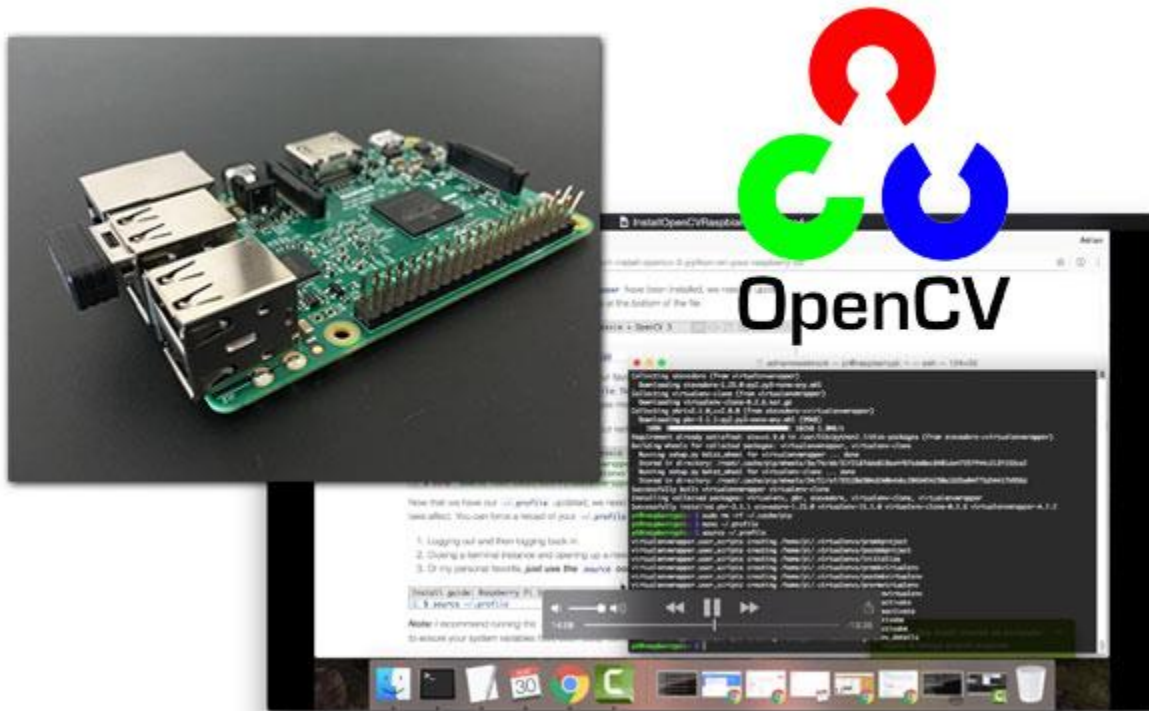# Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi

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

Source: https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/

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 ssh pi@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 on your Raspberry Pi
Shell

```
1 $ sudo raspi-config
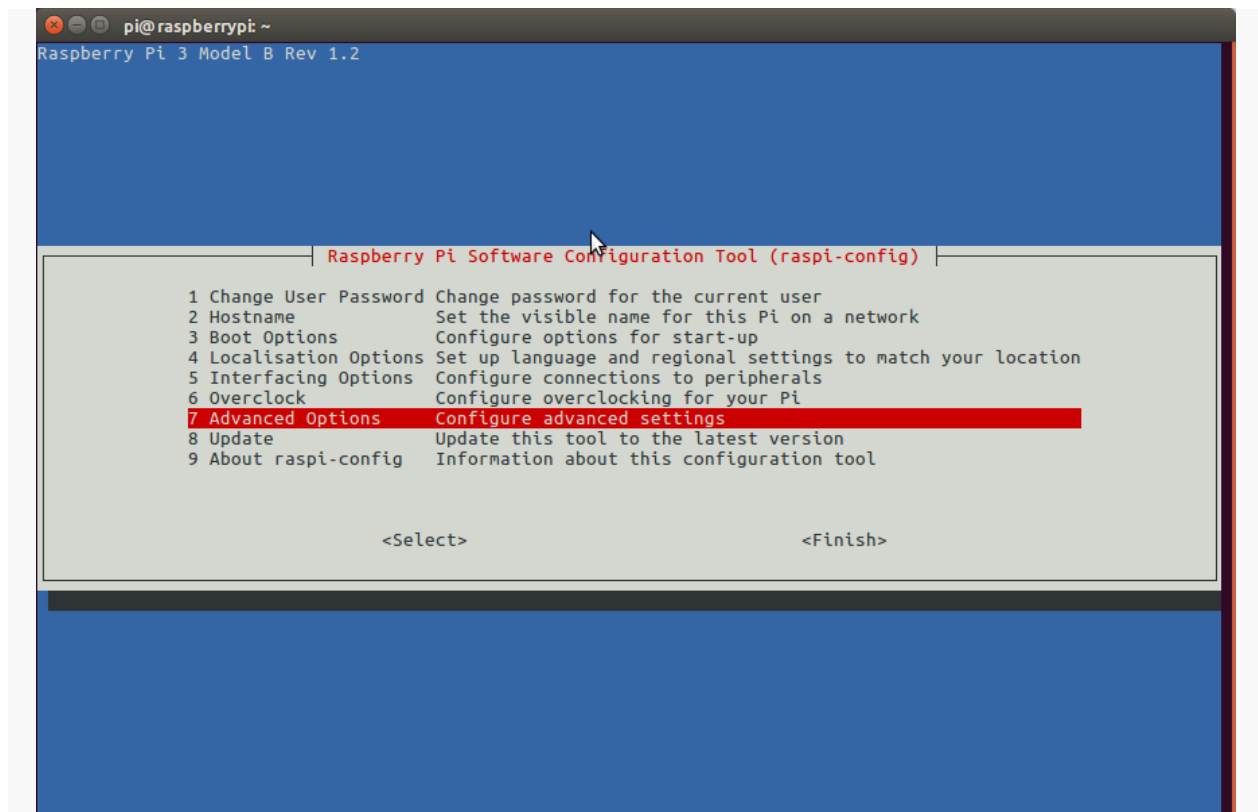```

And then select the "Advanced Options" menu item:

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

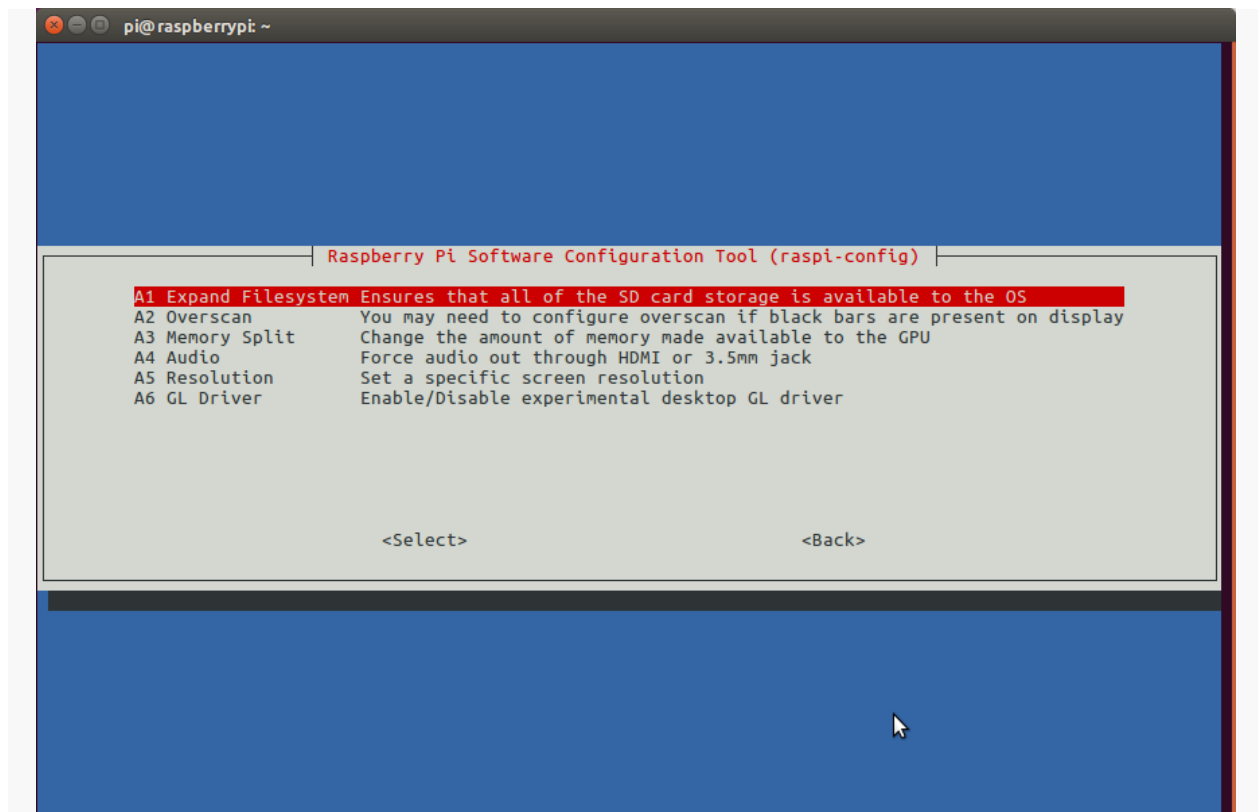Followed by selecting "Expand filesystem":

**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

```
1 $ 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

```
 1  $ df -h
 2  Filesystem      Size  Used Avail Use% Mounted on
 3  /dev/root       30G  4.2G  24G  15% /
 4  devtmpfs       434M    0  434M   0% /dev
 5  tmpfs          438M    0  438M   0% /dev/shm
 6  tmpfs          438M  12M  427M   3% /run
 7  tmpfs          5.0M 4.0K 5.0M   1% /run/lock
 8  tmpfs          438M    0  438M   0% /sys/fs/cgroup
 9  /dev/mmcblk0p1  42M  21M  21M  51% /boot
10  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:

Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
Shell
```
1  $ sudo apt-get purge wolfram-engine
2  $ sudo apt-get purge libreoffice*
3  $ sudo apt-get clean
4  $ 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 brief 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
```
1  $ 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
```
1  $ 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
```
1  $ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```
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

```
1 $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
2 $ 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:

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

```
1 $ 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 on your Raspberry Pi
Shell

```
1 $ 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 on your Raspberry Pi
Shell

```
1 $ 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 on your Raspberry Pi

Shell

```
1 $ cd ~
2 $ wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
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 + Python on your Raspberry Pi
Shell

```
1 $ wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
2 $ 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

```
1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
3 $ 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 virtualenvwrapperinstalled 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

```
1 $ sudo pip install virtualenv virtualenvwrapper
2 $ 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

```
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
4 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

```
1 $ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
2 $ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
3 $ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.profile
4 $ 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

   ```
   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:

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

```
1 $ 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

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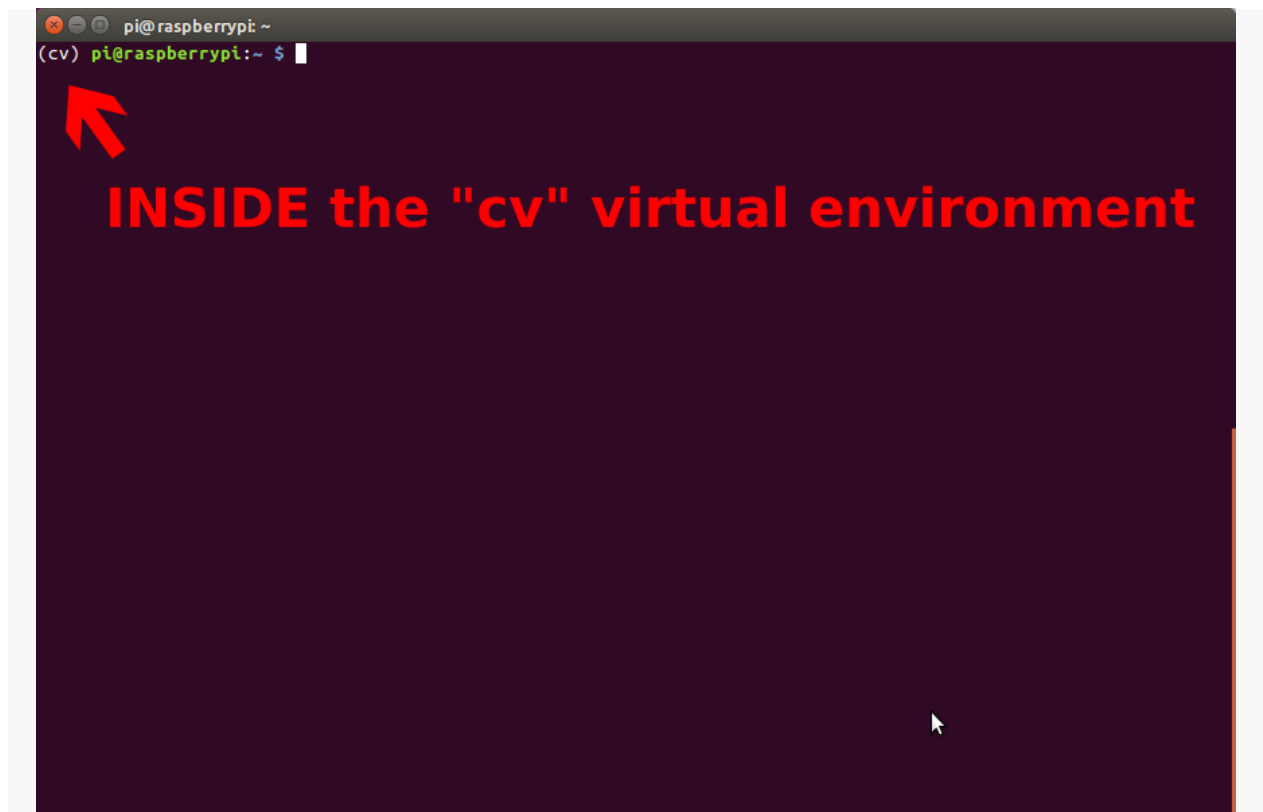
Figure 3: Make sure you see the "*(cv)*" text on your prompt, indicating that you *are* in the cv virtual environment.

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

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 on your Raspberry Pi
Shell

```
1 $ 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 :

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

```
1 $ 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
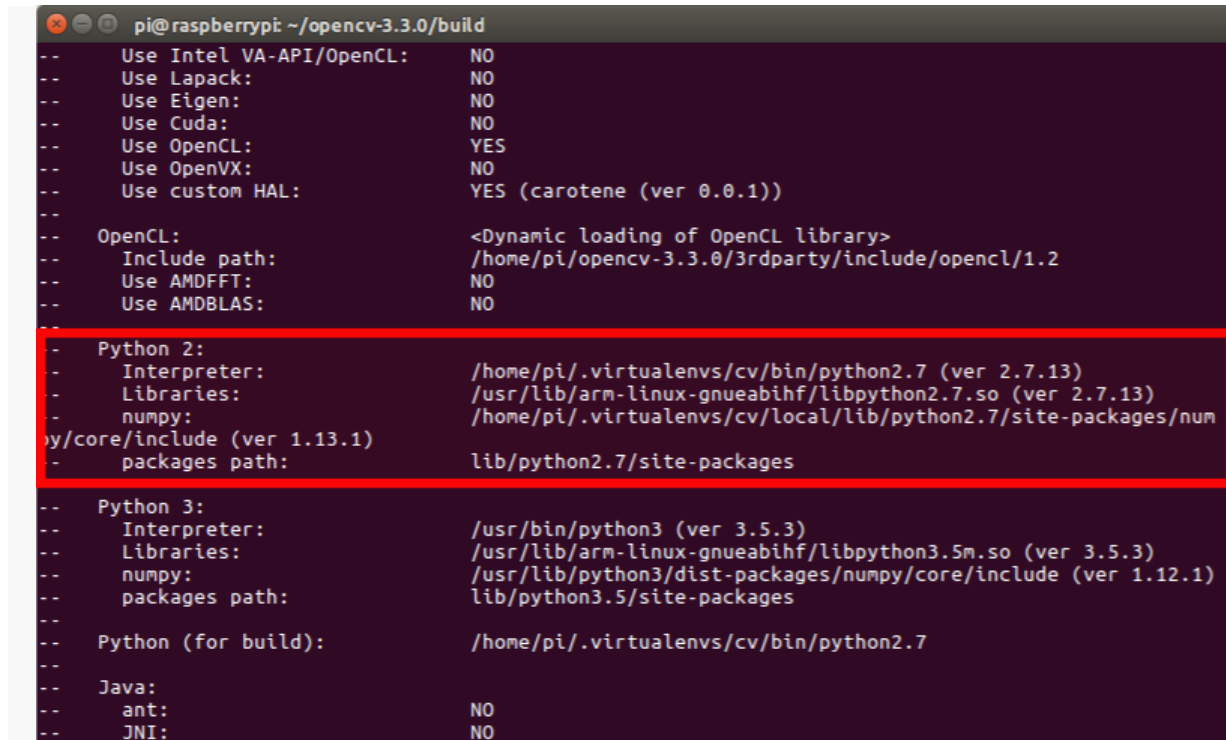
```
1 $ cd ~/opencv-3.3.0/
2 $ mkdir build
3 $ cd build
4 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
5     -D CMAKE_INSTALL_PREFIX=/usr/local \
6     -D INSTALL_PYTHON_EXAMPLES=ON \
7     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
8     -D BUILD_EXAMPLES=ON ..
```

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:

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:

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.

Configure your swap space size before compiling

Before you start the compile process, you should **increase your swap space size**. This enables OpenCV to **compile with all four cores** of the Raspberry PI without the compile hanging due to memory problems.

Open your /etc/dphys-swapfile and then edit the CONF_SWAPSIZE variable:
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
Shell

```
1 # set size to absolute value, leaving empty (default) then uses computed value
2 #   you most likely don't want this, unless you have an special disk situation
3 # CONF_SWAPSIZE=100
4 CONF_SWAPSIZE=1024
```

Notice that I've commented out the 100MB line and added a 1024MB line. This is the secret to getting compiling with multiple cores on the Raspbian Stretch.

If you skip this step, OpenCV might not compile.

To activate the new swap space, restart the swap service:

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

```
1 $ sudo /etc/init.d/dphys-swapfile stop
2 $ sudo /etc/init.d/dphys-swapfile start
```

*Note: It is possible to burn out the Raspberry Pi microSD card because flash memory has a limited number of writes until the card won't work. It is **highly recommended** that you change this setting back to the default when you are done compiling and testing the install (see below). To read more about swap sizes corrupting memory, see* [this page.](#)

Finally, we are now ready to compile OpenCV:

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

```
1 $ make -j4
```

Timing: 1h 30m

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

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

```
1 $ sudo make install
2 $ 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

```
1 $ ls -l /usr/local/lib/python2.7/site-packages/
2 total 1852
3 -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

```
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/python3.5/site-packages . Again, you can verify this with the ls command:

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

```
1 $ ls -l /usr/local/lib/python3.5/site-packages/
2 total 1852
3 -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

```
1 $ cd /usr/local/lib/python3.5/site-packages/
2 $ 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

```
1 $ cd ~/.virtualenvs/cv/lib/python3.5/site-packages/
2 $ 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.
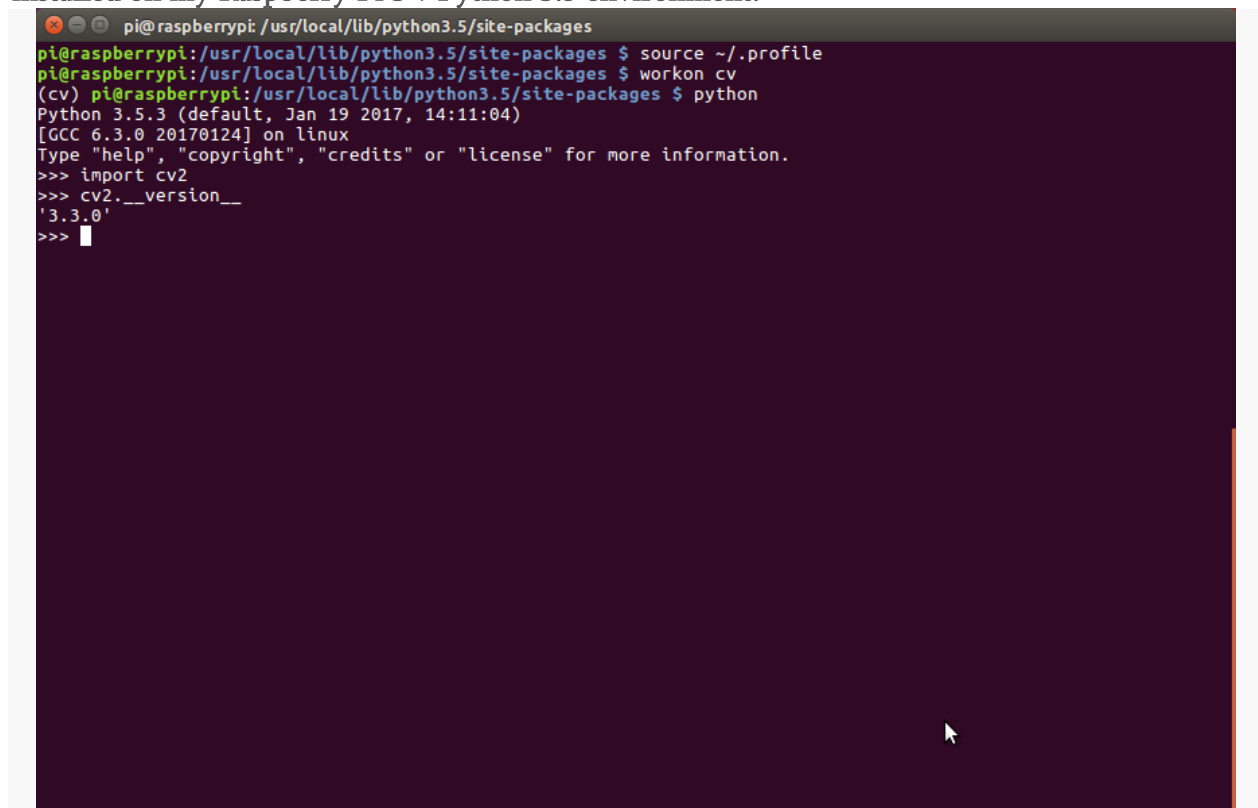
Open up a new terminal, execute the source and workon commands, and then finally attempt to import the Python + OpenCV bindings:

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

```
1 $ source ~/.profile
2 $ workon cv
3 $ python
4 >>> import cv2
5 >>> cv2.__version__
6 '3.3.0'
7 >>>
```

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:**

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

```
1 $ 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.

Don't forget to change your swap size back!

Open your /etc/dphys-swapfile and then edit the CONF_SWAPSIZE variable:
Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
Shell

```
1 # set size to absolute value, leaving empty (default) then uses computed value
2 #   you most likely don't want this, unless you have an special disk situation
3 CONF_SWAPSIZE=100
4 # CONF_SWAPSIZE=1024
```

Notice that I've commented out the 1024MB line and uncommented the 100MB line.

If you skip this step, your memory card won't last as long. As stated above, larger swap spaces may lead to memory corruption, so I recommend setting it back to 100MB.

To revert to the smaller swap space, restart the swap service:

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

```
1 $ sudo /etc/init.d/dphys-swapfile stop
2 $ sudo /etc/init.d/dphys-swapfile start
```

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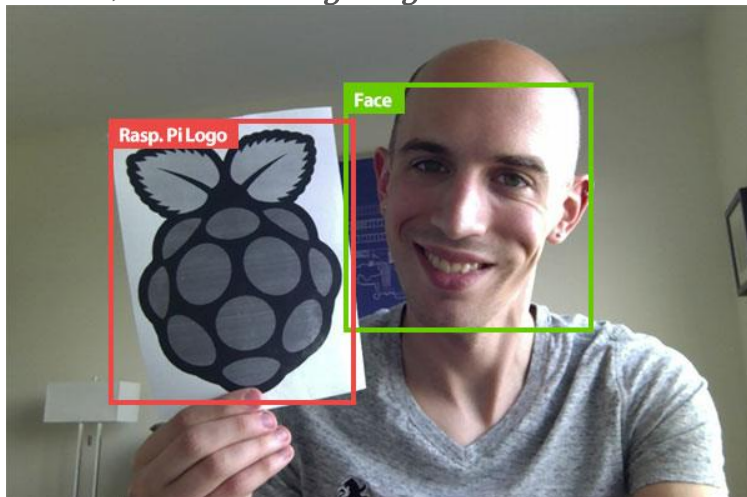
All code examples covered in the book are guaranteed to run on the Raspberry Pi 2 and Pi 3 as well! Most programs will also run on the B+ and Zero models, but might be a bit slow due to the limited computing power of the B+ and Zero.

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.