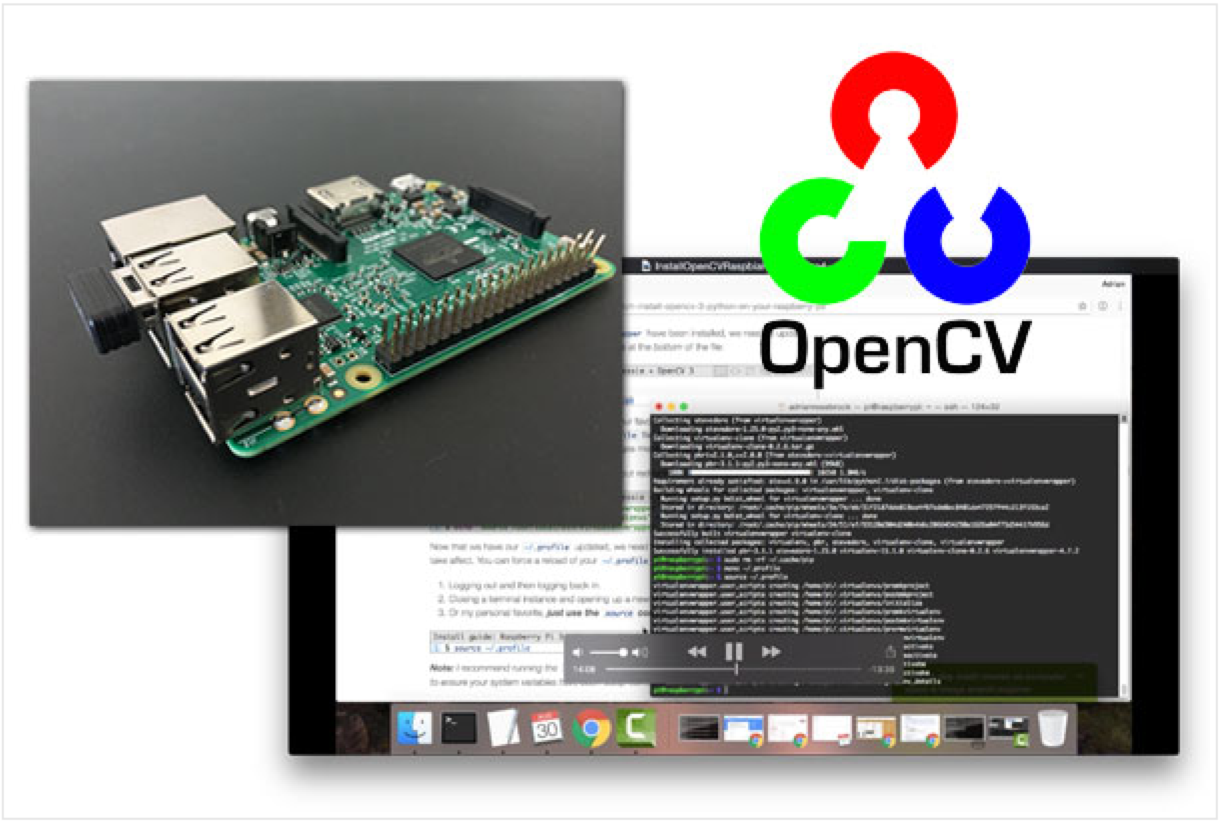
# Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi

by Adrian Rosebrock on September 4, 2017 in OpenCV 3, Raspberry Pi, Tutorials



It's been over two years since the release of Raspbian Jessie. As of August 17th, 2017, the Raspberry Pi foundation has officially released the successor to Raspbian Jessie — ***Raspbian Stretch.***

Just as I have done in previous blog posts, I'll be demonstrating **how to install OpenCV 3 with Python bindings on Raspbian *Stretch*.**

If you are looking for previous installation instructions for different platforms, please consult this list:

- Install guide: Raspberry Pi 3 + **Raspbian Jessie** + OpenCV 3
- How to install OpenCV 3.0 on ***Raspbian Jessie***.
- Installing OpenCV on your *Raspberry Pi Zero* running ***Raspbian Jessie***.
- Installing OpenCV 3.0 for both Python 2.7 and Python 3+ on ***Raspbian Wheezy*.**
- Install OpenCV 2.4 for Python 2.7 on ***Raspbian Wheezy***.

Otherwise, let's proceed with getting OpenCV 3 with Python bindings installed on Raspian Stretch!

## The quick start video tutorial

If this is your first time installing OpenCV or you are just getting started with Linux I *highly suggest* that you watch the video below and follow along with me as you guide you step-by-step on how to install OpenCV 3 on your Raspberry Pi running Raspbian Stretch:

Otherwise, if you feel comfortable using the command line or if you have previous experience with Linux environments, feel free to use the text-based version of this guide below.

## Assumptions

In this tutorial, I am going to assume that you already own a **Raspberry Pi 3** with **Raspbian Stretch installed**.

If you don't already have the Raspbian Stretch OS, you'll need to upgrade your OS to take advantage of Raspbian Stretch's new features.

To upgrade your Raspberry Pi 3 to Raspbian Stretch, you may download it here and follow these upgrade instructions (or these for the NOOBS route which is recommended for beginners). The former instructions take approximately 10 minutes to download via a torrent client and about 10 minutes to flash the SD card at which point you can power up and proceed to the next section.

***Note:*** *If you are upgrading your Raspberry Pi 3 from Raspbian Jessie to Raspbian Stretch, there is the potential for problems.* ***Proceed at your own risk****, and consult the Raspberry Pi forums for help.*

***Important:*** *It is my recommendation that you proceed with a* ***fresh install of Raspbian Stretch!*** *Upgrading from Raspbian Jessie is* ***not****recommended.*

Assuming that your OS is up to date, you'll need one of the following for the remainder of this post:

- *Physical access* to your Raspberry Pi 3 so that you can open up a terminal and execute commands
- *Remote access* via SSH or VNC.

I'll be doing the majority of this tutorial via SSH, but as long as you have access to a terminal, you can easily follow along.

***Can't SSH?*** If you see your Pi on your network, but can't ssh to it, you may need to enable SSH. This can easily be done via the Raspberry Pi desktop preferences menu (you'll need an HDMI cable and a keyboard/mouse) or running `sudo service ssh start` from the command line of your Pi.

After you've changed the setting and rebooted, you can test SSH directly on the Pi with the localhost address. Open a terminal and type `sshpi@127.0.0.1` to see if it is working.

***Keyboard layout giving you problems?*** Change your keyboard layout by going to the Raspberry Pi desktop preferences menu. I use the standard US Keyboard layout, but you'll want to select the one appropriate for your keyboard or desire (any Dvorkac users out there?).

## Installing OpenCV 3 on a Raspberry Pi 3 running Raspbian Stretch

If you've ever installed OpenCV on a Raspberry Pi (or any other platform before), you know that the process can be quite time consuming with many dependencies and pre-requisites that have to be installed. **The goal of this tutorial is to thus guide you step-by-step through the compile and installation process.**

In order to make the installation process go more smoothly, I've included timings for each step so you know when to take a break, grab a cup of coffee, and checkup on email while the Pi compiles OpenCV.

Let's go ahead and get started installing OpenCV 3 on your Raspberry Pi 3 running Raspbian Stretch.

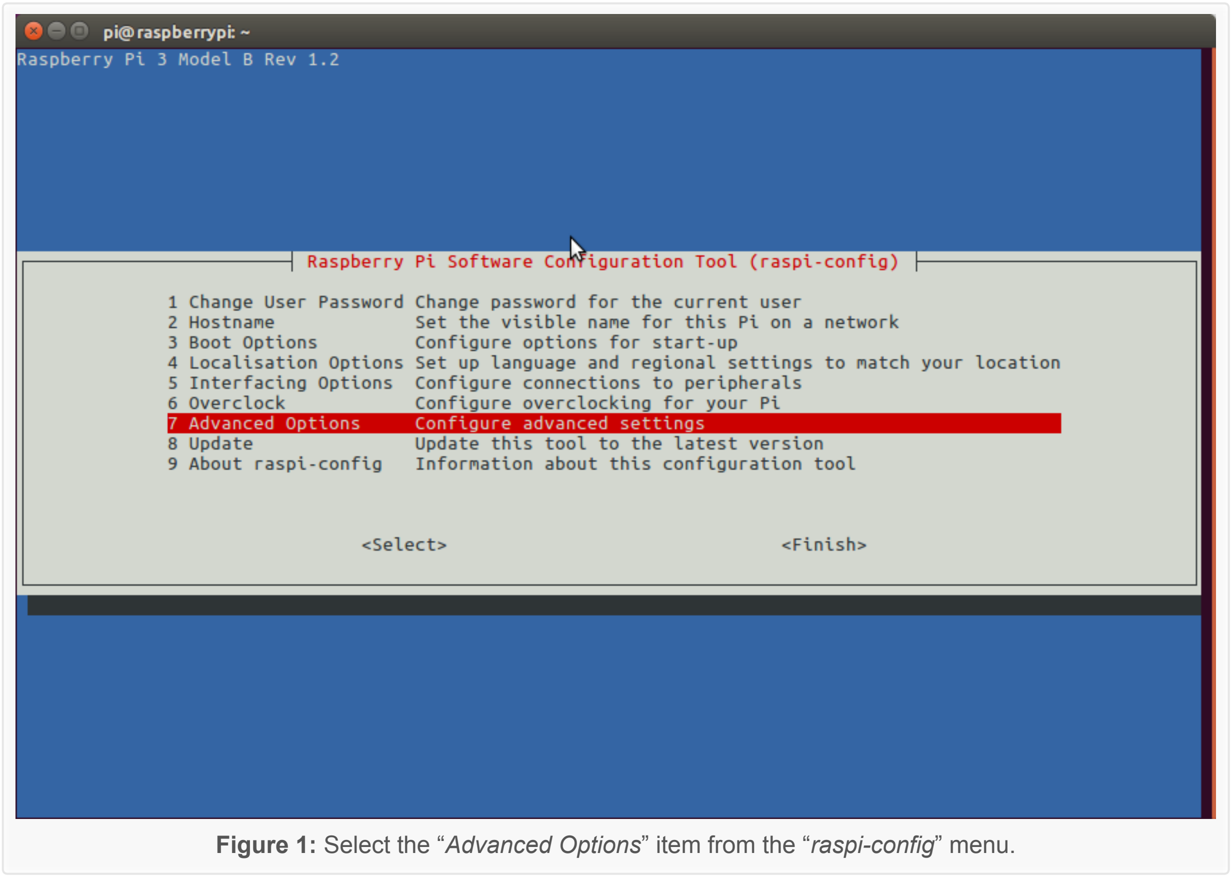
### Step #1: Expand filesystem

Are you using a *brand new* install of Raspbian Stretch?

If so, the first thing you should do is expand your filesystem to include *all available space* on your micro-SD card:

```
Raspbian Stretch: Install OpenCV 3 + Python o                    Shell
$ sudo raspi-config
```

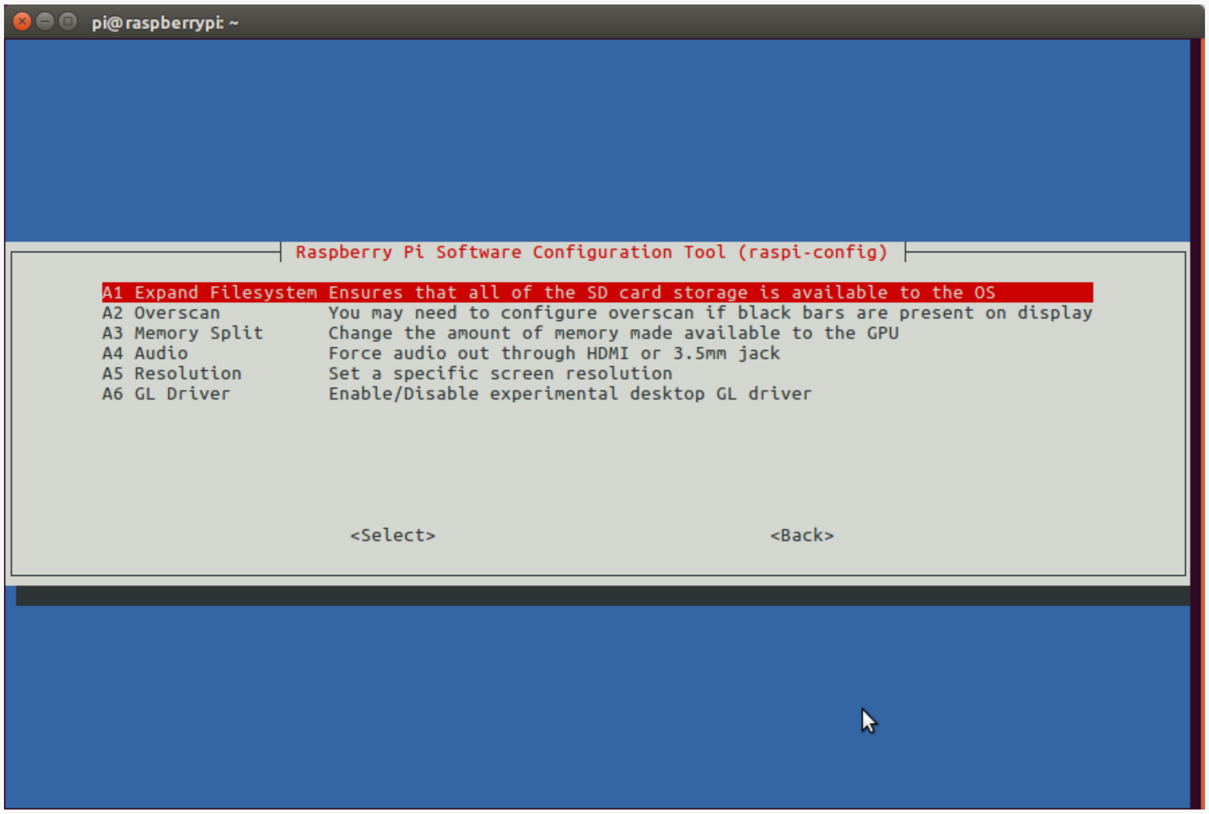
And then select the "Advanced Options" menu item:



```
Raspberry Pi 3 Model B Rev 1.2

         Raspberry Pi Software Configuration Tool (raspi-config)

   1 Change User Password Change password for the current user
   2 Hostname             Set the visible name for this Pi on a network
   3 Boot Options         Configure options for start-up
   4 Localisation Options Set up language and regional settings to match your location
   5 Interfacing Options  Configure connections to peripherals
   6 Overclock            Configure overclocking for your Pi
   7 Advanced Options     Configure advanced settings
   8 Update               Update this tool to the latest version
   9 About raspi-config   Information about this configuration tool

                 <Select>                              <Finish>
```

**Figure 1:** Select the "*Advanced Options*" item from the "*raspi-config*" menu.

Followed by selecting "Expand filesystem":



```
Raspberry Pi Software Configuration Tool (raspi-config)

A1 Expand Filesystem Ensures that all of the SD card storage is available to the OS
A2 Overscan          You may need to configure overscan if black bars are present on display
A3 Memory Split      Change the amount of memory made available to the GPU
A4 Audio             Force audio out through HDMI or 3.5mm jack
A5 Resolution        Set a specific screen resolution
A6 GL Driver         Enable/Disable experimental desktop GL driver
```

**Figure 2:** Expanding the filesystem on your Raspberry Pi 3.

Once prompted, you should select the first option, ***"A1. Expand File System"***, ***hit Enter*** on your keyboard, arrow down to the ***"<Finish>"***button, and then reboot your Pi — you may be prompted to reboot, but if you aren't you can execute:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ sudo reboot
```

After rebooting, your file system should have been expanded to include all available space on your micro-SD card. You can verify that the disk has been expanded by executing `df -h` and examining the output:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        30G  4.2G   24G  15% /
devtmpfs        434M     0  434M   0% /dev
tmpfs           438M     0  438M   0% /dev/shm
tmpfs           438M   12M  427M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           438M     0  438M   0% /sys/fs/cgroup
/dev/mmcblk0p1   42M   21M   21M  51% /boot
tmpfs            88M     0   88M   0% /run/user/1000
```

As you can see, my Raspbian filesystem has been expanded to include all 32GB of the micro-SD card.

However, even with my filesystem expanded, I have already used 15% of my 32GB card.

If you are using an 8GB card you may be using close to 50% of the available space, so one simple thing to do is to delete both LibreOffice and Wolfram engine to free up some space on your Pi:

```
$ sudo apt-get purge wolfram-engine
$ sudo apt-get purge libreoffice*
$ sudo apt-get clean
$ sudo apt-get autoremove
```

**After removing the Wolfram Engine and LibreOffice, *you can reclaim almost 1GB!***

## Step #2: Install dependencies

This isn't the first time I've discussed how to install OpenCV on the Raspberry Pi, so I'll keep these instructions on the briefer side, allowing you to work through the installation process: I've also included the ***amount of time it takes to execute each command*** (some depend on your Internet speed) so you can plan your OpenCV + Raspberry Pi 3 install accordingly (OpenCV itself takes approximately **4 hours to compile** — more on this later).

The first step is to update and upgrade any existing packages:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ sudo apt-get update && sudo apt-get upgrade
```

**Timing: 2m 14s**

We then need to install some developer tools, including CMake, which helps us configure the OpenCV build process:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ sudo apt-get install build-essential cmake pkg-config
```

**Timing: 19s**

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.:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Python
$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev lib
```

**Timing: 21s**

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:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-
$ sudo apt-get install libxvidcore-dev libx264-dev
```

**Timing: 32s**

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:

```
$ sudo apt-get install libgtk2.0-dev libgtk-3-dev
```

**Timing: 1m 36s**

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

```
Raspbian Stretch: Install OpenCV 3 + Python                Shell
$ sudo apt-get install libatlas-base-dev gfortran
```

**Timing: 23s**

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:

```
Raspbian Stretch: Install OpenCV 3 + Python                Shell
$ sudo apt-get install python2.7-dev python3-dev
```

**Timing: 45s**

If you're working with a fresh install of the OS, it is possible that these versions of Python are already at the newest version (you'll see a terminal message stating this).

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 #3: Download the OpenCV source code

Now that we have our dependencies installed, let's grab the `3.3.0` archive of OpenCV from the official OpenCV repository. This version includes the `dnn` module which we discussed in a previous post where we did Deep Learning with OpenCV (***Note:*** As future versions of openCV are released, you can replace `3.3.0` with the latest version number):

```
Raspbian Stretch: Install OpenCV 3 + Python                Shell
$ cd ~
$ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3
$ unzip opencv.zip
```

**Timing: 41s**

We'll want the *full install* of OpenCV 3 (to have access to features such as SIFT and SURF, for instance), so we also need to grab the opencv_contrib repository as well:

```
Raspbian Stretch: Install OpenCV 3 + Pyth                  Shell
$ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_con
$ unzip opencv_contrib.zip
```

**Timing: 37s**

You might need to expand the command above using the "<=>" button during your copy and paste. The `.zip` in the `3.3.0.zip` may appear to be cutoff in some browsers. The full URL of the OpenCV 3.3.0 archive is:

https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip

***Note:*** *Make sure your* `opencv` *and* `opencv_contrib` *versions are the same (in this case,* `3.3.0`*). If the versions numbers do not match up, then you'll likely run into either compile-time or runtime errors.*

## Step #4: Python 2.7 or Python 3?

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
$ sudo python3 get-pip.py
```

**Timing: 33s**

You may get a message that pip is already up to date when issuing these commands, but it is best not to skip this step.

If you're a longtime PyImageSearch reader, then you'll know that I'm a *huge fan* of both virtualenv and virtualenvwrapper. Installing these packages is not a requirement and you can *absolutely* get OpenCV installed without them, but that said, ***I highly recommend you install them*** as other existing PyImageSearch tutorials (as well as future tutorials) also leverage Python virtual environments. I'll also be assuming that you have both `virtualenv` and `virtualenvwrapper` installed throughout the remainder of this guide.

**So, given that, what's the point of using `virtualenv` and `virtualenvwrapper` ?**

First, it's important to understand that a virtual environment is a *special tool* used to keep the dependencies required by different projects in separate places by creating *isolated, independent* Python environments for each of them.

In short, it solves the *"Project X depends on version 1.x, but Project Y needs 4.x"* dilemma. It also keeps your global `site-packages` neat, tidy, and free from clutter.

If you would like a full explanation on why Python virtual environments are good practice, absolutely give this excellent blog post on RealPython a read.

It's ***standard practice*** in the Python community to be using virtual environments of some sort, so I *highly recommend* that you do the same:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/.cache/pip
```

**Timing: 35s**

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:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

In previous tutorials, I've recommended using your favorite terminal-based text editor such as `vim` , `emacs` , or `nano` to update the `~/.profile` file. If you're comfortable with these editors, go ahead and update the file to reflect the changes mentioned above.

Otherwise, you should simply use `cat` and output redirection to handle updating `~/.profile` :

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

Now that we have our `~/.profile` updated, we need to reload it to make sure the changes take affect. You can force a reload of your `~/.profile` file by:

1. Logging out and then logging back in.
2. Closing a terminal instance and opening up a new one
3. Or my personal favorite, ***just use the*** `source` ***command:***

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ source ~/.profile
```

***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 your Python virtual environment

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ mkvirtualenv cv -p python2
```

This command will create a new Python virtual environment named `cv` using ***Python 2.7***.

If you instead want to use ***Python 3***, you'll want to use this command instead:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ mkvirtualenv cv -p python3
```

**Timing: 24s**

Again, ***I can't stress this point enough:*** the `cv` Python virtual environment is ***entirely independent and sequestered*** from the default Python version included in the download of Raspbian Stretch. Any Python packages in the *global* `site-packages` directory *will not* be available to the `cv` virtual environment. Similarly, any Python packages installed in `site-packages` of `cv` *will not* be available to the global install of Python. Keep this in mind when you're working in your Python virtual environment and it will help avoid a lot of confusion and headaches.

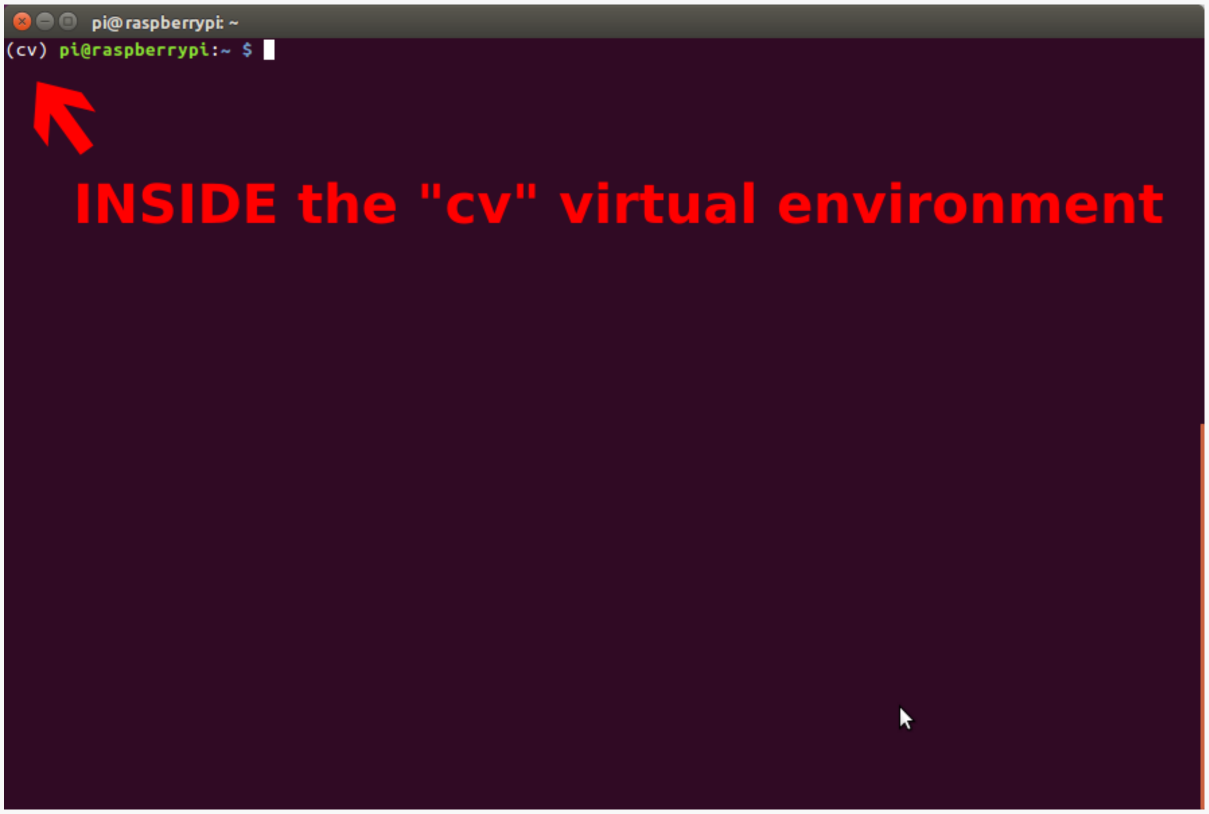
### How to check if you're in the "cv" virtual environment

If you ever reboot your Raspberry Pi; log out and log back in; or open up a new terminal, you'll need to use the `workon` command to re-access the `cv` virtual environment. In previous blog posts, I've seen readers use the `mkvirtualenv` command — ***this is entirely unneeded!*** The `mkvirtualenv` command is meant to be executed only once: to actually *create* the virtual environment.

After that, you can use `workon` and you'll be dropped down into your virtual environment:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi          Shell
$ source ~/.profile
$ workon cv
```

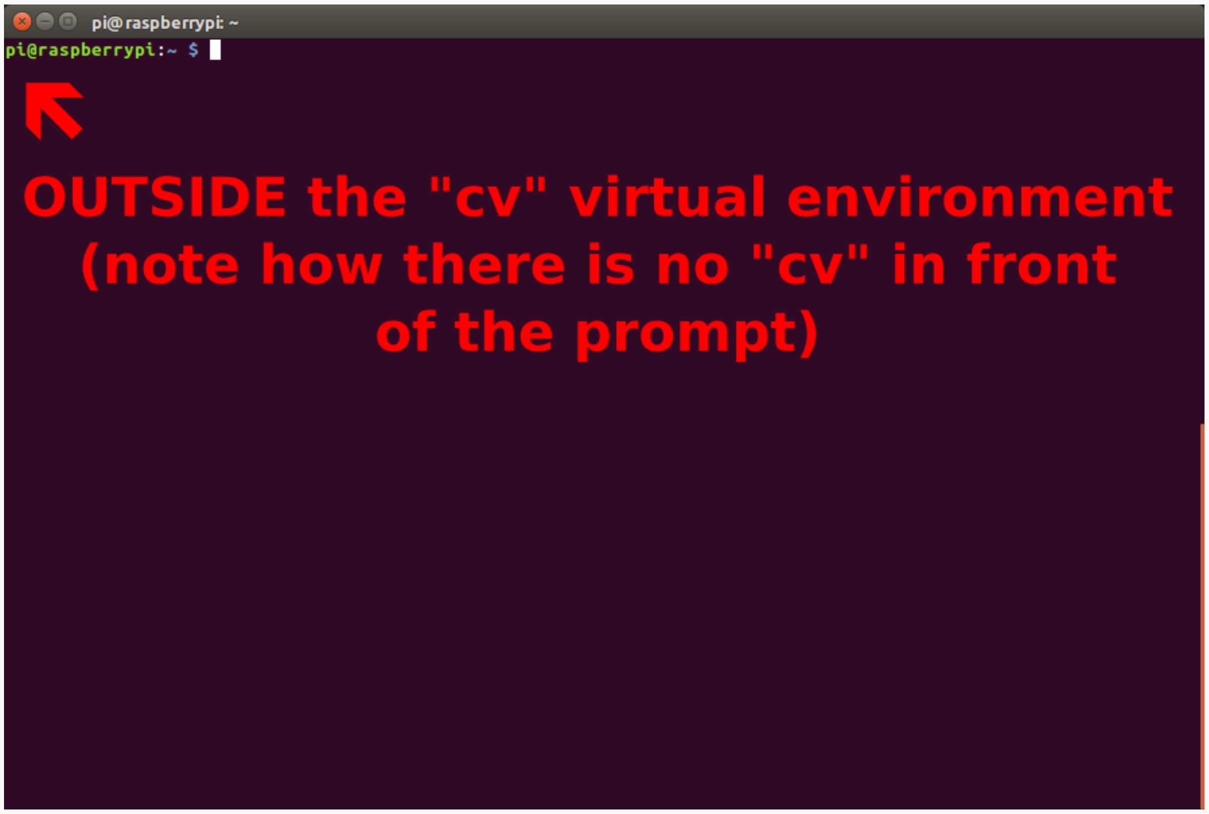
To validate and ensure you are in the `cv` virtual environment, examine your command line — *if you see the text* `(cv)` *preceding your prompt, then you **are** in the* `cv` *virtual environment:*



```
(cv) pi@raspberrypi:~ $
```

**Figure 3:** Make sure you see the "*(cv)*" text on your prompt, indicating that you ***are*** in the cv virtual environment.

Otherwise, if you ***do not*** see the `(cv)` text, then you ***are not*** in the `cv` virtual environment:



```
pi@raspberrypi:~ $
```

pi@raspberrypi: ~
OUTSIDE the "cv" virtual environment (note how there is no "cv" in front of the prompt)

**Figure 4:** If you do not see the "*(cv)*" text on your prompt, then you ***are not*** in the cv virtual environment and need to run "*source*" and "*workon*" to resolve this issue.

To fix this, simply execute the `source` and `workon` commands mentioned above.

## Installing NumPy on your Raspberry Pi

Assuming you've made it this far, you should now be in the `cv` virtual environment (which you should stay in for the rest of this tutorial). Our only Python dependency is NumPy, a Python package used for numerical processing:

```
Raspbian Stretch: Install OpenCV 3 + Python                    Shell
$ pip install numpy
```

**Timing: 11m 12s**

Be sure to grab a cup of coffee or go for a nice walk, the NumPy installation can take a bit of time.

***Note:*** *A question I've often seen is **"Help, my NumPy installation has hung and it's not installing!"** Actually, it is installing, it just takes time to pull down the sources and compile. You can verify that NumPy is compiling and installing by running* `top` *. Here you'll see that your CPU cycles are being used compiling NumPy. Be patient. The Raspberry Pi isn't as fast as your laptop/desktop.*

## Step #5: Compile and Install OpenCV

We are now ready to compile and install OpenCV! Double-check that you are in the `cv` virtual environment by examining your prompt (you should see the `(cv)` text preceding it), and if not, simply execute `workon` :

```
$ workon cv
```

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

Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi Shell

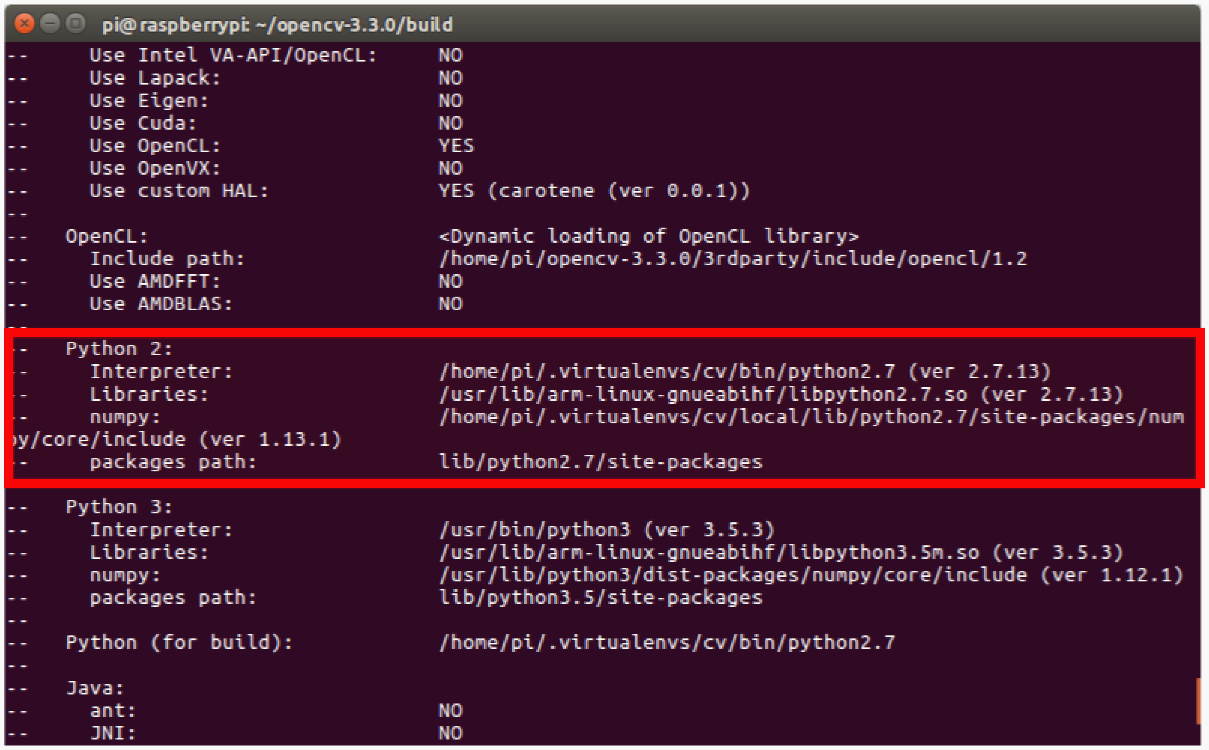
```
$ cd ~/opencv-3.3.0/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

**Timing: 2m 56s**

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 my screenshot below:

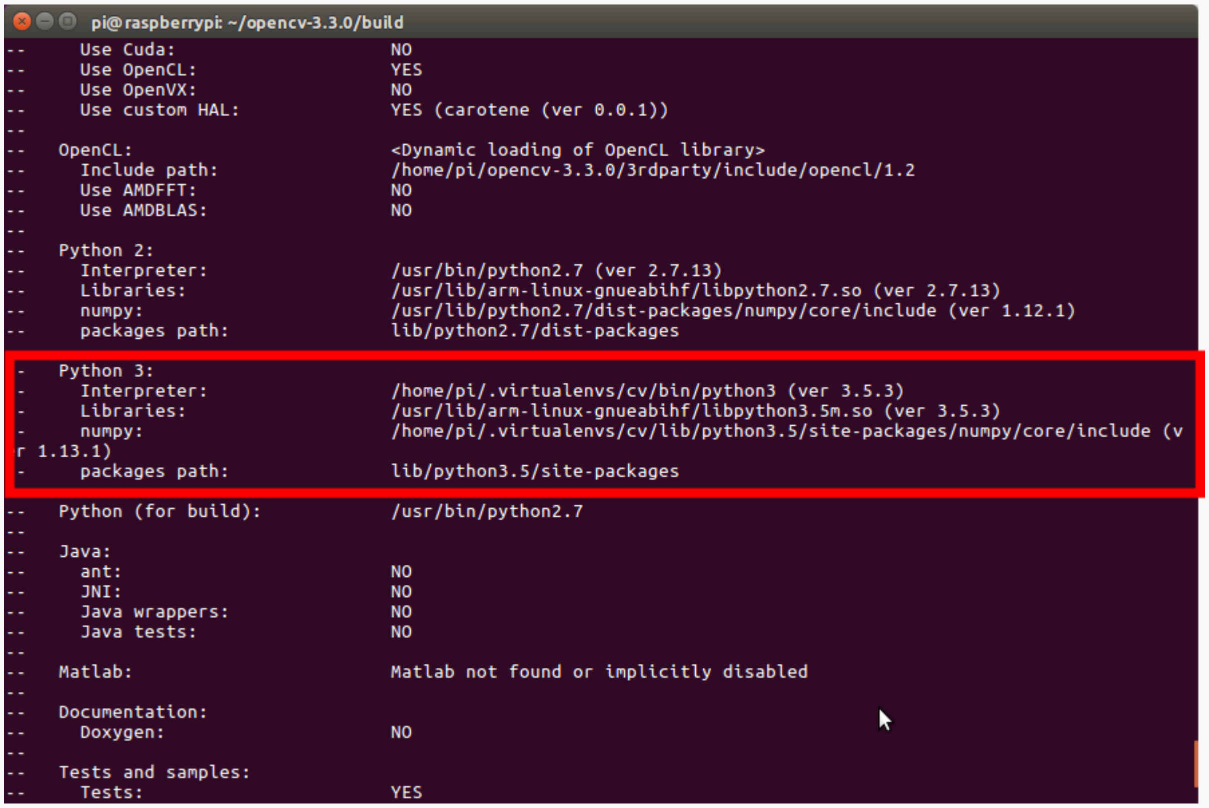


```
pi@raspberrypi: ~/opencv-3.3.0/build
--     Use Intel VA-API/OpenCL:     NO
--     Use Lapack:                  NO
--     Use Eigen:                   NO
--     Use Cuda:                    NO
--     Use OpenCL:                  YES
--     Use OpenVX:                  NO
--     Use custom HAL:              YES (carotene (ver 0.0.1))
--
--   OpenCL:                        <Dynamic loading of OpenCL library>
--     Include path:                /home/pi/opencv-3.3.0/3rdparty/include/opencl/1.2
--     Use AMDFFT:                  NO
--     Use AMDBLAS:                 NO
--
--   Python 2:
--     Interpreter:                 /home/pi/.virtualenvs/cv/bin/python2.7 (ver 2.7.13)
--     Libraries:                   /usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.13)
--     numpy:                       /home/pi/.virtualenvs/cv/local/lib/python2.7/site-packages/num
py/core/include (ver 1.13.1)
--     packages path:               lib/python2.7/site-packages
--
--   Python 3:
--     Interpreter:                 /usr/bin/python3 (ver 3.5.3)
--     Libraries:                   /usr/lib/arm-linux-gnueabihf/libpython3.5m.so (ver 3.5.3)
--     numpy:                       /usr/lib/python3/dist-packages/numpy/core/include (ver 1.12.1)
--     packages path:               lib/python3.5/site-packages
--
--   Python (for build):            /home/pi/.virtualenvs/cv/bin/python2.7
--
--   Java:
--     ant:                         NO
--     JNI:                         NO
```

**Figure 6:** Checking that Python 3 will be used when compiling OpenCV 3 for Raspbian Stretch on the Raspberry Pi 3.

Notice how the `Interpreter` points to our `python2.7` binary located in the `cv` virtual environment. The `numpy` variable also points to the NumPy installation in the `cv` environment.

Similarly, ***if you're compiling OpenCV for Python 3***, make sure the `Python 3` section looks like the figure below:



```
pi@raspberrypi: ~/opencv-3.3.0/build
--     Use Cuda:                    NO
--     Use OpenCL:                  YES
--     Use OpenVX:                  NO
--     Use custom HAL:              YES (carotene (ver 0.0.1))
--
--   OpenCL:                        <Dynamic loading of OpenCL library>
--     Include path:                /home/pi/opencv-3.3.0/3rdparty/include/opencl/1.2
--     Use AMDFFT:                  NO
--     Use AMDBLAS:                 NO
--
--   Python 2:
--     Interpreter:                 /usr/bin/python2.7 (ver 2.7.13)
--     Libraries:                   /usr/lib/arm-linux-gnueabihf/libpython2.7.so (ver 2.7.13)
--     numpy:                       /usr/lib/python2.7/dist-packages/numpy/core/include (ver 1.12.1)
--     packages path:               lib/python2.7/dist-packages
-
-   Python 3:
-     Interpreter:                 /home/pi/.virtualenvs/cv/bin/python3 (ver 3.5.3)
-     Libraries:                   /usr/lib/arm-linux-gnueabihf/libpython3.5m.so (ver 3.5.3)
-     numpy:                       /home/pi/.virtualenvs/cv/lib/python3.5/site-packages/numpy/core/include (v
r 1.13.1)
-     packages path:               lib/python3.5/site-packages
-
--   Python (for build):            /usr/bin/python2.7
--
--   Java:
--     ant:                         NO
--     JNI:                         NO
--     Java wrappers:               NO
--     Java tests:                  NO
--
--   Matlab:                        Matlab not found or implicitly disabled
--
--   Documentation:
--     Doxygen:                     NO
--
--   Tests and samples:
--     Tests:                       YES
```

**Figure 6:** Checking that Python 3 will be used when compiling OpenCV 3 for Raspbian Stretch on the Raspberry Pi 3.

Again, the `Interpreter` points to our `python3.5` binary located in the `cv` virtual environment while `numpy` points to our NumPy install.

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.

Finally, we are now ready to compile OpenCV:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ make
```

**Timing: 4h 0m**

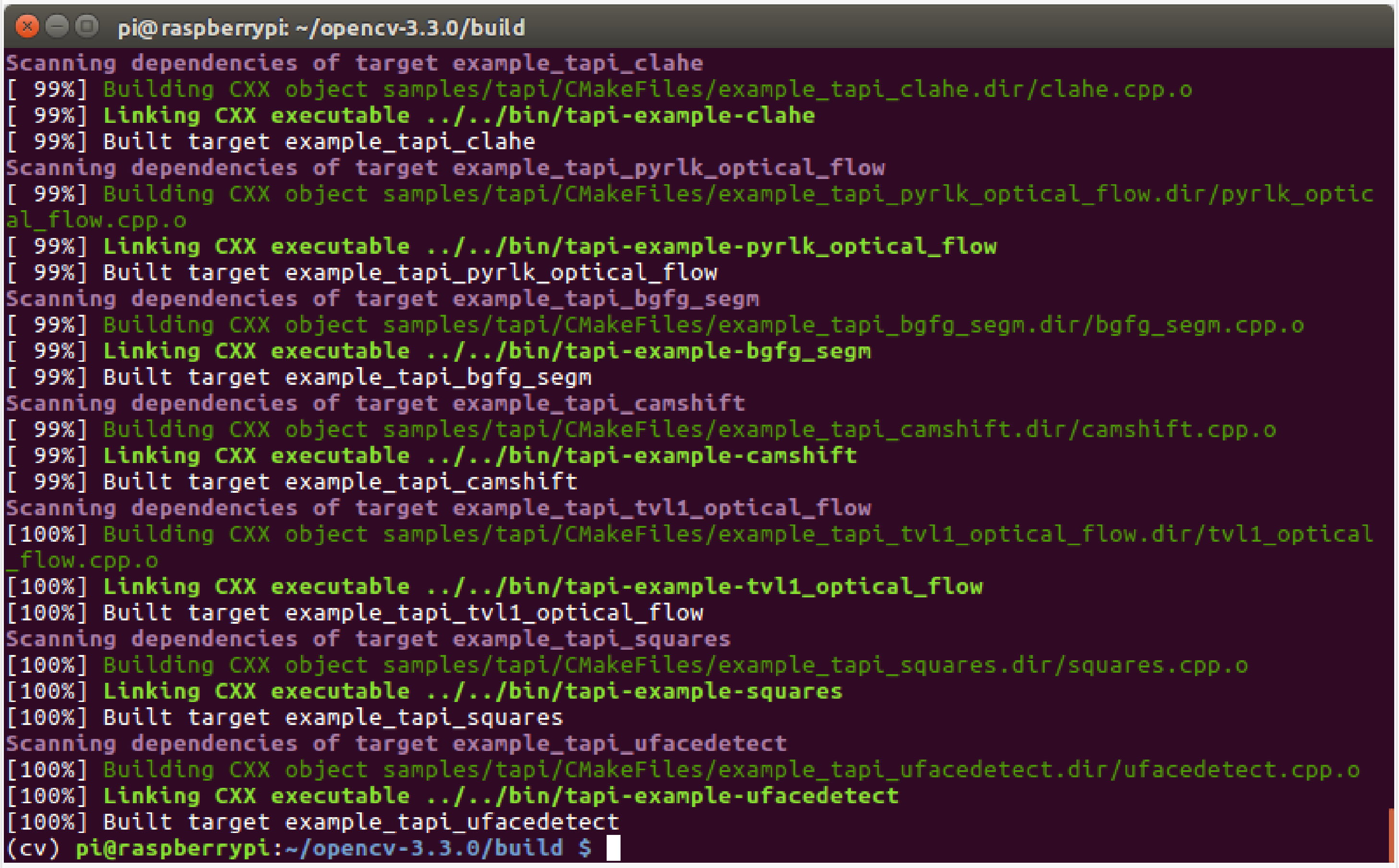
I was not able to successfully leverage compiling with 2 or 4 cores (the Pi 3 has 4 cores) which would *significantly* cut down on compile time (previous posts here on PyImageSearch demonstrate that OpenCV and be compiled in a little over 60 minutes).

I recommend you **not** compile with `make -j2` or `make -j4` as your compile will likely freeze up at 90%. I have not been able to diagnose whether this is an issue with (1) the Pi overheating, (2) a race condition with the compile, (3) the Raspbian Stretch OS, or some combination of all three. Based on all my experimentations thus far I think it's an issue introduced with the Raspbian Stretch OS.

If you do have a compile error using `-j2` or `-j4`, I suggest starting the compilation over again and using only one core:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ make clean
$ make
```

Once OpenCV 3 has finished compiling, your output should look similar to mine below:



```
pi@raspberrypi: ~/opencv-3.3.0/build
Scanning dependencies of target example_tapi_clahe
[ 99%] Building CXX object samples/tapi/CMakeFiles/example_tapi_clahe.dir/clahe.cpp.o
[ 99%] Linking CXX executable ../../bin/tapi-example-clahe
[ 99%] Built target example_tapi_clahe
Scanning dependencies of target example_tapi_pyrlk_optical_flow
[ 99%] Building CXX object samples/tapi/CMakeFiles/example_tapi_pyrlk_optical_flow.dir/pyrlk_optic
al_flow.cpp.o
[ 99%] Linking CXX executable ../../bin/tapi-example-pyrlk_optical_flow
[ 99%] Built target example_tapi_pyrlk_optical_flow
Scanning dependencies of target example_tapi_bgfg_segm
[ 99%] Building CXX object samples/tapi/CMakeFiles/example_tapi_bgfg_segm.dir/bgfg_segm.cpp.o
[ 99%] Linking CXX executable ../../bin/tapi-example-bgfg_segm
[ 99%] Built target example_tapi_bgfg_segm
Scanning dependencies of target example_tapi_camshift
[ 99%] Building CXX object samples/tapi/CMakeFiles/example_tapi_camshift.dir/camshift.cpp.o
[ 99%] Linking CXX executable ../../bin/tapi-example-camshift
[ 99%] Built target example_tapi_camshift
Scanning dependencies of target example_tapi_tvl1_optical_flow
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_tvl1_optical_flow.dir/tvl1_optical
_flow.cpp.o
[100%] Linking CXX executable ../../bin/tapi-example-tvl1_optical_flow
[100%] Built target example_tapi_tvl1_optical_flow
Scanning dependencies of target example_tapi_squares
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_squares.dir/squares.cpp.o
[100%] Linking CXX executable ../../bin/tapi-example-squares
[100%] Built target example_tapi_squares
Scanning dependencies of target example_tapi_ufacedetect
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_ufacedetect.dir/ufacedetect.cpp.o
[100%] Linking CXX executable ../../bin/tapi-example-ufacedetect
[100%] Built target example_tapi_ufacedetect
(cv) pi@raspberrypi:~/opencv-3.3.0/build $
```

**Figure 7:** Our OpenCV 3 compile on Raspbian Stretch has completed successfully.

From there, all you need to do is install OpenCV 3 on your Raspberry Pi 3:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ sudo make install
$ sudo ldconfig
```

**Timing: 52s**

## Step #6: Finish installing OpenCV on your Pi

We're almost done — just a few more steps to go and you'll be ready to use your Raspberry Pi 3 with OpenCV 3 on Raspbian Stretch.

### For Python 2.7:

Provided your **Step #5** finished without error, OpenCV should now be installed in `/usr/local/lib/python2.7/site-pacakges` . You can verify this using the `ls` command:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ ls -l /usr/local/lib/python2.7/site-packages/
total 1852
-rw-r--r-- 1 root staff 1895772 Mar 20 20:00 cv2.so
```

***Note:** In some cases, OpenCV can be installed in `/usr/local/lib/python2.7/dist-packages` (note the `dist-packages` rather than `site-packages` ). If you do not find the `cv2.so` bindings in `site-packages` , we be sure to check `dist-packages` .*

Our final step is to sym-link the OpenCV bindings into our `cv` virtual environment for Python 2.7:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
$ 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/python3.5/site-packages` . Again, you can verify this with the `ls` command:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ ls -l /usr/local/lib/python3.5/site-packages/
total 1852
-rw-r--r-- 1 root staff 1895932 Mar 20 21:51 cv2.cpython-34m.so
```

I honestly don't know why, perhaps it's a bug in the CMake script, but when compiling OpenCV 3 bindings for Python 3+, the output `.so` file is named `cv2.cpython-35m-arm-linux-gnueabihf.so` (or some variant of) rather than simply `cv2.so` (like in the Python 2.7 bindings).

Again, I'm not sure exactly *why* this happens, but it's an easy fix. All we need to do is rename the file:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ cd /usr/local/lib/python3.5/site-packages/
$ sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so
```

After renaming to `cv2.so` , we can sym-link our OpenCV bindings into the `cv` virtual environment for Python 3.5:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ cd ~/.virtualenvs/cv/lib/python3.5/site-packages/
$ ln -s /usr/local/lib/python3.5/site-packages/cv2.so cv2.so
```

## Step #7: Testing your OpenCV 3 install

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

But before we pop the champagne and get drunk on our victory, let's first verify that your OpenCV installation is working properly.

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

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ source ~/.profile
$ workon cv
$ python
>>> import cv2
>>> cv2.__version__
'3.3.0'
>>>
```

As you can see from the screenshot of my own terminal, **OpenCV 3 has been successfully installed on my Raspberry Pi 3 + Python 3.5 environment:**

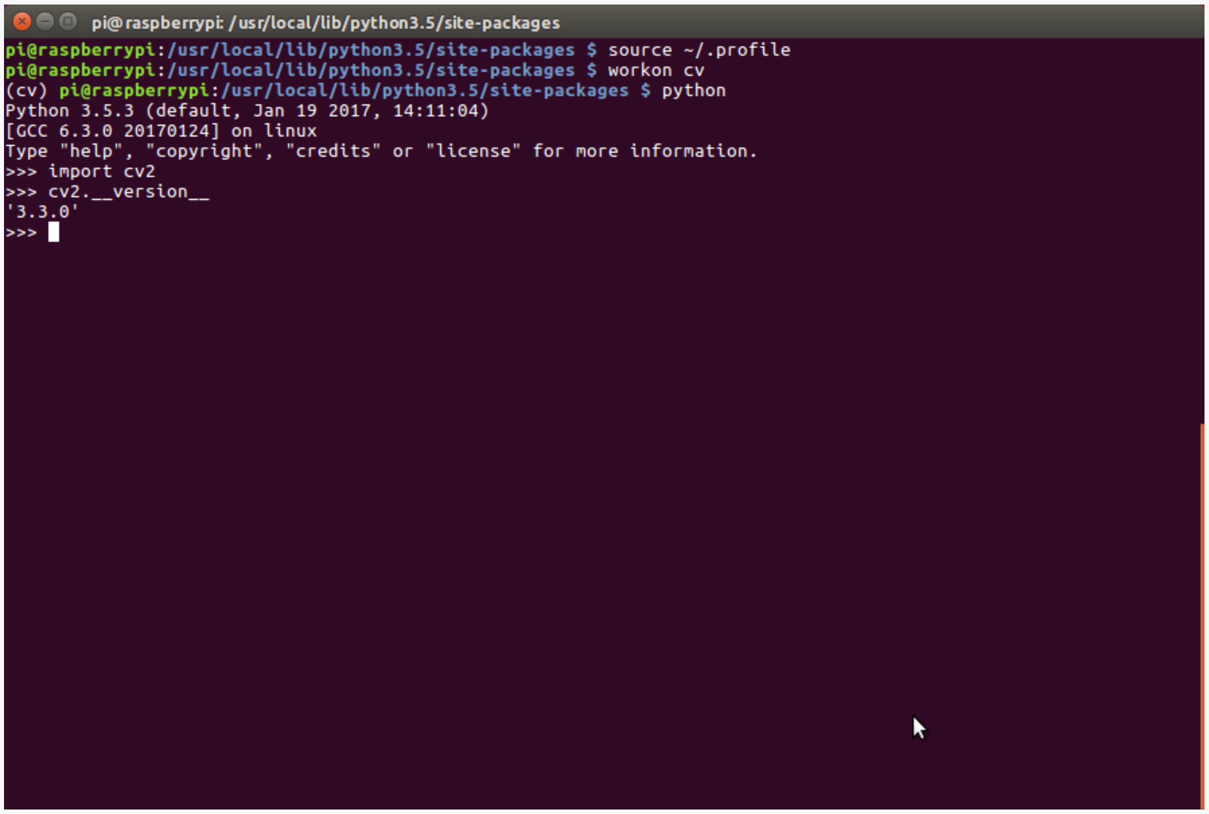

```
pi@raspberrypi: /usr/local/lib/python3.5/site-packages
pi@raspberrypi:/usr/local/lib/python3.5/site-packages $ source ~/.profile
pi@raspberrypi:/usr/local/lib/python3.5/site-packages $ workon cv
(cv) pi@raspberrypi:/usr/local/lib/python3.5/site-packages $ python
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.3.0'
>>>
```

**Figure 8:** Confirming OpenCV 3 has been successfully installed on my Raspberry Pi 3 running Raspbian Stretch.

Once OpenCV has been installed, you can remove both the `opencv-3.3.0` and `opencv_contrib-3.3.0` directories to free up a bunch of space on your disk:

```
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi                Shell
$ rm -rf opencv-3.3.0 opencv_contrib-3.3.0
```

However, be cautious with this command! Make sure OpenCV has been properly installed on your system before blowing away these directories. A mistake here could cost you ***hours*** in compile time.

## Troubleshooting and FAQ

***Q.*** When I try to execute `mkvirtualenv` and `workon` , I get a “command not found error”.

***A.*** There are three reasons why this could be happening, all of them related to **Step #4**:

1. Make certain that you have installed `virtualenv` and `virtualenvwrapper` via `pip` . You can check this by running `pipfreeze` and then examining the output, ensuring you see occurrences of both `virtualenv` and `virtualenvwrapper` .
2. You might not have updated your `~/.profile` correctly. Use a text editor such as `nano` to view your `~/.profile` file and ensure that the proper `export` and `source` commands are present (again, check **Step #4** for the contents that should be appended to `~/.profile` .
3. You did not `source` your `~/.profile` after editing it, rebooting, opening a new terminal, etc. Any time you open a new terminal and want to use a virtual environment, make sure you execute `source~/.profile` to load the contents — this will give you access to the `mkvirtualenv` and `workon` commands.

***Q.*** After I open a new terminal, logout, or reboot my Pi, I cannot execute `mkvirtualenv` or `workon` .

***A.*** See **reason #3** from the previous question.

***Q.*** When I (1) open up a Python shell that imports OpenCV or (2) execute a Python script that calls OpenCV, I get an error: `ImportError: No modulenamed cv2` .

***A.*** Unfortunately, this error is extremely hard to diagnose, mainly because there are multiple issues that could be causing the problem. To start, make sure you are in the `cv` virtual environment by using `workon cv` . If the `workon` command fails, then see the first question in this FAQ. If you're *still* getting an error, investigate the contents of the `site-packages` directory for your `cv` virtual environment. You can find the `site-packages` directory in `~/.virtualenvs/cv/lib/python2.7/site-packages/` or `~/.virtualenvs/cv/lib/python3.5/site-packages/` (depending on which Python version you used for the install). Make sure that your sym-link to the `cv2.so` file is valid and points to an existing file.

***Q.*** I'm running into other errors.

***A.*** Feel free to leave a comment and I'll try to provide guidance; however, please understand that without physical access to your Pi it can often be hard to diagnose compile/install errors. If you're in a rush to get OpenCV up and running on your Raspberry Pi be sure to take a look at the *Quickstart Bundle* and *Hardcopy Bundle* of my book, *Practical Python and OpenCV*. Both of these bundles include a Raspbian .img file with OpenCV pre-configured and pre-installed. Simply download the .img file, flash it to your Raspberry Pi, and boot! This method is by far the *easiest, hassle free* method to getting started with OpenCV on your Raspberry Pi.

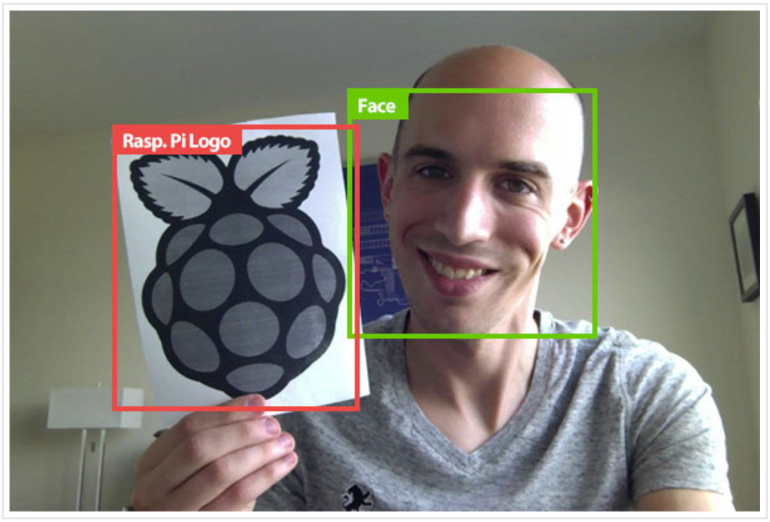
## So, what's next?

Congrats! You have a brand new, fresh install of OpenCV on your Raspberry Pi — and I'm sure you're just itching to leverage your Raspberry Pi to build some awesome computer vision apps.

But I'm also willing to bet that *you're just getting started learning computer vision and OpenCV*, and you're probably feeling a bit confused and overwhelmed on where exactly to start.

Personally, I'm a big fan of **learning by example**, so a good first step would be to read this blog post on accessing your Raspberry Pi Camera with the `picamera` module. This tutorial details the *exact steps* you need to take to (1) capture photos from the camera module and (2) access the raw video stream.

And if you're *really interested* in leveling-up your computer vision skills, you should definitely check out my book, *Practical Python and OpenCV + Case Studies*. My book not only *covers the basics of computer vision and image processing*, but also teaches you how to solve real world computer vision problems including ***face detection in images and video streams***, ***object tracking in video***, and ***handwriting recognition.***



**All code examples covered in the book are guaranteed to run on the Raspberry Pi 2 and Pi 3 as well!** Most programs will also run on the B+ and Zero models, but might be a bit slow due to the limited computing power of the B+ and Zero.

So let's put your fresh install of OpenCV on your Raspberry Pi to good use — ***just click here to learn more about the real-world projects you can solve using your Raspberry Pi + Practical Python and OpenCV .***

## Summary

In this blog post, we learned how to upgrade your ***Raspberry Pi 3***'s OS to ***Raspbian Stretch*** and to install ***OpenCV 3*** with either Python 2.7 or Python 3 bindings.

If you are running a different version of Raspbian (such as *Raspbian Wheezy*) or want to install a different version of OpenCV (such as OpenCV 2.4), please consult the following tutorials:

- Install guide: Raspberry Pi 3 + **Raspbian Jessie** + OpenCV 3
- How to install OpenCV 3.0 on ***Raspbian Jessie****.*
- Installing OpenCV on your *Raspberry Pi Zero* running ***Raspbian Jessie****.*
- Installing OpenCV 3.0 for both Python 2.7 and Python 3+ on ***Raspbian Wheezy.***
- Install OpenCV 2.4 for Python 2.7 on ***Raspbian Wheezy***.

**Are you looking for a project to work on with your new install of OpenCV on Raspbian Stretch?** Readers have been big fans of this post on Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox.

## *But before you go…*

I tend to utilize the Raspberry Pi quite a bit on this blog, so if you're interested in learning more about the Raspberry Pi + computer vision, ***enter your email address in the form below to be notified when these posts go live!***

**install**, **opencv 3**, **pi 3**, **python 2.7**, **python 3**, **raspberry pi**, **raspbian**, **tutorials**

## 124 Responses to *Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi*

**Hilman** September 4, 2017 at 7:22 pm # REPLY

Adrian. I want to ask something.
I have the Raspberry Pi 2 (and its camera module), but I just don't know what kind of project I can do with it.
Since I am busy, if possible, I want to make a project that can contribute the most to what I am learning right now (mainly machine learning). Do you have any idea?
I were thinking of using it for scrapping data, but do not know where to begin. I would be very happy if you could recommend some suggestions.

Thanks!

**Adrian Rosebrock** September 5, 2017 at 9:18 am # REPLY

Hi Hilman — the Raspberry Pi 2 is a bit underpowered so I wouldn't recommend *training* a machine learning classifier on your

Pi, but I could see *deploying* one. Have you considered training an image classifier to recognize a particular object on your laptop/desktop and then actually running it on your Raspberry Pi?

Also, keep in mind that all chapters inside Practical Python and OpenCV will run on the Raspberry Pi. Go through any of those chapters and you can execute the projects on the Pi (such as face detection + tracking). Those chapters make for excellent starting points for projects.

I hope that helps!

**Hilman** September 5, 2017 at 8:59 pm # REPLY

Ah… I forgot about that book. Will take a look at it later. Thanks!

**Jorge** September 5, 2017 at 4:40 am # REPLY

Great post as always.

I wanted to share a neat little trick: You can actually enable SSH on the RPi just by placing an empty "ssh" file (case sensitive!) in the root of the sd card (once Raspbian is flashed).
This makes running a 100% "headless" RPi possible: From there you can keep working with SSH or enable VNC and see the desktop from there.

**Adrian Rosebrock** September 5, 2017 at 9:11 am # REPLY

Wow, that's a neat trick! Thanks for sharing Jorge.

**Hidenori Kaga** September 5, 2017 at 8:09 am # REPLY

Your page has been explained in an easy-to-understand and polite manner, so it's always helpful very much.
I immediately installed the CV with reference to this page.
However, Python IDLE in the program menu causes an error in importing. How can I use CV from IDLE?

**Adrian Rosebrock** September 5, 2017 at 9:10 am # REPLY

Hi Hidenori — as far as I understand, the GUI version of Python IDLE does not support virtual environments, thus you cannot use it. I would suggest you use IDLE via the command line (so you can access the Python virtual environment) or use Jupyter Notebooks. I hope that helps!

**Steve Nicholson** September 5, 2017 at 3:26 pm # REPLY

If you aren't going to run the tests, you can save a fair amount of compile time by not including them. To do that, add "-D BUILD_TESTS=OFF" and "-D BUILD_PERF_TESTS=OFF" to the CMake command line.

**Abkul Orto** September 6, 2017 at 6:41 am # REPLY

Great tutorial.

All your " Quickstart Bundle and Hardcopy Bundle book, Practical Python and OpenCV" I bought are a jewel i am happy to have invested in.

in your blog on "Drowsiness detection with OpenCV of May 8, 2017 in dlib, Facial Landmarks, Tutorials" you suggested :

"If you intend on using a Raspberry Pi for this, I would:

1. Use Haar cascades rather than the HOG face detector. While Haar cascades are less accurate, they are also faster.
2. Use skip frames and only detect faces in every N frames. This will also speedup the pipeline."

I have Pi 3, Kindly do a tutorial with an example to implement the above options.

**Adrian Rosebrock** September 7, 2017 at 7:05 am # REPLY

Hi Abkul — thank you for picking up a copy of Practical Python and OpenCV, I appreciate your support! And yes, I will be covering an updated drowsiness detector for the Raspberry Pi in the future. I can't say exactly when this will be as I'm very busy finishing up the new deep learning book, but it will happen before the end of the year.

**Islam** September 6, 2017 at 10:52 am # REPLY

Where new blog about pixel by pixel for loops

**Adrian Rosebrock** September 7, 2017 at 7:01 am # REPLY

I already covered the blog post you are referring to here. I'll also be doing an updated one on OpenMP in the future.

**Rick Free** September 6, 2017 at 1:37 pm # REPLY

Followed step-by-step and it worked like a charm. Compile took about 4 hours, as expected. Thanks!

**Adrian Rosebrock** September 7, 2017 at 6:59 am # REPLY

Congrats on getting OpenCV installed, Rick! Nice job.

**Galang Hakim** September 7, 2017 at 9:18 am # REPLY

Hi andrian

thanks for sharing
I have success installed opencv to raspberry pi 3
thanks to your guidance

but I wonder
for compiling OpenCV 3 for Python 2.7 and python3.5
the libraries, numpy and site packages only for phyton 3

I tried for 3x
the result are still same
any idea about that ?

**Adrian Rosebrock** September 11, 2017 at 9:32 am # REPLY

You would need to create two *separate* Python virtual environments. One for Python 2.7 and one for Python 3. Form there you can run CMake + make from inside each virtual environment to build OpenCV.

**Charles Horan** September 7, 2017 at 2:13 pm # REPLY

Worked like a charm..mind you I installed opencv3.3 rather than 3.1 … still worked great 😀

**Adrian Rosebrock** September 11, 2017 at 9:30 am # REPLY

Congrats on getting OpenCV installed, Charles! Nice job.

**Larry Pechacek** September 8, 2017 at 8:26 am # REPLY

Upon following this latest tutorial The compile did hang up around 91% using 4 cores on RPi2. Following the tutorial RPi2 on Jessie however compiled on Stretch using all 4 cores without issue.

**Adrian Rosebrock** September 11, 2017 at 9:25 am # REPLY

Thank you for sharing your experience, Larry!

**Andrew** September 10, 2017 at 5:41 pm # REPLY

I am following this install on a Pi 2b. If I backup the sd card – will it work on a Pi 3 ?

**Stephane** September 12, 2017 at 3:12 am # REPLY

Hi Adrian,

I installed OpenCv on my raspberry pi3 (2017-08-16-raspbian-stretch) by following step by step your latest tutorial.
The compilation went well and the result is similar to yours.

When I launch a simple program, see what it returns me:

```
(cv) pi@raspberrypi:~ $ sudo modprobe bcm2835-v4l2
(cv) pi@raspberrypi:~ $ python script.py
Unable to init server: Could not connect: Connection refused
```

(video test:1029): Gtk-WARNING **: cannot open display:
(cv) pi@raspberrypi:~ $

This program works fine on my computer with Linux Mint.
Thanks for your reply and sorry for my approximate English.

Bonjour de France 🙂

Stéphane.

**Adrian Rosebrock** September 12, 2017 at 7:13 am #

REPLY

How are you accessing your Raspberry Pi? Over SSH? Enable X11 forwarding when you SSH into your Pi:

```
$ ssh -X pi@your_ip_address
```

**Stephane** September 12, 2017 at 2:20 pm #

REPLY

Via ssh well on 🙂

Ok I'll try the x11 server activation

cordially

**Chris** September 14, 2017 at 6:24 am #

REPLY

The issue with the multi threaded build is the lack of size of the swap file. You need to increase it to something like 1GB for doing intensive builds.

**Michael** September 14, 2017 at 4:23 pm #

REPLY

I tried serveral times and did follow your great tutorial in detail. Anyhow, I am not able to get the virtualenvwrapper working (and, due to this, I have issues later on).

After the installation of virtualenv and virtualenvwrapper (both were successful, including the dependancies) and updating the ~/.profile file, I always get the error:

pi@raspberrypi:~ $ source ~/.profile
/usr/bin/python: No module named virtualenvwrapper
virtualenvwrapper.sh: There was a problem running the initialization hooks.

If Python could not import the module virtualenvwrapper.hook_loader, check that virtualenvwrapper has been installed for VIRTUALENVWRAPPER_PYTHON=/usr/bin/python and that PATH is set properly.

I did not find any clue to overcome this problem.

Any help is appreciated.

**Adrian Rosebrock** September 18, 2017 at 2:17 pm # REPLY

Which Python version were you trying to install OpenCV + Python for? Python 2.7? Or Python 3?

**Gerard** October 5, 2017 at 1:16 am # REPLY

I'm trying to do 2.7 and get the same error

**Adrian Rosebrock** October 6, 2017 at 5:11 pm # REPLY

I would suggest *explicitly* setting your Python version for virtualenvwrapper inside your `.profile` file, like this:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python2
source /usr/local/bin/virtualenvwrapper.sh
```

**Stephen Hayes** September 20, 2017 at 1:50 pm # REPLY

In setting up the ~/.profile, I had to add the line: "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" before "source /usr/local/bin/virtualenvwrapper.sh". Otherwise, I got the "No module named virtualenvwrapper.hook_loader" error.

**Eyal** October 18, 2017 at 2:21 am # REPLY

I also had to add export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" otherwise I"No module named virtualenvwrapper.hook_loader" error.

But what is the implication for the next step: mkvirtualenv cv -p python2

Thanks

**Adrian Rosebrock** October 19, 2017 at 4:55 pm # REPLY

It should have have an impact.

**Tom** September 17, 2017 at 6:06 am # REPLY

Just a word of warning: Stretch takes more space than Jessie (~4GB vs ~3GB) so to prevent OpenCV from building on a 8GB card.

**Adrian Rosebrock** September 18, 2017 at 2:09 pm # REPLY

Thanks for sharing, Tom!

**Stephen Hayes** September 18, 2017 at 4:16 pm # REPLY

In setting up the ~/.profile, I had to add the line: "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" before "source /usr/local/bin/virtualenvwrapper.sh". Otherwise, I got the "No module named virtualenvwrapper.hook_loader" error.

**Stephen Hayes** September 20, 2017 at 10:18 pm # REPLY

Another note: It is possible to run the make with 4 cores. Stretch suffers a bit from software bloat, so the 2GB of memory isn't sufficient to compile with 4 cores. Edit /etc/dphys-swapfile
and change CONF_SWAPSIZE to 2048 and reboot. You can then do make -j4. you also need to do the same thing to install dlib as that will also hang under stretch when it runs out of memory.

**Adrian Rosebrock** September 21, 2017 at 7:15 am # REPLY

Great point Stephen, thank you for sharing.

**toukir** September 22, 2017 at 2:23 am # REPLY

why can't I import cv2 directly from python shell? but when I go to terminal and write work on and import cv2 then it works.
But I like to import cv2 from the script. so what can I do for that? I am using python 2.7 and pi3.
please help me.

**Adrian Rosebrock** September 22, 2017 at 8:54 am # REPLY

To clarify, are you trying to execute a Python script from your terminal? If so you still need to use "workon" before executing the script (workon only needs to be executed once):

```
$ workon cv
$ python your_script.py
```

**George** October 1, 2017 at 1:05 pm # REPLY

Same problem here, except i'm using Python 3.5.3. Did you find a solution ? Does anyone have a solution ?

**Syed Tauseef** September 22, 2017 at 11:26 am # REPLY

Hey Adrian Rosebrock ! Thanks for such a wonderful and simple tutorial, currently i am installing opencv (currently at installing numpy) i would like to know whether some of the open cv modules are dependent or matplotlib ? If so is it similar to installing like numpy ? with cv virtual environment "pip install matplotlib"

**Adrian Rosebrock** September 22, 2017 at 11:29 am # REPLY

No, there are no OpenCV modules that are dependent on matplotlib. You can install it via:

```
$ pip install matplotlib
```

**Syed Tauseef** September 23, 2017 at 5:02 am # REPLY

Thanks ! I was successfully able to install open cv on my pi3 . At end you forgot to mention about deleting those zips (open cv and contrib ) file which might add few more mb of free space.

**Syed Tauseef** September 23, 2017 at 6:06 am # REPLY

How do we access the environment variable in VNC or desktop terminal as" source ~/.profile "and" workon cv " give invalid option . SSH i am able to get the cv environment.

**Adrian Rosebrock** September 23, 2017 at 9:59 a... # REPLY

Running:

```
$ source ~/.profile
$ workon cv
```

Will work over SSH, VNC, and terminal on the desktop.

I'm not sure what error message you are getting, but again, the above commands will work just fine on all setups (provided there is not a misconfiguration, of course).

**Syed Tauseef** September 23, 2017 at 10:55 am #

Yes ! Now i am able run these line without any errors fron VNC ,dont kown what was the reason for such error . Anyways thanks !

**Niyazi Toker** September 26, 2017 at 5:03 am # REPLY

It does not worked for me. I took Memory Error, although I have memory.
I tried "pip –no-cache-dir install matplotlib" command. Although I took again another error, at the end of I have installed matplotlib.

**Adrian Rosebrock** September 26, 2017 at 8:06 a... # REPLY

Correct, using:

```
$ pip install matplotlib --no-cache-dir
```

Will help if you are running into a MemoryError.

**Himanshu Singh** September 26, 2017 at 6:49 am # REPLY

Hi Adrian,

I have followed your tutorial to install opencv3 using python3 in raspberri Pi3 stretch.

I ran following commands:
source ~/.profile
workon cv
python file_name.py

error:

Traceback (most recent call last):
File “face_detection.py”, line 2, in
importError : No module named ‘picamera’

I tried to install ‘picamera’ using following commands:
sudo apt-get update
sudo apt-get install python3-picamera

Result:

python3-picamera is already installed the newest version (1.13)
0 upgraded, 0 newly installed, 0 to remove and 20 not upgraded

Again I tried to run:
python file_name.py

same importError :

Traceback (most recent call last):
File “face_detection.py”, line 2, in
importError : No module named ‘picamera’

I am not getting why this is happening. It’ll be very helpful if you can help me out of this.

**Adrian Rosebrock** September 26, 2017 at 8:06 am # REPLY

You installed picamera via apt-get which will install it into your system install of Python, not the virtual environment. You should use:

```
$ workon cv
$ pip install picamera
```

And you'll be all set.

**Himanshu Singh** September 26, 2017 at 11:00 am # REPLY

Thanks man! It's working perfectly fine 🙂 🙂 _/_

**James** September 27, 2017 at 5:19 am # REPLY

Really great instructions Adrian, thank you, saved me loads of time. However, i installed everything and found it used up 9.5GB of the sd card, so it might be worth mentioning that 16GB is recommend. I noticed on your setup it only took ~4.2GB. Is that after removing LibreOffice and Wolfram engine?

**Adrian Rosebrock** September 27, 2017 at 6:37 am # REPLY

Correct, that is after removing LibreOffice and Wolfram Engine.

**James** September 27, 2017 at 2:28 pm # REPLY

So worth mentioning if you use a 8GB sd card, you will need to remove LibreOffice and Wolfram Engine, otherwise when you try and build openCV you will most likely run out of space and get an error, then have to build it again, and this step can take hours so not something you want to happen.

**Bryan** September 28, 2017 at 8:12 pm # REPLY

when I try to download https://gethub.com/Itseez/opencv/archive/3.3.0.zip the connection just times out over and over

**Adrian Rosebrock** October 2, 2017 at 10:26 am # REPLY

Hi Bryan — please double-check your internet connection and try again. There might have also been a problem with GitHub when you tried to download the code.

**arjun** September 30, 2017 at 11:40 am # REPLY

can openCV3.3 be installed in jessie

**Adrian Rosebrock** October 2, 2017 at 9:54 am # REPLY

Yes, absolutely. I provide a number of OpenCV install tutorials for various Raspbian distributions here.

**Ace** October 2, 2017 at 4:08 am # REPLY

after expand the file system and reboot …
i cant access to the raspberry pi through Remote Desktop.
but able to use ssh access to the pi

Anyone can help ?

**Adrian Rosebrock** October 2, 2017 at 9:25 am # REPLY

Hmm, I don't think this is an issue related to expanding the file system and rebooting. Can you ensure the remote desktop service is properly running?

**Banjo** October 3, 2017 at 8:24 pm # REPLY

Love the content, I wish I had snapped up the newest course at the early bird discount.

If anyone waited 4 hours for compile, and without thinking pasted the "make clean" command. I feel your pain…from now on, read twice, paste once.

Silver lining: I'll never forget what make clean does.

**Adrian Rosebrock** October 4, 2017 at 12:35 pm # REPLY

Ugh, I'm sorry to hear that Banjo. I'll be posting an updated Raspberry Pi install tutorial this coming Monday, October 9th which will enable you to compile OpenCV on the Raspberry Pi in about 45 minutes.

**Sinan** October 5, 2017 at 1:42 pm # REPLY

Hi Adrian,
When I reached the creation of virtual environment part, I accidently created another virtual part for Python 2.7 as well. But I want to work with Python 3.
In Cmake output, both interpreter and numpy of Python 2.7 and Python 3 are located in the virtual environment cv.
Will it cause any problem in the future?

Thanks in advance

**Adrian Rosebrock** October 6, 2017 at 5:00 pm # REPLY

Just to be safe I would suggest deleting both your Python virtual environment via the `rmvirtualenv` command and correctly creating your Python 3 one. From there, delete your "build" directory, re-create it, and re-run CMake.

**zet** October 6, 2017 at 3:49 am # REPLY

thank you

**Franco** October 7, 2017 at 5:25 pm # REPLY

Hello, thank you Adrian for this tutorial.
I am having trouble at step 5:
when I try the cmake -D……
I get a bash: cmake: command not found.
Any ideas why?

**Adrian Rosebrock** October 9, 2017 at 12:33 pm # REPLY

Please make sure you "Step #2" where the "cmake" command is installed.

**Huy Nguyen** October 7, 2017 at 11:26 pm # REPLY

Hi Adrian,

I followed the instructions and were able to install OpenCV 3.3.0 on my Raspberry Pi 3. Everything seemed to work pretty well until I ran into a Face Recognition Script that contained the following line:

model = cv2.face.createEigenFaceRecognizer()

The following Error Message was shown when I ran Python on it:

“AttributeError: module ‘cv2’ has no attribute ‘face’ ”

Everywhere I looked, the answer seemed to point to the “OpenCV’s extra modules” which I thought was already installed with opencv_contrib from the github.

Any idea on how I can fix this?

TIA,

Huy –

**Adrian Rosebrock** October 9, 2017 at 12:32 pm # REPLY

Indeed, it sounds like your install of OpenCV did not include the “opencv_contrib” modules. You will need to re-compile and re-install OpenCV.

**Leon** October 8, 2017 at 12:26 am # REPLY

Hi.
the virtualenvironment is killing me.
Unless I’m in it, I can’t see cv2 library.

If I am in it, python can’t see picamera.array and other modules.

This makes the home surveillance blog that was suggested to try opencv out impossible.

time spent thus far: 10h compiling (even without make -j, it crashed at 83%, although power cycle -ie too hung to ssh into and stop cleanly) and 3h on this. Definitely not for the faint hearted!

Is there an easy way to get needed modules into virtual environment? I have to abandon this soon – it is consuming too much time. A pity really, because I was hoping to bring it to my classroom.

Cheers,
Leon

**Adrian Rosebrock** October 9, 2017 at 12:31 pm # REPLY

Hi Leon — that is the intended behavior of Python virtual environments. Python environments keep your system Python packages separate from your development ones. As far as your Pi crashing during the compile, take a look a this blog post which provides a solution. The gist is that you need to increase your swap space.

**Aaron** October 8, 2017 at 2:12 pm # REPLY

Will this tutorial work on a raspberry pi 2? Currently its os is raspian wheezy and from the offial website you can only download raspian stretch

**Adrian Rosebrock** October 9, 2017 at 12:21 pm # REPLY

Hi Aaron — can you please clarify your comment? Are you running Raspbian Wheezy on your Pi and want to install OpenCV?

**Aaron** October 13, 2017 at 3:44 am # REPLY

I have changed to Raspian Stretch but i have a Raspberry Pi 2 just wondering will this tutorial work for it as you are using a Raspberry Pi 3

**Adrian Rosebrock** October 13, 2017 at 8:35 am # REPLY

I have not tested on the Raspberry Pi 2, but yes, this tutorial should work.

**Haseeb Ahmed** October 10, 2017 at 6:36 am # REPLY

Hi Adrian, i am using sift for features extratction. but it is showing that
“module” has no attribute “xfeatures2d”. I have downloaded opencv_contrib and unzip it properly but still getting error when i run my code on raspberry pi 3.

**Adrian Rosebrock** October 13, 2017 at 9:10 am # REPLY

Hi Haseeb — it sounds like your path to the `opencv_contrib` module during the CMake step was incorrect. Double-check the path, re-run CMake, and re-compile + re-install OpenCV.

**Bryan** October 12, 2017 at 12:24 pm # REPLY

Made it to the open CV compile
stopped at 86%

Spec ( Ras pi 0 wireless , stretch , python 2.7)

should I attempt to change to python 3.0, reformat the sd card and start over, or attempt optimizing open CV and make -j4? useing your oct 9th article.

Thanks

**Adrian Rosebrock** October 13, 2017 at 8:39 am # REPLY

I would actually update your swap size, as I discuss in this post. From there, delete your “build” directory, re-create it, and re-compile.

**bob** October 13, 2017 at 5:03 am # REPLY

is there a way you can you run the Idle shell for python in the virtual environment

**Adrian Rosebrock** October 13, 2017 at 8:33 am # REPLY

No, IDLE does not respect Python virtual environments. Please use the command line. If you like IDLE, try using Jupyter

Notebooks that do work with Python virtual environments.

**Raju** October 13, 2017 at 7:40 am # REPLY

Hi,

I am not able to import PIL from virtualenv. Can you please suggest how to fix the issue?

Thanks in advance
Raju

**Adrian Rosebrock** October 13, 2017 at 8:32 am # REPLY

How did you install PIL? Did you install it into the Python virtual environment?

```
$ workon cv
$ pip install pillow
```

**Keith Glasnapp** October 14, 2017 at 7:12 am # REPLY

Why can't your tutorial be run from a shell script? It would eliminate a lot of errors and retries?

**Adrian Rosebrock** October 14, 2017 at 10:30 am # REPLY

It can be executed via a shell script in some situations; however, it's also important to understand how the compile works, especially if you intend on optimizing your install. I also offer a pre-configured Raspbian .img file inside Practical Python and OpenCV.

**Terry** October 14, 2017 at 11:28 am # REPLY

Can this tutorial work on a 8GB card?

**Adrian Rosebrock** October 16, 2017 at 12:34 pm # REPLY

If you purge Wolfram Alpha and LibreOffice you should be okay, but I would really suggest using a 16GB card.

**rich** October 15, 2017 at 8:51 pm # REPLY

Oh man. I found this link for stretch. It seems to work because everything went smoothly untill I try to compile opencv. At this step cd ~/opencv-3.3.0/ it tells me that the directory does't exist.

So I mkdir one then mkdir build, cd into build. when try to compile, it tells me it doesn't contain CMakeLists.txt , where is this CMakeLists.txt ?

**Adrian Rosebrock** October 16, 2017 at 12:22 pm # REPLY

Hi Rich — it's hard to say what the exact issue is without seeing the directory structure of your project. Can you ensure that OpenCV was properly downloaded and unzipped in your home directory?

**Daniel** October 16, 2017 at 2:13 am # REPLY

I'm very new to this and I just followed your video and now I have it downloaded so thank you. However, I don't know what to do know, like how to write and run scripts in this virtual environment because every time I type python into it, the shell pops up. Could someone tell me how to open blank scripts so I can start writing code. Thank you

**Adrian Rosebrock** October 16, 2017 at 12:20 pm # REPLY

Hey Daniel — you would need to supply the path to the Python script you would like to execute:

```
$ python your_script.py
```

Open up the a file in your favorite plaintext editor, save it as a .py file, and insert your code. From there execute it via the command line.

**Marc** October 16, 2017 at 10:54 am # REPLY

Could add "nohup" to long-running commands? It's not unusual to lose the ssh connection and it gets frustrating as the installation is pretty long already.

**Adrian Rosebrock** October 16, 2017 at 12:14 pm # REPLY

You could absolutely use "nohop"; however, I prefer using "screen".

**Scott** October 17, 2017 at 12:24 pm # REPLY

In step 5, after cmake completes I type 'make' but there are no Makefiles that it can run against.

**Adrian Rosebrock** October 19, 2017 at 5:02 pm # REPLY

Please check your output of "CMake" as it likely returned an error (and thus no Makefiles were generated).

**olivia** October 19, 2017 at 12:09 am # REPLY

Hello adrian,
thank you. this is the best tutorial i ever seen.
but i got one problem
when i'm run this script

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

i just get 300+ (3 digit) even though you get 1852

and then i'm
$ ls -l /usr/local/lib/python3.5/site-packages/
i got 3500+ (4 digit) and you got 1852

please help me adrian to fix this, thank you

**Adrian Rosebrock** October 19, 2017 at 12:20 pm # REPLY

Hi Olivia — thanks for the comment. That number will depend upon the packages you have installed in your environment.

**olivia** October 29, 2017 at 9:07 am # REPLY

i just follow your step adrian, why i get that value? is that ok for my next step when i want to using python?

**Adrian Rosebrock** October 30, 2017 at 2:58 pm # REPLY

I think it is safe to carry on with the instructions. Don't worry about these values.

**Akash** October 19, 2017 at 3:44 am # REPLY

Hello Adrian,
I have installed the opencv inside the virtual environment ,
but i'm unable to access outside (i.e)., Inside the IDLE.
How to access it.

**Adrian Rosebrock** October 19, 2017 at 8:36 am # REPLY

Hi Akash — thanks for the comment. Unfortunately, IDLE cannot access Python virtual environments. I would suggest using Jupyter Notebooks which do work with Python virtual environments.

**Cahit** October 21, 2017 at 11:11 pm # REPLY

Hello Adrian. I just wanted to say thanks. It took a long time but it was a problem-free installation. Nice guide.

**Adrian Rosebrock** October 22, 2017 at 8:24 am # REPLY

Congrats on getting OpenCV installed, nice job!

**Enzo** October 23, 2017 at 1:37 pm # REPLY

Hi Adrian, great walkthrough, tutorial or wathever you want to call it, made it without any trouble in the times that are say… although this only work as long as someone work on the virtual environment, is