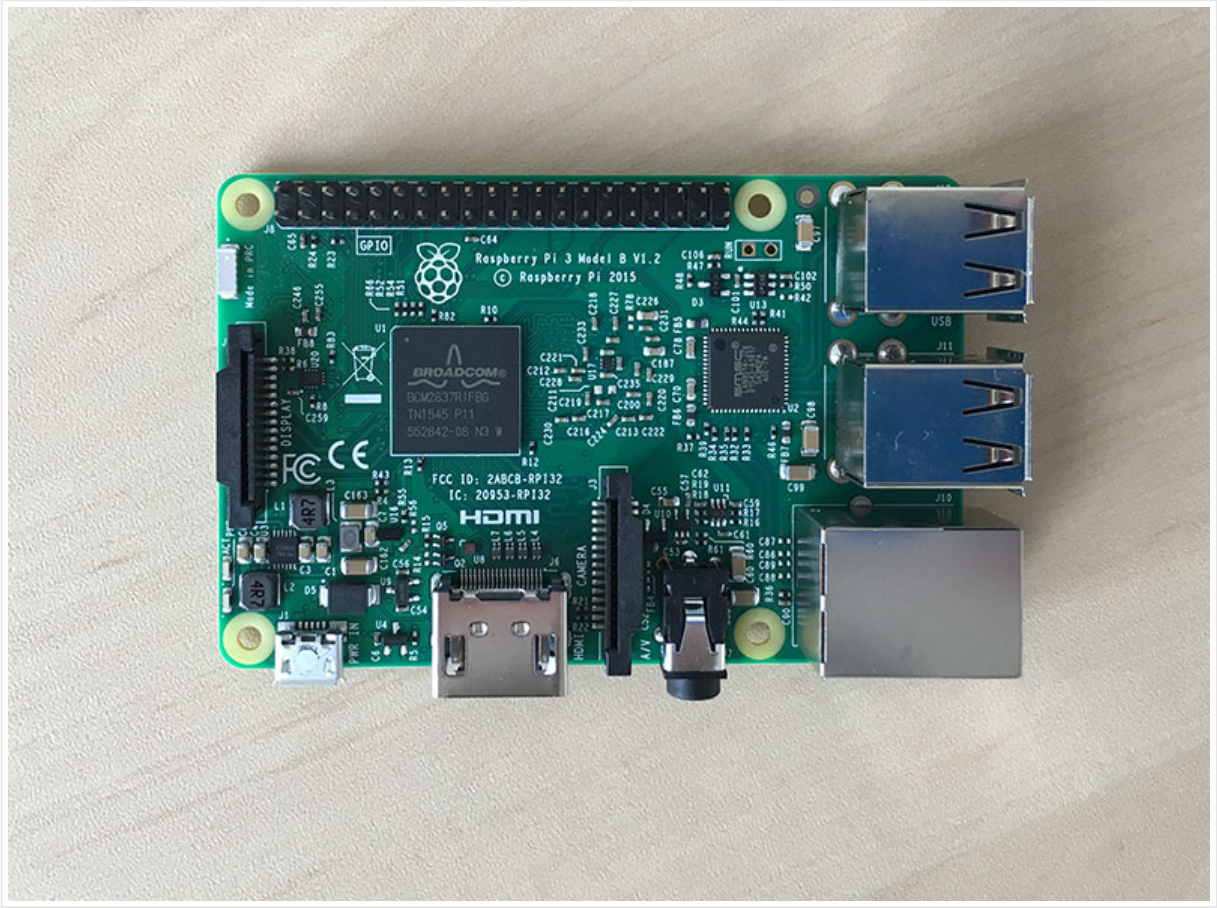
pyimagesearch
be awesome at building image search engines

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3

by **Adrian Rosebrock** on April 18, 2016 in **OpenCV 3**, **Raspberry Pi**, **Tutorials**



Can you believe it's been over ***four years*** since the original Raspberry Pi model B was released? Back then the Pi Model B shipped with only 256MB of RAM and a 700MHz single core processor.

Just over ***one year ago*** the Raspberry Pi 2 was unleash Pi", this beast made an impact on the computer world like ar 900MHz ***quad-core*** processor — quite the upgrade fro

In my opinion, ***the Raspberry Pi 2 is what made computer vision possible on the Pi platform*** (at least from a Python + OpenCV perspective). The original model B simply didn't have the processing capacity (or the RAM) to be powerful enough to process images video streams for anything more than trivial operations — *the Pi 2 changed all that.*

In fact, the Raspberry Pi 2 had *such a meaningful impact* on the computer vision space, that I even took the time to make a *all* code examples in *Practical Python and OpenCV* compatible with the Pi.

And now we have the Raspberry Pi 3:

- 1.2Ghz 64-bit quad-core processor.
- 1GB RAM.
- Integrated 802.11n wireless and bluetooth.

Personally, I was hoping for a bit more RAM (perhaps in t
processor with *33% increased performance* is *well worth*

Just as I have done in previous blog posts, I'll be demon
bindings on Raspbian Jessie.

If you are looking for previous installation instructions for

- How to install OpenCV 3.0 on ***Raspbian Jessie***.
- Installing OpenCV on your *Raspberry Pi Zero* runnin
- Installing OpenCV 3.0 for both Python 2.7 and Pyth
- Install OpenCV 2.4 for Python 2.7 on ***Raspbian W***

Otherwise, let's proceed with getting OpenCV 3 installed

Assumptions

In this tutorial, I am going to assume that you already ow
ed.

You should also have *either*:

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

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.

Installing OpenCV 3 on a Raspberry Pi 3 running Raspbian Jessie

If you've ever installed OpenCV on a Raspberry Pi (or an
an
be quite time consuming with many dependencies and p

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. That said, the *Pi 3 is substantially faster than the Pi 2*, so the time it takes to compile OpenCV has decreased ***dramatically***.

Anyway, let's go ahead and get started installing OpenCV 3 on your brand new Raspberry Pi 3 running Raspbian Jessie.

Step #1: Expand filesystem

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

If so, the first thing you should do is expand your filesyste rd:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie l
1 $ sudo raspi-config
```

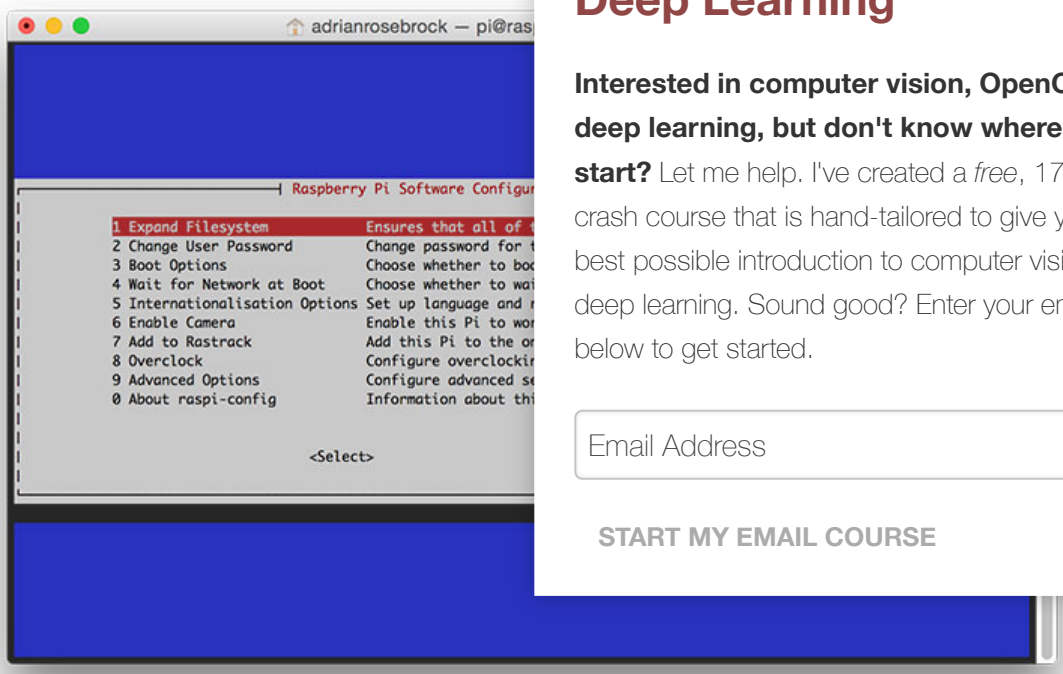


Figure 1: Expanding the filesystem on your Raspberry Pi 3.

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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3 Shell
1 $ sudo reboot
```

After rebooting, your file system should have been expan
card. You can verify that the disk has been expanded by

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       7.2G  3.3G  3.6G  48% /
devtmpfs        459M     0  459M   0% /dev
tmpfs           463M     0  463M   0% /dev/shm
tmpfs           463M  6.4M  457M   2% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           463M     0  463M   0% /sys/fs/cgroup
/dev/mmcblk0p1   60M   20M   41M  34% /boot
tmpfs            93M     0   93M   0% /run/user/1000
```

As you can see, my Raspbian filesystem has been expanded to include all 8GB of the micro-SD card.

However, even with my filesystem expanded, I have already used 48% of my 8GB card!

OpenCV, along with all its dependencies, will need a few... the Wolfram engine to free up some space on your Pi:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
$ sudo apt-get purge wolfram-engine
```

After removing the Wolfram Engine, *you can recla...*

Step #2: Install dependencies

This isn't the first time I've discussed how to install Open... on the briefer side, allowing you to work through the insta... ***it takes to execute each command*** so you can plan... (OpenCV itself takes **1h 12m** to compile).

The first step is to update and upgrade any existing pack...

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
$ sudo apt-get update
$ sudo apt-get upgrade
```

Timing: 1m 26s

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

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
$ sudo apt-get install build-essential cmake pkg-config
```

Timing: 40s

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.:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
$ sudo apt-get install libjpeg-dev libtiff5-d...
```

Timing: 32s

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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
1 $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
2 $ sudo apt-get install libxvidcore-dev libx264-dev
```

Timing: 34s

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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ sudo apt-get install libgtk2.0-dev
```

Timing: 3m 6s

Many operations inside of OpenCV (namely matrix opera dependencies:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ sudo apt-get install libatlas-base-dev gfor
```

Timing: 46s

These optimization libraries are *especially important* for re

Lastly, let's install both the Python 2.7 and Python 3 hea bindings:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ sudo apt-get install python2.7-dev python3-
```

Timing: 45s

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 `3.1.0` archive of OpenCV from the official OpenCV repository. (*Note:* As future versions of openCV are released, you can replace `3.1.0` with the latest version number):

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ cd ~
2 $ wget -O opencv.zip https://github.com/Itsee
3 $ unzip opencv.zip
```

Timing: 1m 26s

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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
1 $ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.1.0.zip
2 $ unzip opencv_contrib.zip
```

Timing: 43s

You might need to expand the command above using the "<=>" button during your copy and paste. The `.zip` in the `3.1.0.zip` may appear to be cutoff in some browsers. The full URL of the OpenCV 3.1.0 archive is:

https://github.com/Itseez/opencv_contrib/archive/3.1.0.

Note: *Make sure your* `opencv` *and* `opencv_contrib` *versions numbers do not match up, then you'll likely run*

Step #4: Python 2.7 or Python 3?

Before we can start compiling OpenCV on our Raspberr manager:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
```

Timing: 20s

If you're a longtime PyImageSearch reader, then you'll kn virtualenvwrapper. Installing these packages is not a requ without them, but that said, ***I highly recommend you*** future will also leverage Python virtual environments. I'll al `virtualenvwrapper` installed throughout the remainder

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 e 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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
1 $ sudo pip install virtualenv virtualenvwrapper
2 $ sudo rm -rf ~/.cache/pip
```

Timing: 9s

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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 source /usr/local/bin/virtualenvwrapper.sh
```

In previous tutorials, I've recommended using your favorit or `nano` to update the `~/.profile` file. If you're comfort to reflect the changes mentioned above.

Otherwise, you should simply use `cat` and output redi

```
Install guide: Raspberry Pi 3 + Raspbian Jessie                               l
1 $ echo -e "\n# virtualenv and virtualenvwrapp
2 $ echo "export WORKON_HOME=$HOME/.virtualenvs
3 $ echo "source /usr/local/bin/virtualenvwrapp
```

Now that we have our `~/.profile` updated, we need 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
3. Or my personal favorite, ***just use the*** `source` ***co***

```
Install guide: Raspberry Pi 3 + Raspbian Jessie                               l
1 $ 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:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3                    Shell
1 $ 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 t

```
1 $ mkvirtualenv cv -p python3
```

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 Jessie. 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 unneed***
executed only once: to actually *create* the virtual environ

After that, you can use `workon` and you'll be dropped

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ source ~/.profile
2 $ workon cv
```

To validate and ensure you are in the `cv` virtual environ *ext* `(cv)` *preceding your prompt, then you* ***are*** *in the* `cv`

Figure 2: Make sure you see the *"(cv)"* text on your pro

Otherwise, if you ***do not*** see the `(cv)` text, then you ***a***

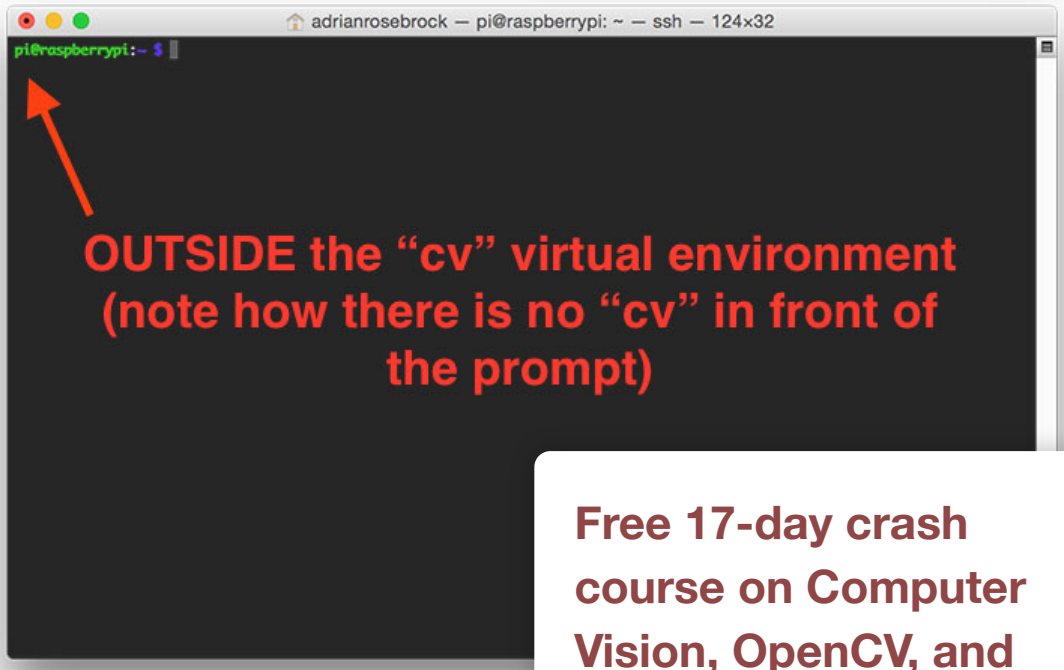


Figure 3: If you do not see the *"(cv)"* text on your prompt, the
"source" and *"workon*

To fix this, simply execute the `source` and `workon` co

Installing NumPy on your Raspberry

Assuming you've made it this far, you should now be in t
for the rest of this tutorial). Our only Python dependency
processing:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
1 $ pip install numpy
```

Timing: 9m 39s

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

Note: *Another 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. 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 p

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
```

```
$ workon cv
```

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

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3 — Shell

```
$ cd ~/opencv-3.1.0/
$ 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-3.1.0/modules \
    -D BUILD_EXAMPLES=ON ..
```

Timing: 1m 57s

Now, before we move on to the actual compilation step,

Start by scrolling down the section titled `Python 2` and

If you are compiling OpenCV 3 for Python 2.7, the
paths to the `Interpreter` , `Libraries` , `numpy` and

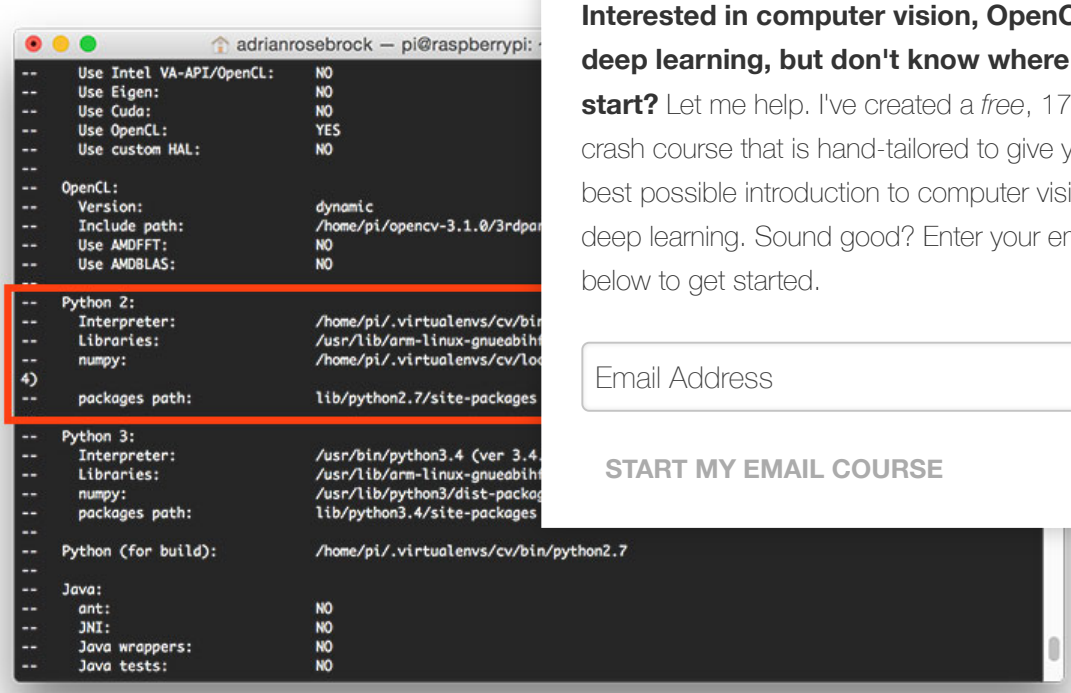


Figure 4: Ensuring that Python 2.7 will be used when compiling OpenCV 3 for Raspbian Jessie on the Raspberry Pi 3.

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***,
below:

Free 17-day crash course on Computer Vision, OpenCV, and Deep Learning

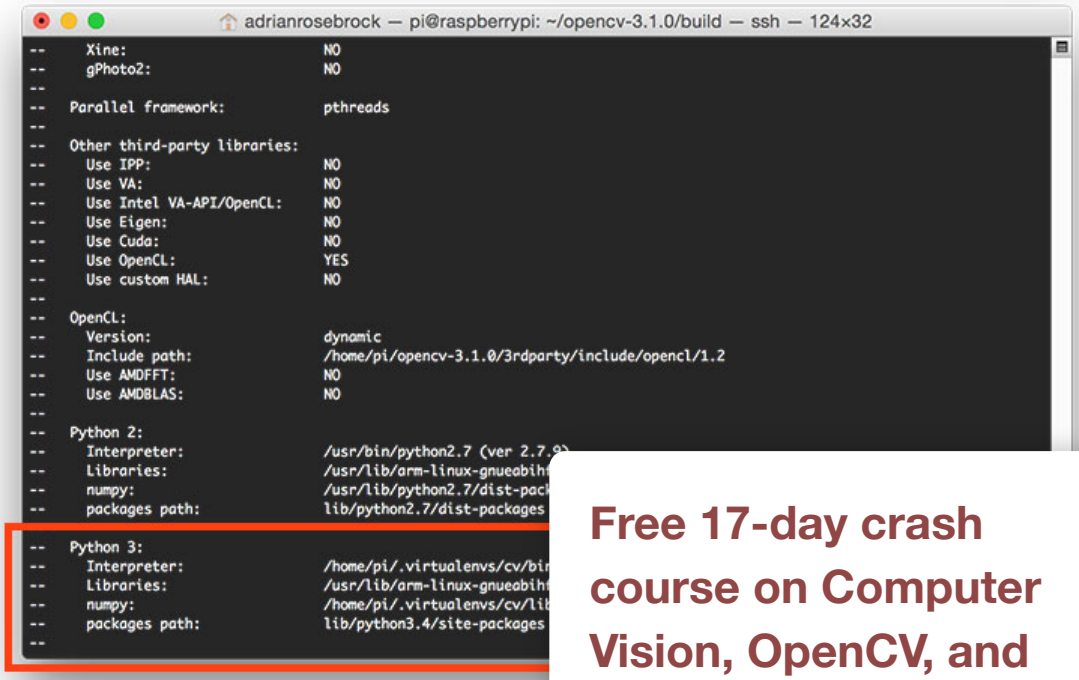


Figure 5: Checking that Python 3 will be used when comp

Again, the `Interpreter` points to our `python3.4` bina
`numpy` points to our NumPy install.

In either case, if you ***do not*** see the `cv` virtual environ
because you are NOT in the `cv` ***virtual environm***

If this is the case, access the `cv` virtual environment us
outlined above.

Finally, we are now ready to compile OpenCV:

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3 — Shell

```
$ make -j4
```

Timing: 1h 12m

Note: *Compiling OpenCV in 72 minutes on the Raspberry Pi 3 is a* ***24%*** *improvement over the previous 95 minutes for the Raspberry Pi 2. That extra 300MHz makes a big difference!*

The `-j4` command controls the number of cores to leverage when compiling OpenCV 3. The Raspberry Pi 3 has *four cores*, thus we supply a value of `4` to allow OpenCV to compile faster.

However, due to race conditions, there are times when
happens to you, I suggest starting the compilation over a

Install guide: Raspberry Pi 3 + Raspbian Jessie

```
$ make clean
$ make
```

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

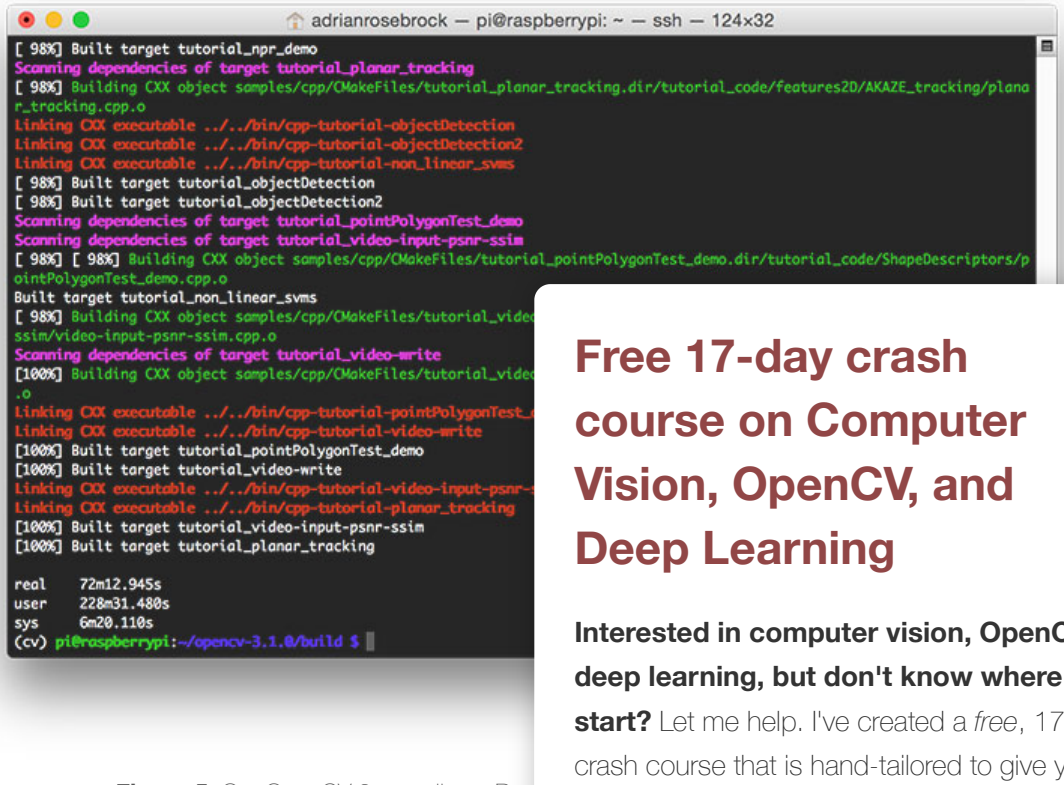


Figure 5: Our OpenCV 3 compile on Ras

From there, all you need to do is install OpenCV 3 on yo

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
$ 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.

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:

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

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:

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3 — 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/python3.4/site-packages` . Again, yo

Install guide: Raspberry Pi 3 + Raspbian Jessie

```
$ ls -l /usr/local/lib/python3.4/site-package
total 1852
-rw-r--r-- 1 root staff 1895932 Mar 20 21:51
```

I honestly don't know why, perhaps it's a bug in the CMa
Python 3+, the output `.so` file is named `cv2.cpython-`
`cv2.so` (like in the Python 2.7 bindings).

Again, I'm not sure exactly *why* this happens, but it's an

Install guide: Raspberry Pi 3 + Raspbian Jessie

```
$ cd /usr/local/lib/python3.4/site-packages/
$ sudo mv cv2.cpython-34m.so cv2.so
```

After renaming to `cv2.so` , we can sym-link our OpenC
3.4:

Install guide: Raspberry Pi 3 + Raspbian Jessie

```
$ cd ~/.virtualenvs/cv/lib/python3.4/site-pac
$ ln -s /usr/local/lib/python3.4/site-package
```


Step #7: Testing your OpenCV 3 install

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

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:

Install guide: Raspberry Pi 3 + Raspbian Jessie

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

```
>>> import cv2
>>> cv2.__version__
'3.1.0'
>>>
```

As you can see from the screenshot of my own terminal, **OpenCV 3 has been successfully installed on my Raspberry Pi 3 + Python 2.7 environment:**

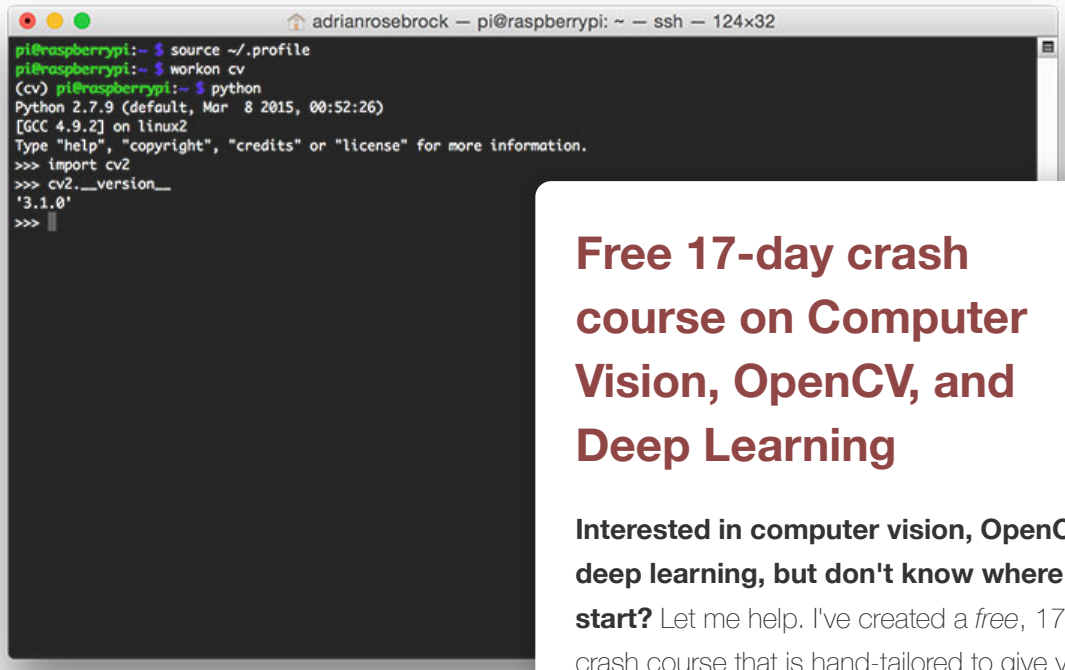


Figure 5: Confirming OpenCV 3 has been successfully i

Once OpenCV has been installed, you can remove both
directories to free up a bunch of space on your disk:

```
Install guide: Raspberry Pi 3 + Raspbian Jessie
$ rm -rf opencv-3.1.0 opencv_contrib-3.1.0
```

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 `pip freeze` and then examining th
`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 Open
get an error: `ImportError: No module named cv2`.

A. Unfortunately, this error is extremely hard to diagnose, be
causing the problem. To start, make sure you are in the
`workon` command fails, then see the first question in th
contents of the `site-packages` directory for your `cv`
directory in `~/.virtualenvs/cv/lib/python2.7/site-p`
`~/.virtualenvs/cv/lib/python3.4/site-packages/`
install). Make sure that your sym-link to the `cv2.so` file

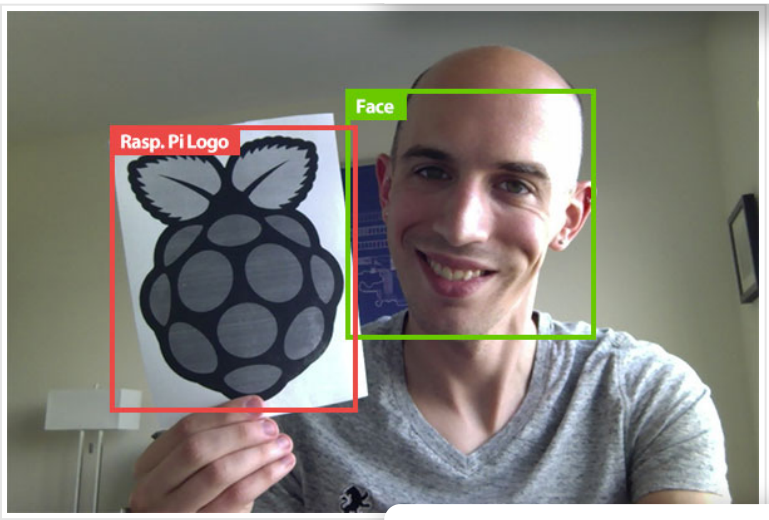
So, what's next?

Congrats! You have a brand new, fresh install of OpenCV
itching to leverage your Raspberry Pi to build some awes

But I'm also willing to bet that *you're just getting started le*
probably feeling a bit confused and overwhelmed on wh

Personally, I'm a big fan of **learning by example**, so a
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 guara
well! Most programs will also run on the B+ and Zero m
computing power of the B+ and Zero.

So let's put your fresh install of OpenCV on your Raspbe
about the real-world projects you can solve using
OpenCV .

Summary

In this blog post, we learned how to install ***OpenCV 3*** w
your ***Raspberry Pi 3*** running ***Raspbian Jessie***.

If you are running a different version of Raspbian (such a
of OpenCV (such as OpenCV 2.4), please consult the fo

- How to install OpenCV 3.0 on ***Raspbian Jessie***.
- Installing OpenCV on your *Raspberry Pi Zero* runnin
- Installing OpenCV 3.0 for both Python 2.7 and Pyth
- Install OpenCV 2.4 for Python 2.7 on ***Raspbian Wheezy***.

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!***

Resource Guide (it's totally free).