminal the following:

1. ***sudo nano /etc/apt/apt.conf.d/10proxy***

A text editor will open. Type the following in the editor and save the file using Ctrl+X:

***Acquire::http::Proxy http://authproxy.serc.iisc.ernet.in:3128;***

Now you can connect your Pi to the internet using your WiFi.

**Step5:**

OpenCV, along with all its dependencies, will need a few gigabytes during the compile, so you should delete the Wolfram engine to free up some space on your Pi:

1. ***sudo apt-get purge wolfram-engine***

**Step6: Install Dependencies**

Update the system by entering the commands

1. ***sudo apt-get update***
2. ***sudo apt-get upgrade***

After that, for installing the browser iceweasel enter the following command in the terminal

1. ***sudo apt-get install iceweasel***

and set the proxy setting in the browser

We then need to install some developer tools, including [CMake](https://cmake.org/), which helps us configure the OpenCV build process:

1. ***sudo apt-get install build-essential cmake pkg-config***

Next, we need to install some image I/O packages that allow us to load various image file formats from disk. Examples of such file formats include JPEG, PNG, TIFF, etc.:

1. ***sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev***

Just as we need image I/O packages, we also need video I/O packages. These libraries allow us to read various video file formats from disk as well as work directly with video streams:

1. ***sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev***
2. ***sudo apt-get install libxvidcore-dev libx264-dev***

The OpenCV library comes with a sub-module named highgui  which is used to display images to our screen and build basic GUIs. In order to compile the highgui  module, we need to install the GTK development library:

1. ***sudo apt-get install libgtk2.0-dev***

Many operations inside of OpenCV (namely matrix operations) can be optimized further by installing a few extra dependencies:

1. ***sudo apt-get install libatlas-base-dev gfortran***

These optimization libraries are especially important for resource constrained devices such as the Raspberry Pi.

Lastly, let’s install both the Python 2.7 and Python 3 header files so we can compile OpenCV with Python bindings:

1. ***sudo apt-get install python2.7-dev python3-dev***

If you skip this step, you may notice an error related to the Python.h  header file not being found when running make  to compile OpenCV.

**Step 7:Download the OpenCV source code**

Download the OpenCV-3.1.0.zip and opencv\_contrib-3.1.0.zip files from official opencv repository through the following links:

* <https://github.com/Itseez/opencv/archive/3.1.0.zip>
* <https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip>

Unzip the opencv files by entering the following command

1. ***unzip opencv-3.1.0.zip***
2. ***unzip opencv\_contrib-3.1.0.zip***

**Step 8: Python2.7 or python3 ?**

Before we can start compiling OpenCV on our Raspberry Pi 3, we first need to install pip, a Python package manager:

1. ***sudo apt-get install python-pip python-numpy python-scipy python-matplotlib***

To install virtualenv and virtualenvwrapper

1. ***sudo apt-get install virtualenv virtualenvwrapper***
2. ***sudo rm -rf ~/.cache/pip***

Now that both virtualenv  and virtualenvwrapper  have been installed, we need to update our ~/.profile  file to include the following lines at the bottom of the file:

# virtualenv and virtualenvwrapper

export WORKON\_HOME=$HOME/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

and save ~/.profile and execute the following command

1. ***source ~/.profile***

If it shows the following error:

virtualenvwrapper.sh: no such file or directory

then, replace

source /usr/local/bin/virtualenvwrapper.sh

with

source /usr/share/virtualenvwrapper/virtualenvwrapper.sh

and re-run the command (source ~/.profile) again

***Note:*** I recommend running the *source ~/.profile*  file **each time** you open up a new terminal to ensure your system variables have been setup correctly.

**Creating Python virtual environment:**

Next, let’s create the Python virtual environment that we’ll use for computer vision development

**For python 2.7:**

1. ***mkvirtualenv cv***

**For python 3:**

1. ***python3 –m virtualenv /path to the directory***
2. ***sudo apt-get install python3.4-venv***
3. ***pyvenv-3.4 /path to the directory***
4. ***source /path to the directory/bin/activate***

In our case path to the directory is:

**/home/pi/.virtualenvs/cv**

**Step 9: How to check if you’re in “cv” virtual environment**

1. ***source ~/.profile***
2. ***workon cv***

To validate and ensure you are in the cv virtual environment, examine your command line — if you see the text *(cv)* preceding your prompt, then you ***are*** in the *cv* virtual environment:



**Step 10: Compile and install opencv**

1. ***workon cv***

Once you have ensured you are in the cv virtual environment, we can setup our build using CMake:

1. ***cd ~/opencv-3.1.0***
2. ***mkdir build***
3. ***cd build***

***33. cmake -D CMAKE\_BUILD\_TYPE=RELEASE \***

***-D INSTALL\_C\_EXAMPLES=OFF \***

***-D INSTALL\_PYTHON\_EXAMPLES=ON \***

***-D BUILD\_EXAMPLES=ON \***

***-D CMAKE\_INSTALL\_PREFIX=/usr/local \***

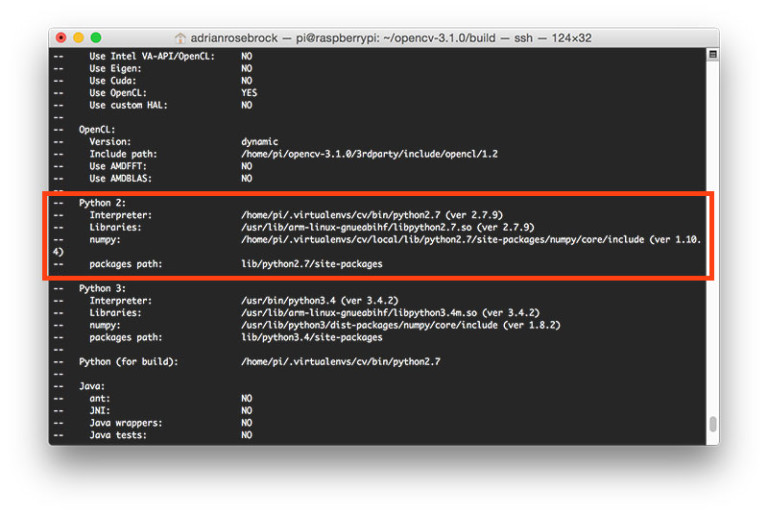
***-D WITH\_V4L=ON ..***

Now, before we move on to the actual compilation step, *make sure you examine the output of CMake!*

Start by scrolling down the section titled Python 2 and Python 3 .

*If you are compiling OpenCV 3 for Python 2.7*, then make sure your Python 2  section includes valid paths to the Interpreter , Libraries , numpy  and packages path , similar to the screenshot below

Notice how the Interpreter points to our python2.7 and python3.4 binary located in the cv virtual environment. The numpy variable also points to the NumPy installation in the cv environment as in the below image



In either case, if you *do not* see the cv virtual environment in these variables paths, *it’s almost certainly because you are NOT in the cv virtual environment prior to running CMake!*

If this is the case, access the cv  virtual environment using workon cv  and re-run the cmake  command outlined above

Now we finally ready to compile OpenCV:

NOTE: From here don’t stop the process uptill sudo ldconfig. The total process will take approx. 7 hrs.

1. ***make –j4***
2. **make clean**
3. **make**
4. **sudo make install**
5. **sudo ldconfig**

**STEP 11: Finish installing OpenCV on Pi**

If all above steps finish without errors, then follow the commands for using our opencv3 with raspberry pi 3

OpenCV should now be installed in /usr/local/lib/python2.7/site-pacakges . You can verify this using the ls command:

1. ***ls -l /usr/local/lib/python2.7/site-packages/***

if it shows total 0 then it means your files are not created in the site-packages and might be there in dist-packages. Replace site-packages with dist-packages and then you should get approximately a total of 1800-1900 with the following line

***-rw-r--r-- 1 root staff 1895772 Mar 20 20:00 cv2.so***

Our final step is to [sym-link](https://en.wikipedia.org/wiki/Symbolic_link) the OpenCV bindings into our cv  virtual environment for Python 2.7:

1. ***cd ~/.virtualenvs/cv/lib/python2.7/site-packages/***
2. ***ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so***

**For Python 3:**

After running make install, your OpenCV + Python bindings should be installed in

**/usr/local/lib/python3.4/site-packages**

Again, you can verify this with the ls command:

***ls -l /usr/local/lib/python3.4/site-packages/***

The output must be like this:

**total 1852**

**-rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so**

If not, replace site-packages with dist-packages as done previously

After that, enter the following commands to rename cv2.cpython-34m.so to cv2.so

1. ***cd /usr/local/lib/python3.4/site-packages/***
2. ***sudo mv cv2.cpython-34m.so cv2.so***

After renaming to cv2.so, we can sym-link our OpenCV bindings into the cv  virtual environment for Python 3.4

1. ***cd ~/.virtualenvs/cv/lib/python3.4/site-packages/***
2. ***ln -s /usr/local/lib/python3.4/site-packages/cv2.so cv2.so***

**Step 12: Testing your OpenCV 3 install**

***Congratulations, you now have OpenCV 3 installed on your Raspberry Pi 3 running Raspbian Jessie!***

Open up a new terminal, execute the source command, and then finally attempt to import the Python + OpenCV bindings:

**$ source ~/.profile**

**$ python**

**>>> import cv2**

**>>> cv2.\_\_version\_\_**

**'3.1.0'**

**>>>**

If the correct version is displayed, you have successfully installed OpenCV on your Raspberry Pi 3

**NOTE:**

For capturing video:

cv2.videocapture code is not working on Raspberry Pi. So, for taking the video we have to follow the following code instead.

**from picamera import PiCamera**

**from picamera.array import PiRGBArray**

**import cv2**

**camera=PiCamera()**

**camera.resolution=(640,480)**

**camera.framerate=32**

**rawCapture=PiRGBArray(camera,size=(640,480))**

**for frame in camera.capture\_continuous(rawCapture,format=”bgr”,use\_video\_port=True):**

**image=frame.array()**

**# middle program statements**

**cv2.imshow(“frame”,image)**

**rawCapture.truncate(0)**

**camera.release()**

**cv2.distroyAllWindows()**