Navigation

pyimagesearch
be awesome at building image search engines

Ubuntu 18.04: How to install OpenCV

by **Adrian Rosebrock** on May 28, 2018 in **OpenCV 3**, **Tutorials**

Like 58

In this blog post you will learn how to install OpenCV on Ubuntu 18.04.

In the past, I've authored a handful of installation guides for Ubuntu:

- Ubuntu 16.04: How to install OpenCV with Python 2.7 and Python 3.5+
- Install OpenCV 3.0 and Python 2.7+ on Ubuntu
- Install OpenCV 3.0 and Python 3.4+ on Ubuntu
- (…and be sure to see this page if you're looking for macOS and Raspberry Pi installation guides)

The folks at Canonical have been working hard.

On April 26, 2018, they've released a new Long Term Support (LTS) version of Ubuntu for the community: **Ubuntu 18.04 LTS (Bionic Beaver)**.

Support for Ubuntu 16.04 LTS continues until April 2021 so rest-assured — *you don't have to upgrade your 16.04 OS to continue working on your image processing projects.*

That said, if you want to upgrade to Ubuntu 18.04 and use the latest-and-greatest, I think you'll be quite pleased with the new changes in Ubuntu 18.04.

Let's get down to business and install OpenCV with Python 3 bindings.

To learn how to stand up your Ubuntu 18.04 system with OpenCV, *just keep reading.*

Note: *While you won't see an Ubuntu 17.10 specific guide here on my blog (*non-*LTS), these instructions may work with 17.10 (you'll just have to proceed at your own risk).*

Ubuntu 18.04: How to install OpenCV

One major change in Ubuntu 18.04 is that they've dropped Python 2.7 completely.

You can still install Python 2.7 *if-needed*, **but now Python 3 is the default on the OS.**

Given that, this guide supports Python 3. If you need Python 2.7 support, read this entire guide *first* and then check the first question of the **Troubleshooting your install (FAQ)** section near the bottom of this blog post for some Python 2.7 pointers.

Step #0: Get comfortable — you'll be using Python 3.6

Let's familiarize ourselves with Python 3 on Ubuntu 18.04.

To run Python 3 on Ubuntu 18.04, you must call `python3` explicitly.

Let's see which version is installed on our system:

Ubuntu 18.04: How to install OpenCV Shell
```
$ python3 --version
Python 3.6.5
```

And now, let's launch a Python 3 shell just to test the waters:

Ubuntu 18.04: How to install OpenCV Shell
```
$ python3
>> print("OpenCV + Ubuntu 18.04!")
OpenCV + Ubuntu 18.04!
>> quit()
```

That's easy enough, so let's get on with installing OpenCV on Ubuntu 18.04.

Step #1: Install OpenCV dependencies on Ubuntu 18.04

All steps today will be accomplished in the terminal/command line. Before we begin, open a terminal or connect via SSH.

From there, we need to refresh/upgrade the pre-installed packages/libraries with the apt-get package manager:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
$ sudo apt-get update
$ sudo apt-get upgrade
```

Followed by installing developer tools:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
$ sudo apt-get install build-essential cmake unzip pkg-config
```

You most likely already have `pkg-config` installed on Ubuntu 18.04, but be sure to include it in the install command for sanity.

Next, we need to install some OpenCV-specific prerequisites. OpenCV is an image processing/computer vision library and therefore it needs to be able to load standard image file formats such as JPEG, PNG, TIFF, etc. The following image I/O packages will allow OpenCV to work with image files:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
$ sudo apt-get install libjpeg-dev libpng-dev libtiff-dev
```

Similarly, let's include video I/O packages as we often work with video on the PyImageSearch blog. You'll need the following packages so you can work with your camera stream and process video files:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
$ sudo apt-get install libxvidcore-dev libx264-dev
```

OpenCV's highgui module relies on the GTK library for GUI operations. The highgui module will allow you to create elementary GUIs which display images, handle kepresses/mouse clicks, and create sliders and trackbars. Advanced GUIs should be built with TK, Wx, or QT. See this blog post to learn how to make an OpenCV GUI with TK.

Let's install GTK:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
$ sudo apt-get install libgtk-3-dev
```

I always recommend the following two libraries which will optimize various OpenCV functions:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
$ sudo apt-get install libatlas-base-dev gfortran
```

And finally, our last requirement is to install Python 3 headers and libraries:

```shell
Ubuntu 18.04: How to install OpenCV                                            Shell
```

```
$ sudo apt-get install python3-dev
```

Step #2: Download the official OpenCV source

At the time of this writing, the most recent release is OpenCV 3.4.1. These instructions should continue to work with future OpenCV 3.x versions as well.

Since we're continuing to work in the terminal, let's download the official OpenCV release using `wget` :

Ubuntu 18.04: How to install OpenCV Shell
```
$ cd ~
$ wget -O opencv.zip https://github.com/opencv/opencv/archive/3.4.1.zip
```

Followed by the `opencv_contrib` module:

Ubuntu 18.04: How to install OpenCV Shell
```
$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/3.4.1.zip
```

Note: *If your browser is cutting off the full command either use the "<>" button in the toolbar above to expand the code block or copy and paste the following URL: https://github.com/opencv/opencv_contrib/archive/3.4.1.zip*

So, what's the contrib repo?

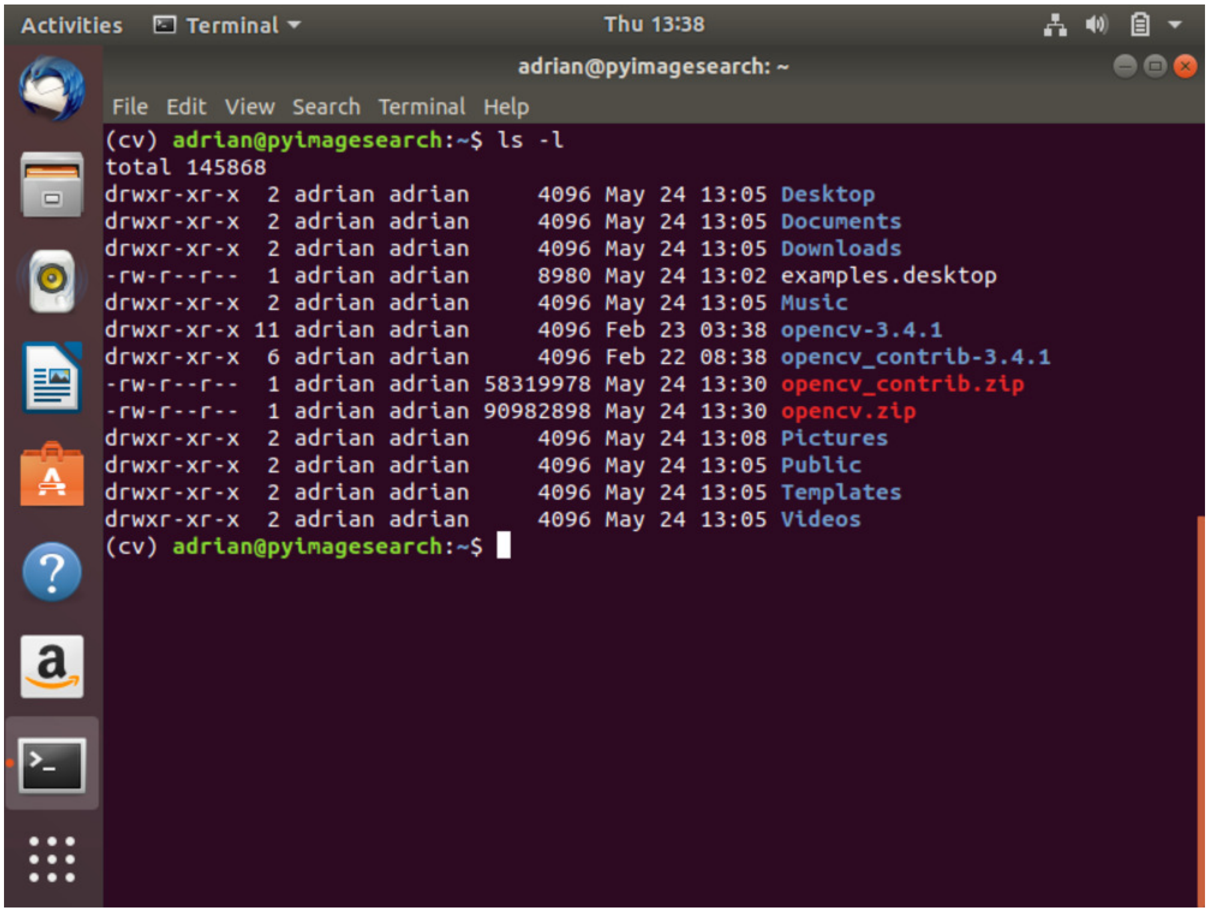
The contrib repository contains algorithms such as SIFT, SURF, and others. In the past, these implementations were included in the default installation of OpenCV 2.4; however, they were moved beginning with OpenCV 3+.

Modules that are actively being developed and/or modules that are patented (not free for commercial/industry use) are included in the contrib module. SIFT and SURF fall into this category. You can learn more about the thought process behind this move in the following blog post: *Where did SIFT and SURF go in OpenCV 3?*

Important: *Both* `opencv` *and* `opencv_contrib` *versions must be identical. Notice that both URLs point to 3.4.1. Feel free to install a different version while still using this guide — just be sure to update both URLs.*

Now, let's unzip the archives:

Ubuntu 18.04: How to install OpenCV Shell
```
$ unzip opencv.zip
$ unzip opencv_contrib.zip
```



```
(cv) adrian@pyimagesearch:~$ ls -l
total 145868
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Desktop
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Documents
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Downloads
-rw-r--r--  1 adrian adrian     8980 May 24 13:02 examples.desktop
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Music
drwxr-xr-x 11 adrian adrian     4096 Feb 23 03:38 opencv-3.4.1
drwxr-xr-x  6 adrian adrian     4096 Feb 22 08:38 opencv_contrib-3.4.1
-rw-r--r--  1 adrian adrian 58319978 May 24 13:30 opencv_contrib.zip
-rw-r--r--  1 adrian adrian 90982898 May 24 13:30 opencv.zip
drwxr-xr-x  2 adrian adrian     4096 May 24 13:08 Pictures
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Public
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Templates
drwxr-xr-x  2 adrian adrian     4096 May 24 13:05 Videos
(cv) adrian@pyimagesearch:~$
```

Figure 1: After downloading and unzipping `opencv` and `opencv_contrib`, our home directory listing should look similar to what is displayed in the terminal.

Step #3: Configure your Python 3 environment

The first step we're taking to configure our Python 3 development environment is to install pip, a Python Package Manager.

To install pip, simply enter the following in your terminal:

```
Ubuntu 18.04: How to install OpenCV                                    Shell
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
```

Making use of virtual environments for Python development

If you are familiar with my blog and install guides therein, the following statement might make me sound like a broken record but I'll repeat it anyway:

I use both virtualenv and virtualenvwrapper daily and you should too unless you have a very specific reason not to. These two Python packages facilitate creating independent Python environments for your projects.

It is a best practice to use virtual environments.

Why?

Virtual environments allow you to work on your projects in isolation without spinning up resource hogs such as VMs and Docker images (I definitely do use both VirtualBox and Docker — they have their place).

For example, maybe you have a Python + OpenCV project that requires an older version of scikit-learn (v0.14) but you want to keep using the latest version of scikit-learn (0.19) for all of your newer projects.

Using virtual environments, you could handle these two software version dependencies separately, something that is not possible using *just* the system install of Python.

If you would like more information about Python virtual environments take a look at this article on RealPython or read the first half of the this blog post on PyImageSearch.

Let's go ahead and install `virtualenv` and `virtualenvwrapper` now:

Ubuntu 18.04: How to install OpenCV Shell
```
$ sudo pip install virtualenv virtualenvwrapper
$ sudo rm -rf ~/get-pip.py ~/.cache/pip
```

To finish the install we need to update our `~/.bashrc` file.

Using a terminal text editor such as `vi` / `vim` or `nano` , add the following lines to your `~/.bashrc` :

Ubuntu 18.04: How to install OpenCV Shell
```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
source /usr/local/bin/virtualenvwrapper.sh
```

Alternatively, you can append the lines directly via bash commands:

Ubuntu 18.04: How to install OpenCV Shell
```
$ echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.bashrc
$ echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.bashrc
$ echo "export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3" >> ~/.bashrc
$ echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
```

Next, source the `~/.bashrc` file:

Ubuntu 18.04: How to install OpenCV Shell
```
$ source ~/.bashrc
```

Creating a virtual environment to hold OpenCV and additional packages

Ok, while that may have seemed like a lot of work, we're at the point where we can create your Python 3 virtual environment for OpenCV:

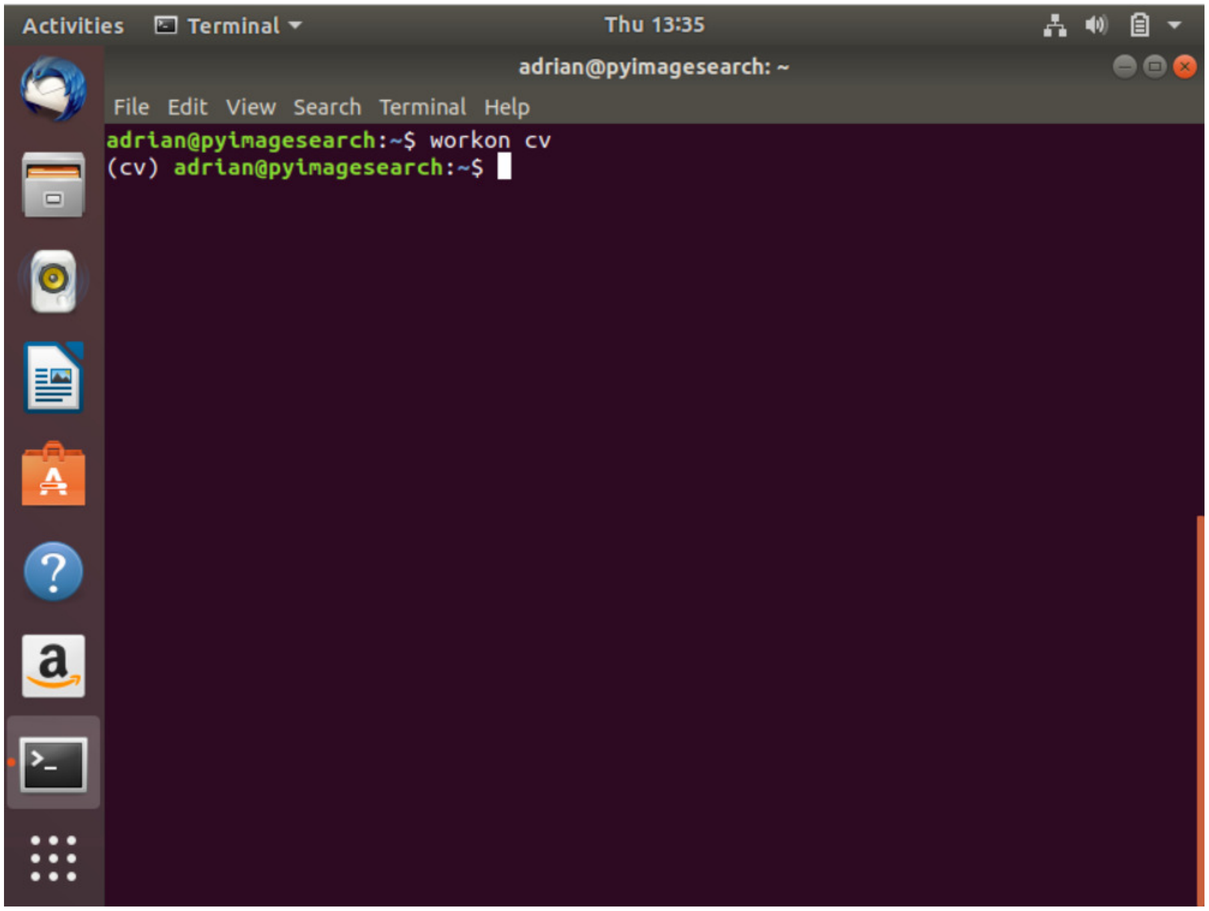
Ubuntu 18.04: How to install OpenCV Shell
```
$ mkvirtualenv cv -p python3
```

This line simply creates a Python 3 virtual environment named `cv` . You can name your environment(s) whatever you'd like — I like to keep them short and sweet while also providing enough information so I'll remember what they are for. You can have as many virtual environments on your system as you'd like!

Let's verify that we're in the cv environment by using the workon command:

```
Ubuntu 18.04: How to install OpenCV                                    Shell
$ workon cv
```

Figure 2 shows what your terminal will look like (assuming you haven't changed any bash prompt settings):



```
adrian@pyimagesearch:~$ workon cv
(cv) adrian@pyimagesearch:~$
```

Figure 2: If you see the `(cv)` at the beginning of the bash prompt, then your virtual environment is active and you're working "inside" the environment. You can now safely install OpenCV with correct Python bindings.

Install NumPy in your environment

Let's install our first package into the environment: NumPy. NumPy is a requirement for working with Python and OpenCV. We simply use pip (while the `cv` Python virtual environment is active):

```
Ubuntu 18.04: How to install OpenCV                                    Shell
$ pip install numpy
```

Step #4: Configure and compile OpenCV for Ubuntu 18.04

Now we're moving. We're ready to compile and install OpenCV.

Before we begin though, let's ensure that we're in the cv virtual environment:

```
Ubuntu 18.04: How to install OpenCV                                  Shell
$ workon cv
```

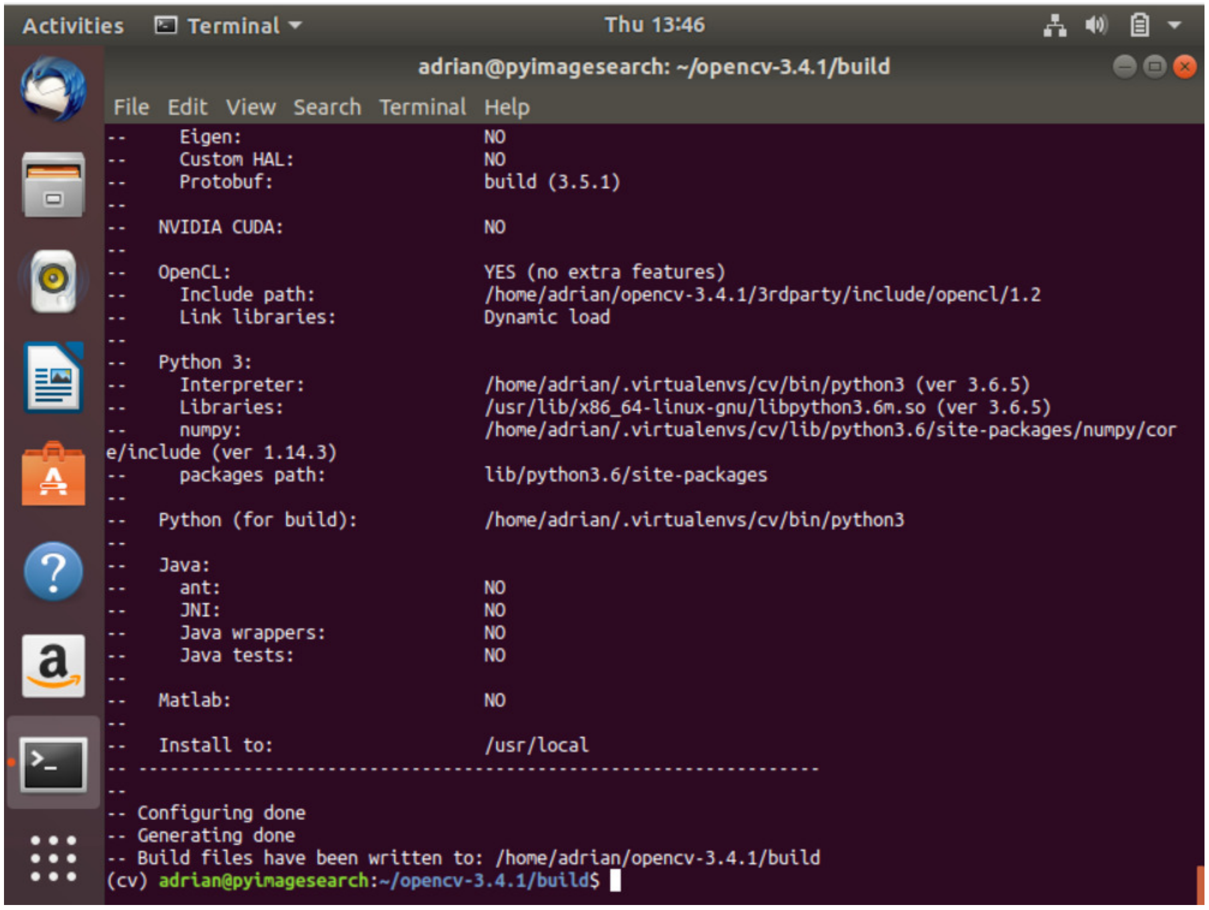
It is very important that the virtual environment is active (you are "inside" the virtual environment) which is why I keep reiterating it. If you are *not* in the `cv` Python virtual environment before moving on to the next step your build files will not be generated properly.

Configure OpenCV with CMake

Let's set up our OpenCV build using `cmake` :

```
Ubuntu 18.04: How to install OpenCV                                  Shell
$ cd ~/opencv-3.4.1/
$ mkdir build
$ cd build
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D INSTALL_C_EXAMPLES=OFF \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
    -D PYTHON_EXECUTABLE=~/.virtualenvs/cv/bin/python \
    -D BUILD_EXAMPLES=ON ..
```

I always recommend that you scroll through the CMake output and check to see if anything looks out of the ordinary. You won't see a "YES" marked next to every setting — that is normal. Be sure you don't see any errors or your compile may fail (warnings are okay).


Activities
Terminal
Thu 13:46
adrian@pyimagesearch: ~/opencv-3.4.1/build
File Edit View Search Terminal Help

```
--     Eigen:                       NO
--     Custom HAL:                  NO
--     Protobuf:                    build (3.5.1)
--
--   NVIDIA CUDA:                   NO
--
--   OpenCL:                        YES (no extra features)
--     Include path:                /home/adrian/opencv-3.4.1/3rdparty/include/opencl/1.2
--     Link libraries:              Dynamic load
--
--   Python 3:
--     Interpreter:                 /home/adrian/.virtualenvs/cv/bin/python3 (ver 3.6.5)
--     Libraries:                   /usr/lib/x86_64-linux-gnu/libpython3.6m.so (ver 3.6.5)
--     numpy:                       /home/adrian/.virtualenvs/cv/lib/python3.6/site-packages/numpy/cor
e/include (ver 1.14.3)
--     packages path:               lib/python3.6/site-packages
--
--   Python (for build):            /home/adrian/.virtualenvs/cv/bin/python3
--
--   Java:
--     ant:                         NO
--     JNI:                         NO
--     Java wrappers:               NO
--     Java tests:                  NO
--
--   Matlab:                        NO
--
--   Install to:                    /usr/local
-- -----------------------------------------------------------------
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/adrian/opencv-3.4.1/build
(cv) adrian@pyimagesearch:~/opencv-3.4.1/build$
```

Figure 3: To compile OpenCV for Ubuntu 18.04, we make use of CMake. The CMake tool will configure settings prior to compilation.

Take a moment to notice that only *"Python 3"* section is shown in the CMake output on Ubuntu 18.04 in **Figure 3**. This is by design as we are only compiling OpenCV with Python 3 support.

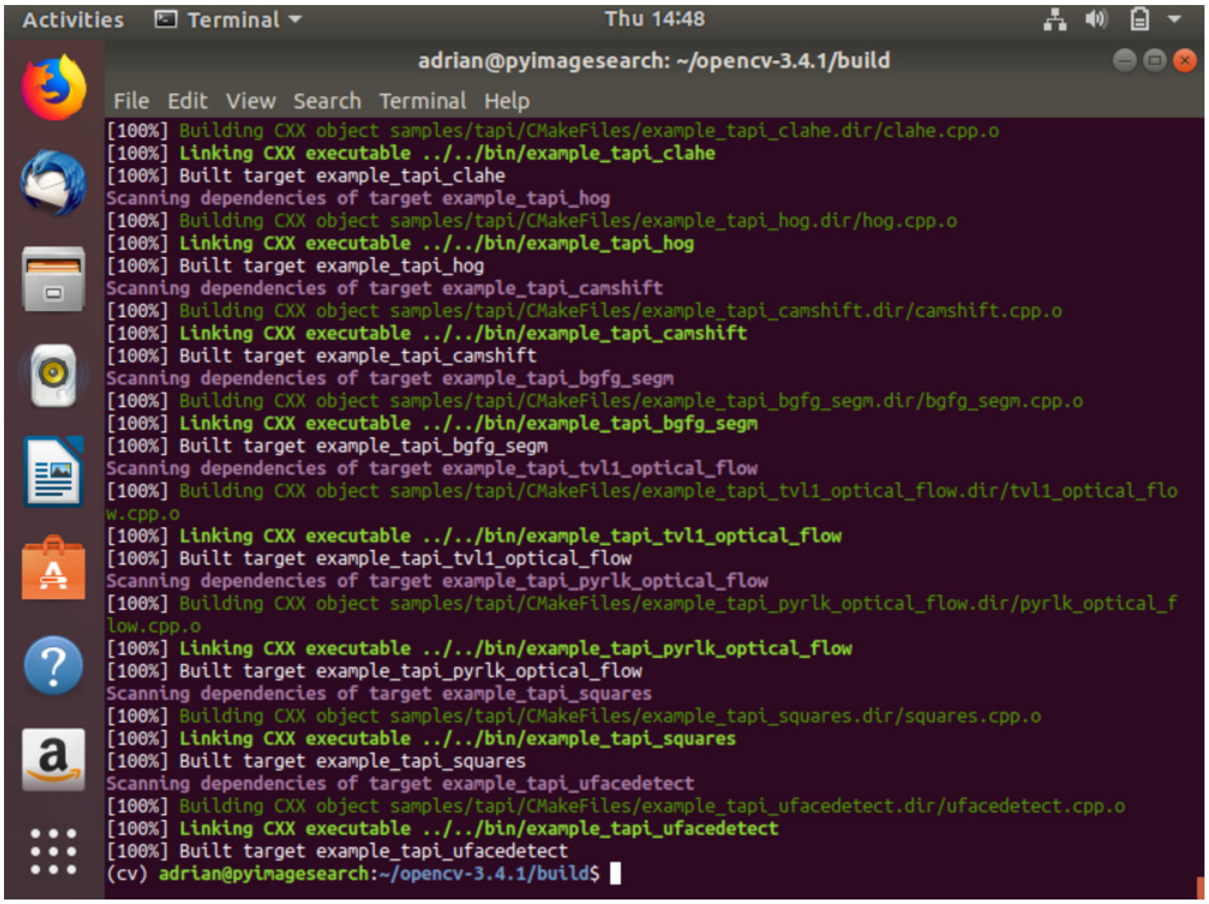
Note: *If you are encountering problems related to* `stdlib.h: No such file or directory` *during either the* `cmake` *or* `make` *phase of this tutorial you'll also need to include the following option to CMake:* `-D ENABLE_PRECOMPILED_HEADERS=OFF` *. In this case I would suggest deleting your build directory, re-creating it, and then re-running* `cmake` *with the above option included. This will resolve the* `stdlib.h` *error.*

Compiling OpenCV on Ubuntu 18.04

Let's compile OpenCV using `make` .

Depending on the number of processors/cores, you may be able to reduce compile time by altering the flag in the command. My computer has 4 cores, so I am using the `-j4` flag. You can update the numeral or leave the flag off altogether:

```
Ubuntu 18.04: How to install OpenCV                                    Shell
$ make -j4
```


Activities
Terminal
Thu 14:48
adrian@pyimagesearch: ~/opencv-3.4.1/build
File Edit View Search Terminal Help

```
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_clahe.dir/clahe.cpp.o
[100%] Linking CXX executable ../../bin/example_tapi_clahe
[100%] Built target example_tapi_clahe
Scanning dependencies of target example_tapi_hog
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_hog.dir/hog.cpp.o
[100%] Linking CXX executable ../../bin/example_tapi_hog
[100%] Built target example_tapi_hog
Scanning dependencies of target example_tapi_camshift
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_camshift.dir/camshift.cpp.o
[100%] Linking CXX executable ../../bin/example_tapi_camshift
[100%] Built target example_tapi_camshift
Scanning dependencies of target example_tapi_bgfg_segm
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_bgfg_segm.dir/bgfg_segm.cpp.o
[100%] Linking CXX executable ../../bin/example_tapi_bgfg_segm
[100%] Built target example_tapi_bgfg_segm
Scanning dependencies of target example_tapi_tvl1_optical_flow
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_tvl1_optical_flow.dir/tvl1_optical_flo
w.cpp.o
[100%] Linking CXX executable ../../bin/example_tapi_tvl1_optical_flow
[100%] Built target example_tapi_tvl1_optical_flow
Scanning dependencies of target example_tapi_pyrlk_optical_flow
[100%] Building CXX object samples/tapi/CMakeFiles/example_tapi_pyrlk_optical_flow.dir/pyrlk_optical_f
low.cpp.o
[100%] Linking CXX executable ../../bin/example_tapi_pyrlk_optical_flow
[100%] Built target example_tapi_pyrlk_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/example_tapi_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/example_tapi_ufacedetect
[100%] Built target example_tapi_ufacedetect
(cv) adrian@pyimagesearch:~/opencv-3.4.1/build$
```

Figure 4: To compile OpenCV with Python 3 on Ubuntu 18.04, we use `make`. Using make compiles OpenCV from source and is preferred over using package managers for installing OpenCV.

This process may take 30 minutes or longer, so go for a nice walk if you are able.

If your compile chokes and hangs, it may be due to a threading race condition. In the event you run into this problem, simply delete your `build` directory, recreate it, and re-run `cmake` and `make` . This time do not include the flag next to `make` .

Installing and verifying OpenCV

Upon a successful, 100% complete compile you can now install OpenCV:

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ sudo make install
$ sudo ldconfig
```

To verify the install, sometimes I like to enter the following command in the terminal:

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ pkg-config --modversion opencv
3.4.1
```

Step #5: Finish your Python+ OpenCV + Ubuntu 18.04 install

We've reached the last lap of the race so stick with it.

At this point, your Python 3 bindings for OpenCV should reside in the following folder:

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ ls /usr/local/lib/python3.6/site-packages/
cv2.cpython-36m-x86_64-linux-gnu.so
```

Let's rename them to simply `cv2.so` :

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ cd /usr/local/lib/python3.6/site-packages/
$ sudo mv cv2.cpython-36m-x86_64-linux-gnu.so cv2.so
```

Our last step is to sym-link our OpenCV `cv2.so` bindings into our `cv` virtual environment:

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ cd ~/.virtualenvs/cv/lib/python3.6/site-packages/
$ ln -s /usr/local/lib/python3.6/site-packages/cv2.so cv2.so
```

Step #6: Testing your OpenCV install on Ubuntu 18.04

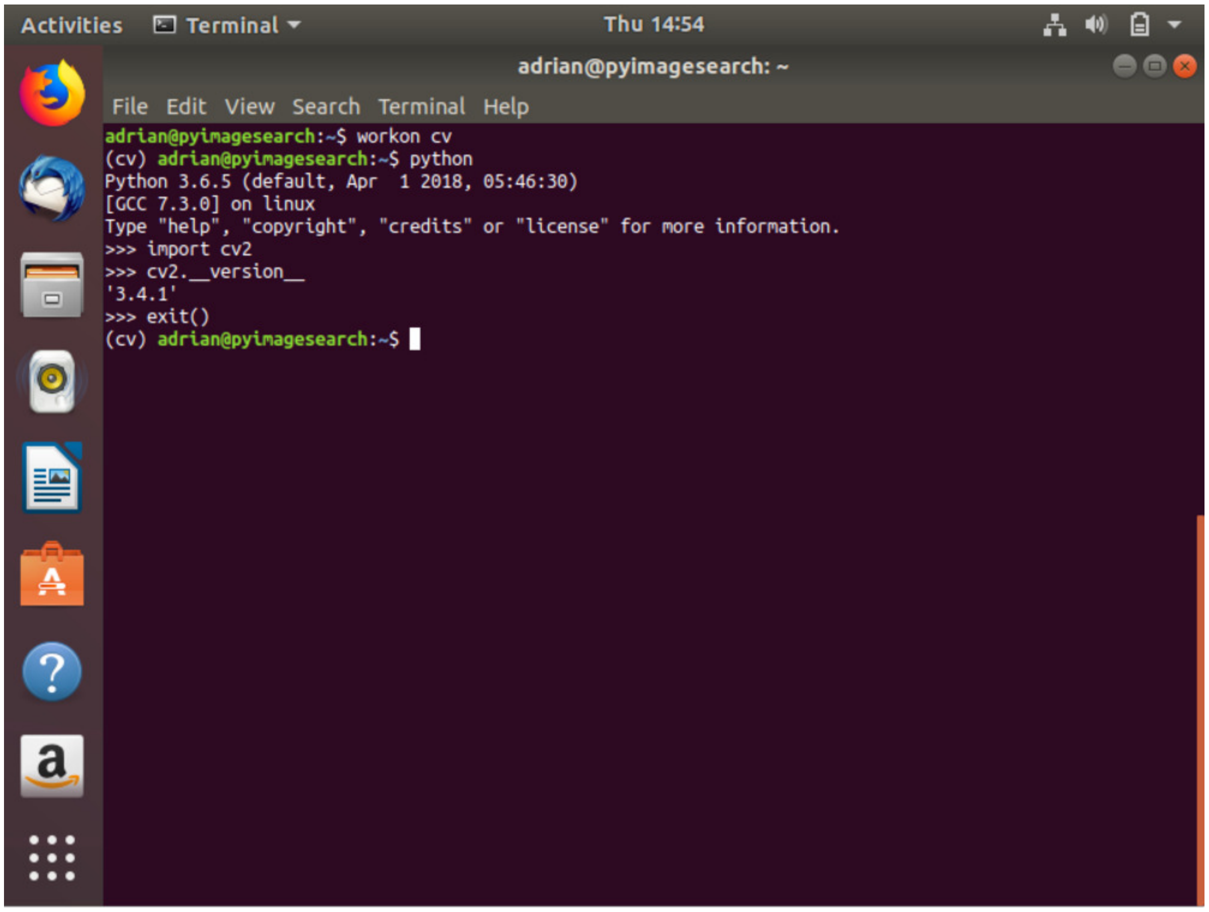
The race is done, but let's verify that we're firing on all cylinders.

To verify that our OpenCV + Ubuntu install is complete, I like to launch Python, import OpenCV, and query for the version (this is useful for sanity if you have multiple versions of OpenCV installed as well):

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ cd ~
$ workon cv
$ python
Python 3.6.5 (default, Apr 1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.1'
```

```
>>> quit()
```

Here's what it looks like on my system:



```
adrian@pyimagesearch:~$ workon cv
(cv) adrian@pyimagesearch:~$ python
Python 3.6.5 (default, Apr  1 2018, 05:46:30)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.1'
>>> exit()
(cv) adrian@pyimagesearch:~$
```

Figure 5: To verify that OpenCV is correctly installed and configured in our Python 3 virtual environment, I like to run the Python interpreter in the terminal. From there you can import OpenCV (`cv2`) and verify the version number matches what you intended to install.

Optionally, at this point, you can safely delete the zips and directories in your home folder:

```
Ubuntu 18.04: How to install OpenCV                                    Shell
$ cd ~
$ rm opencv.zip opencv_contrib.zip
$ rm -rf opencv-3.4.1 opencv_contrib-3.4.1
```

Troubleshooting your install (FAQ)

In this section, I address some of the common questions, problems, and issues that arise when installing OpenCV 3 with Python 3 on Ubuntu 18.04 LTS.

Q. Where is Python 2.7 on Ubuntu 18.04?

A. Python 3 is the default and what comes with Ubuntu 18.04. Python 2.7 users can manually install Python 2.7 at the end of **Step #1:**

```
Ubuntu 18.04: How to install OpenCV                                    Shell
$ sudo apt-get install python2.7 python2.7-dev
```

From there, when you create your virtual environment in **Step #3**, first install pip for Python 2.7:

```
Ubuntu 18.04: How to install OpenCV                                          Shell
$ sudo python2.7 get-pip.py
```

And then (also in **Step #3**) when you make your virtual environment, simply use the relevant Python version flag:

```
Ubuntu 18.04: How to install OpenCV                                         Python
$ mkvirtualenv cv -p python2.7
```

From there everything should be the same.

Q. Why can't I just pip to install OpenCV?

A. There are a number of pip-installable versions of OpenCV available depending on your operating system and architecture. The problem you may run into is that they may be compiled without various optimizations, image I/O support, video I/O support, and `opencv_contrib` support. Use them — but use them at your own risk. This tutorial is meant to give you the *full* install of OpenCV on Ubuntu 18.04 while giving you complete control over the compile.

Q. When I execute `mkvirtualenv` or `workon`, I encounter a "command not found error".

A. There a number of reasons why you would be seeing this error message, all of come from to **Step #3:**

1. First, make sure you have installed `virtualenv` and `virtualenvwrapper` properly using the `pip` package manager. Verify by running `pip freeze`, and ensure that you see both `virtualenv` and `virtualenvwrapper` in the list of installed packages.
2. Your `~/.bashrc` file may have mistakes. View the contents of your `~/.bashrc` file to see the proper `export` and `source` commands are present (check **Step #3** for the commands that should be appended to `~/.bashrc`).
3. You may have forgotten to `source` your `~/.bashrc`. Make sure you run `source ~/.bashrc` after editing it to ensure you have access to the `mkvirtualenv` and `workon` commands.

Q. When I open a new terminal, logout, or reboot my Ubuntu system, I cannot execute the `mkvirtualenv` or `workon` commands.

A. See **#2** from the previous question.

Q. When I try to import OpenCV, I encounter this message: `Import Error: No module named cv2`.

A. There are *multiple* reasons this could be happening and unfortunately, it is hard to diagnose. I recommend the following suggestions to help diagnose and resolve the error:

1. Make sure your `cv` virtual environment is active by using the `workon cv` command. If this command gives you an error, then see the first question in this FAQ.
2. Try investigating the contents of the `site-packages` directory in your `cv` virtual environment. You can find the `site-packages` directory in `~/.virtualenvs/cv/lib/python3.6/site-`

`packages/` depending on your Python version. Make sure (1) there is a `cv2.so` file in the `site-packages` directory and (2) it's properly sym-linked to a valid file.
3. Be sure to check the `site-packages` (and even `dist-packages`) directory for the system install of Python located in `/usr/local/lib/python3.6/site-packages/` , respectively. Ideally, you should have a `cv2.so` file there.
4. As a last resort, check in your `build/lib` directory of your OpenCV build. There *should* be a `cv2.so` file there (if both `cmake` and `make` executed without error). If the `cv2.so` file is present, *manually copy it* into both the system `site-packages` directory as well as the `site-packages` directory for the `cv` virtual environment.

What's possible with OpenCV?

The possibilities with OpenCV are truly endless.

Perhaps you're interested in tracking object movement:

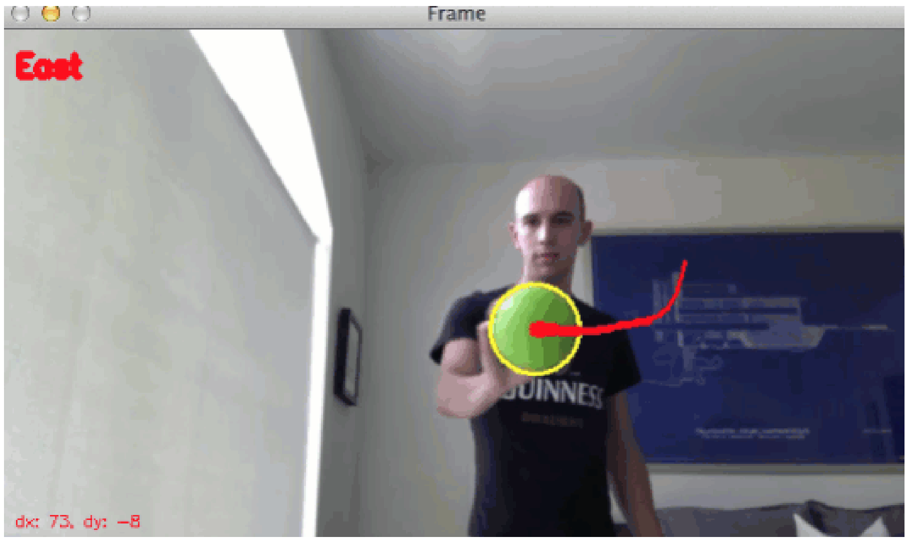


Figure 6: We can use OpenCV to track the motion of objects in real time!

Or maybe you'd like to use OpenCV + deep learning to detect faces in images/video? Trust me, *it's easier than it sounds!*

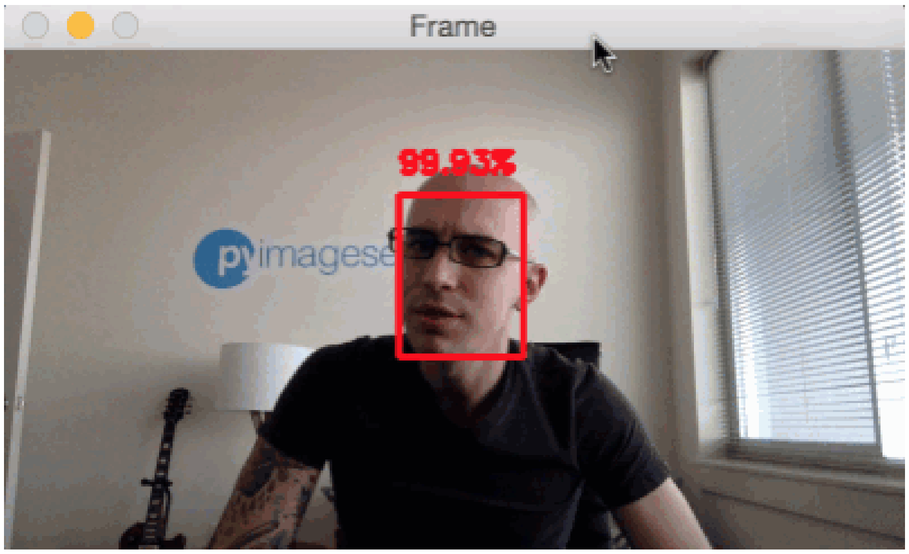


Figure 7: Detecting faces in realtime with OpenCV is easier than it sounds.

Or maybe you're looking for a beginner-level book, designed for newcomers to the world of computer vision?

If so, look no further than my book, *Practical Python and OpenCV + Case Studies.*

Practical Python and OpenCV
+ Case Studies
An Introductory, Example Driven Guide to Image Processing and Computer Vision
Dr. Adrian Rosebrock

Figure 8: *Practical Python and OpenCV + Case Studies* is a book meant for beginners but it is also useful for those that want to solidify their knowledge of the fundamentals. Be sure to give it a read!

Inside this book you'll:

- **Master the fundamentals of computer vision and image processing**, taught on a fun, hands-on manner.

- **Learn by doing**, including *visual examples* and *lots of code* (including line-by-line reviews, ensuring you understand *exactly* what's going on).
- Implement case studies, including **face detection**, **object tracking**, **handwriting recognition**, *and more!*

To learn more about the book (and how it can help you learn computer vision + OpenCV), ***just click here!***

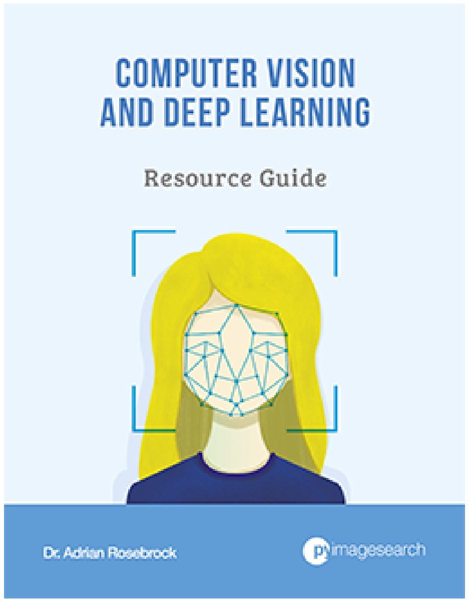
Summary

Today we installed OpenCV 3 with Python 3 bindings on Ubuntu 18.04 LTS.

I hope you found these instructions helpful on getting your own Ubuntu 18.04 machine configured with OpenCV 3.

If you're interested in learning more about OpenCV, computer vision, and image processing, ***be sure to enter your email address in the form below to be notified when new blog posts + tutorials are published!***

Resource Guide (it's totally free).



Enter your email address below to get my **free 17-page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF**. Inside you'll find my hand-picked tutorials, books, courses, and Python libraries to help you master computer vision and deep learning!

Your email address

DOWNLOAD THE GUIDE!

install, **opencv 3**, **python**, **python 3**, **ubuntu**

‹ An OpenCV barcode and QR code scanner with ZBar

Keras: Multiple outputs and multiple losses ›

115 Responses to *Ubuntu 18.04: How to install OpenCV*

El Barto May 28, 2018 at 10:56 am # REPLY

Hi Adrian! Is there a certain reason why you don't clone the OpenCV repo instead of fetching as an archive?

Adrian Rosebrock May 28, 2018 at 11:31 am # REPLY

You could do either, but downloading the official release archive tends to be easier for those who are not used to installing and configuring OpenCV on their systems (less commands to execute hear typically means less chance of causing a problem). If you cloned down the original repo the download would take longer and you would have the added step of checking out the specific release.

Sarkar May 28, 2018 at 11:32 am # REPLY

Hey Adrian, thanks for another awesome guide. Is it possible to use conda (instead of pip) in the above guide?

thanks

Adrian Rosebrock May 28, 2018 at 4:22 pm # REPLY

You should be able to but I haven't tested with Anaconda (I'm not an Anaconda user). The most important part will be during the "cmake" step where you'll want to make sure your "Python 3" libraries, interpreter, etc. point to your Anaconda version, not your system version — this is a mistake I've seen others make.

Anthony Park July 9, 2018 at 10:08 pm # REPLY

Hi, Adrian.
Could you be more specific about the cmake step for Anaconda users? I followed your kind instruction and compiled successfully, but I don't see site-packages created and assuming this is because I have Anaconda version of python. How can I appropriately make sure python 3 libraries, etc point to my Anaconda version?
Thank you 🙂

Adrian Rosebrock July 10, 2018 at 8:17 am # REPLY

Hey Anthony — I don't have any systems with Anaconda installed so I unfortunately cannot provide you with an example cmake command as I do not have the directory structure you would. The closest example I would have is this somewhat related tutorial on installing OpenCV on macOS. Inside the tutorial you'll see that I set the PYTHON3_LIBRARY and PYTHON3_INCLUDE_DIR values. You will need to do the same for Anaconda.

Adam May 28, 2018 at 11:33 am # REPLY

Hey Adrian, thanks a lot for another awesome tutorial!

1) Are you planning to release another one for the lucky GPU owners? 😉
2) Are you aware of any compatibility issues for those who are into your Deep Leaning book? 😉

Cheers,
Adam

Adrian Rosebrock May 28, 2018 at 4:21 pm # REPLY

1. I'm not sure what you mean. Are you asking for specific deep learning library install instructions?

2. I am not aware of any compatibility issues but I also haven't performed exhaustive experiments on Ubuntu 18.04 either.

Adam May 28, 2018 at 5:01 pm # REPLY

1. Yes. I was wondering if the instructions for Ubuntu 18.04 deep learning installation will be similar to 16.04. It took me literally weeks to get all lib versions (TF, cuDNN, Cuda) to work together on 16.04. It was a pain and I failed miserably when tried to wrap it all inside a clean Docker image ;/

2. That's cool Adrian

Adam May 28, 2018 at 5:02 pm # REPLY

Ooops, I can see your recommendation to stick to 16.04 for deep learning, sorry.

Adrian Rosebrock May 29, 2018 at 4:17 pm # REPLY

When it comes to deep learning environments, if it's not broken, don't fix it. Ubuntu 16.04 is perfectly legitimate for deep learning work. I would not be chomping at the bit to immediately switch to Ubuntu 18.04 for deep learning unless there was a very specific reason for doing so.

Thúlio May 28, 2018 at 12:31 pm # REPLY

Hello Adrian! I have a question about the new Ubuntu release. Have you been using Ubuntu 18.04 for deep learning? I've installed it and I'm planning on using TensorFlow and MXNet with GPU support enabled, but I'm afraid I was too eager to try the new release and didn't think about the lack of support for those packages since 18.04 is too new. Should I have sticked with 16.04? Or are TF and MXNet already good to go on 18.04?

Adrian Rosebrock May 28, 2018 at 4:19 pm # REPLY

As far as deep learning goes I would recommend sticking with Ubuntu 16.04 for the time being. Ubuntu 16.04 is more "mature" in terms of support of deep learning libraries. Keep in mind that Ubuntu 18.04 is essentially brand new and would be more likely of encountering issues. Deep learning libraries change rapidly though so I wouldn't get too concerned about it if you're already on 18.04.

Swan May 28, 2018 at 2:58 pm # REPLY

Hi Adrian!
Why do not you use pip install opencv-python ?

Swan May 28, 2018 at 3:00 pm # REPLY

Sorry, I'm not seen the FAQ section

Adrian Rosebrock May 28, 2018 at 4:18 pm # REPLY

No worries!

Doug May 29, 2018 at 12:10 am # REPLY

This article is excellent as always and timely for me, as I spent the last few days making opencv3.4.1 with dnn_modern and python 2.7 and 3.6 libs in a ubuntu16.04 docker. The long build time you noted above – combined with my copious mistakes insured many joyous repetitions of the cmake/make bits, until I got the opencv cmake build options just right. (And I needed? to build python cv libs and load them before I could see if they could find all their shared libs, in turn.) I've tried virtualenv but it was just too easy for me to break my ubuntu desktop with it. The docker approach seems to make up for my sloppiness with virtualenv. I often need a bib when I eat, though, so that might be a pattern..

Adrian Rosebrock May 29, 2018 at 4:15 pm # REPLY

Congrats on getting OpenCV installed on your Ubuntu system, Doug! I agree that Docker is a good use case here. Your rational for finding it too easy to break your main system is part of the reason why I really like cloud-based systems. Configure it once, snapshot it, and then delete + spin up a new instance if you ever break it!

Benjamin May 29, 2018 at 8:18 am # REPLY

Hi Adrian,

I am curious to know why you did not include the optimization in the cmake commands as previously referred to in your optimization post: https://www.pyimagesearch.com/2017/10/09/optimizing-opencv-on-the-raspberry-pi/

The two commands i am referring to are:
-D ENABLE_NEON=ON \
-D ENABLE_VFPV3=ON \

Really enjoying what you are putting out into the community.

Regards
Ben

Adrian Rosebrock May 29, 2018 at 4:14 pm # REPLY

Those optimizations are for ARM processors. Most Ubuntu users will likely be on non-ARM chipsets. If you are installing Ubuntu + OpenCV on an ARM processor you can of course use them.

Ben May 31, 2018 at 9:35 am # REPLY

Hi Adrian,

I am new to your blog but I have been working with success so far on the recent steps 1 & 2 of the Pokemon project (microsoft bing search & keras deep learning)…

Everything seems to be working for me with Anaconda 3.6 and Linux Mint 18.3 Cinnamon 64 bit with installing/using the deep learning libraries..

Im also new to Linux OS and an IT person recommended Mint to me for a place to start…

Would you recommend at all changing to Ubuntu at some point in time? Is there anything special about Linux Mint and using Anaconda? At some point in time I may actually have enough experience to require virtual environments with different versions of software…

Thanks much… Cheers.
Ben

Adrian Rosebrock June 5, 2018 at 8:37 am # REPLY

I'll be honest, I don't know much about Linux Mint so I don't think I'm qualified to say whether you should be using Ubuntu over Linux Mint. That said, I Ubuntu is often used for many deep learning projects and it's very user friendly. I would recommend it if you are using Linux systems.

Chris June 1, 2018 at 12:59 pm # REPLY

Thanks, Adrian. This is a very good tutorial for me to implement opencv on my new computer. I have a question now, how can I work in jupyter notebook and use this cv virtual environment. Thanks very much.

Adrian Rosebrock June 5, 2018 at 8:22 am # REPLY

You would need to install Jupyter into the "cv" Python virtual environment:

```
Python
$ workon cv
$ pip install jupyter
```

Antony Smith June 5, 2018 at 8:10 am # REPLY

Hi Adrian,
I bapassed the virtual env installation, but upon testing the installation (after all else installing correctly), I'm getting the Error:

ImportError: /usr/local/lib/python3.6/site-packages/cv2.so: undefined symbol: _ZTIN2cv3dnn19experimental_dnn_v35LayerE

Antony Smith June 5, 2018 at 8:13 am # REPLY

Okay nevermind, my bad.
Just opened a new terminal and retried and all is right with the world…
Thanks man!
You're the best!

Adrian Rosebrock June 5, 2018 at 8:25 am # REPLY

Congrats on resolving the issue, Anthony! 🙂

Andres June 5, 2018 at 8:25 am # REPLY

hi Adrian

Iḿ having trouble sinvce I can find my site-packages folder in the path you say, there is only the dist-packages directory in that path. I don't know where to look for it or if it is suposed to be only there.

thank you in advance

Andres

Adrian Rosebrock June 5, 2018 at 8:39 am # REPLY

I'm not sure why you would not have a site-packages directory and only a dist-packages directory. They should both be in the same subdirectory. Which step are you on?

Andres June 5, 2018 at 8:42 am # REPLY

I'm on step 5 making the link, all the make an make install has finished without error

Adrian Rosebrock June 7, 2018 at 3:18 pm # REPLY

If the "make install" worked you should check your "dist-packages" directory for the "cv2.so" bindings instead.

Ajith June 26, 2018 at 9:44 am #

I Have the same issue as well. @Andres did you find any solution or path to .so file?

Chemmy July 13, 2018 at 7:06 pm #

I had the same problem on a clean VM.

Try:

sudo apt-get install python3-dev libpython3.6-dev python3-numpy

Delete you build directory and make it again from step 4. You will have the site-packages folder after this and the cv2.cpython-36m-x86_64-linux-gnu.so file.

F.Y.I. I have no idea what I am doing. I just googled the answer and this worked.

AlexSeo October 1, 2018 at 7:36 am #

I solved by:
– creating manually an empty directory called “site-packages” into “/usr/local/lib/python3.6/”
– deleting “build” directory and starting again from step 4.

Andres June 6, 2018 at 8:41 am # REPLY

Hi Adrian

I have changed to Ubuntu 16.04 since im going tu use DNN and it seems is more reliable in that version, I have a doubt about CUDA integration with my project because I’m having trouble integrating the library in opencv, since im going to use tensorflow which one is better to compile CUDA with? or i will need it with booth of them?

Adrian Rosebrock June 7, 2018 at 3:09 pm # REPLY

You can use OpenCV with CUDA but it really depends on what you are doing. If your goal is to be developing DNNs you should install TensorFlow/Keras with CUDA support. OpenCV can be used to make predictions with some DNNs but for training you should be using TensorFlow/Keras.

Anand Setlur June 8, 2018 at 5:13 pm # REPLY

Hi Adrian, I had to modify your cmake script to get it to work with CUDA9.0 in Ubuntu18.04 to use gcc-6 and fix some other build errors by specifying the CUDA_NVCC_FLAGS and PYTHON2_EXECUTABLE options

```
export CC=/usr/bin/gcc-6
export CXX=/usr/bin/g++-6
mkdir -p build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CUDA_NVCC_FLAGS=–expt-relaxed-constexpr \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
-D PYTHON_EXECUTABLE=~/.virtualenvs/cv3/bin/python \
-D PYTHON2_EXECUTABLE=/usr/bin/python2 \
-D BUILD_EXAMPLES=ON ..
make -j4
```

Adrian Rosebrock June 13, 2018 at 6:11 am # REPLY

Thank you for sharing, Anand!

Robert August 4, 2018 at 10:58 pm # REPLY

Thanks Anand! I was still getting errors with your example above until I changed the CUDA_NVCC_FLAGS to:

```
-D CUDA_NVCC_FLAGS=”-D_FORCE_INLINES –expt-relaxed-constexpr” \
```

Robert August 4, 2018 at 11:51 pm # REPLY

This is the complete command I used:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CUDA_NVCC_FLAGS=”-D_FORCE_INLINES –expt-relaxed-constexpr” \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
-D PYTHON_EXECUTABLE=~/.virtualenvs/cv/bin/python \
-D BUILD_EXAMPLES=ON ..
```

artnect June 13, 2018 at 4:17 am # REPLY

Hi Adrian

I have gotten several failed process on building open cv using cmake

can you help me on that ?

the log file is too long i can't post it here.

Adrian Rosebrock June 13, 2018 at 5:23 am # REPLY

Can you create a GitHub Gist and link to it?

artnect June 13, 2018 at 6:37 am # REPLY

https://gist.github.com/AlirezaAghaie/b66542ef2016b6f7cad598d47abf2f4e

Adrian Rosebrock June 15, 2018 at 12:44 pm # REPLY

Looking at your cmake output I don't see any issues. Your "Python 3" section is correctly filled out and your build files have been generated. Keep in mind that many OpenCV configurations are optional. Just because cmake reports "not found" doesn't mean it's an error. You should be able to go ahead and compile OpenCV.

Skanda Kumar June 14, 2018 at 1:59 pm # REPLY

Hi Adrian, I am a novice user of OpenCV. My supervisor at my internship was very specific about installing OpenCV version 2.4. I installed the latest version of Ubuntu(18.04) in my system and I am trying to install the the required version of OpenCV but I am unable to. This was the most detailed post I could find on the internet and I would be grateful if you can help with getting OpenCV 2.4 running in my Ubuntu 18.04. Thanks in advance 🙂

Adrian Rosebrock June 14, 2018 at 3:58 pm # REPLY

Oh man, I imagine trying to install OpenCV 2.4 on Ubuntu 18.04 would be quite a pain. It's a hack but you may want to consider installing a VM with a much older version of Ubuntu 10.04/12.04 and then following the instructions from a previous previous Raspbian tutorial (Ubuntu and Raspbian are both Debian based). I do not know offhand what changes would need to be made for this tutorial and OpenCV 2.4 but I imagine there would be a few, mostly related to the specific apt-get packages installed. Best of luck with it!

clamytoe July 12, 2018 at 12:32 pm # REPLY

Might be too late now, but if you use Anaconda, you can install OpenCV 2.4.2. It probably won't have all of the bells and whistles like this full installation, but it should get you going.

Max Lee June 14, 2018 at 2:28 pm # REPLY

After following instructions, to specify here, I found that contrib modules, while being built, weren't actually getting installed… after digging around, found that you need to modify the cmake command where the extra modules are specified, to add a target for them.

Updated cmake command:
```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules ~/opencv-3.4.1\
-D PYTHON_EXECUTABLE=~/.virtualenvs/cv/bin/python \
-D BUILD_EXAMPLES=ON ..
```

This then allowed me to use the contrib modules for a few things that I was also compiling in C++.

Josef Matondang June 15, 2018 at 2:52 am # REPLY

Hello Adrian, do you know how to change the Python for build from python2.7 to the environment's python3

```
— Python 3:
— Interpreter: /home/josefmtd/.virtualenvs/cv/bin/python3 (ver 3.6.5)
— Libraries: /usr/lib/x86_64-linux-gnu/libpython3.6m.so (ver 3.6.5)
— numpy: /home/josefmtd/.virtualenvs/cv/lib/python3.6/site-packages/numpy/core/include (ver 1.14.5)
— packages path: lib/python3.6/site-packages
—
— Python (for build): /usr/bin/python2.7
```

Adrian Rosebrock June 15, 2018 at 12:01 pm # REPLY

The “Python (for build)” section is buggy and will not be correctly filled in. Ignore it. As long as your “Python 3” section is properly filled in you’ll be able to build the OpenCV bindings.

Hans November 23, 2018 at 9:18 am # REPLY

Hi Adrian, here’s the thing: the Python 3 part is missing from the terminal output, which only shows Python (for build), even though I followed the instructions in the same order. Additionally, the cv2.cpython-36m-x86_64-linux-gnu.so is missing (in fact, the site-packages folder itself was missing.) Could those two errors be related, and if yes, what would be the way to fix them?

Adrian Rosebrock November 25, 2018 at 9:10 am # REPLY

If the “Python 3” section is missing entirely then your .so bindings will not be built. Make sure you are in your Python virtual environment before running “cmake”.

Nathan Sieracki June 17, 2018 at 7:39 pm # REPLY

Josef,

I encountered the same issue. Adrian mentioned this is of no consequence to the buid as long as the Python 3 section is filled in, but I resolved the Python (for build) issue by executing the below cmake tags, which pointed the python executable directory to the Python3, (note the final tag):

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.4.1/modules \
-D PYTHON_EXECUTABLE=~/.virtualenvs/cv/bin/python3 \
-D BUILD_EXAMPLES=ON
-D BUILD_OPENCV_PYTHON3=YES
```

jeremy June 17, 2018 at 12:31 am # REPLY

Hi Adrian, i’ve followed your steps like a recipe book, but i have no idea why my Python (for build) is NO after finishing the cmake.

i've tried other methods but it isn't working.

i am able to access python3.6.5 by typing 'python' in my terminal. 🙁

jeremy June 17, 2018 at 2:33 am # REPLY

sorry i got it to work already.

jeremy June 17, 2018 at 2:33 am # REPLY

turns out that i forgot to change my envs name

Adrian Rosebrock June 19, 2018 at 8:55 am # REPLY

Congrats on resolving the issue, Jeremy!

CrashDummy June 20, 2018 at 8:46 pm # REPLY

Hi Adrian,

Just setup Ubuntu 18.04 and your blog post email popped up in my inbox, amazing 🙂 !

Sort of a side question, I've recently stumble across "pipenv" for setting up pip virtual environments, was wondering have you tried it and would love to hear your comments!

Adrian Rosebrock June 21, 2018 at 5:37 am # REPLY

I've played with pipenv on one system but I personally haven't put it through the ringer yet. I still really like virtualenv.

Jack June 26, 2018 at 12:54 am # REPLY

Hi Adrian
how can I install for c++? I have error for cmake

Adrian Rosebrock June 28, 2018 at 8:25 am # REPLY

If you follow the instructions on this post you will have the C++ OpenCV installed as well. As far as your CMake errors go I cannot provide any suggestions without knowing what your error is.

p pocock June 30, 2018 at 5:10 pm # REPLY

Thank You Guru Adrian! (guru means teacher)
Your clear and straight-up tutorial worked for my Thinkpad P71, even after I installed ubunutstudio 18.4 next to Windows 10 Pro.

No errors (a total rarity for me)!!

Where do I find examples?

Cheers Philip

Adrian Rosebrock July 3, 2018 at 8:38 am # REPLY

Congrats on getting OpenCV installed on your system, that's great! The PyImageSearch blog includes 250+ (free) tutorials on how to use computer vision and OpenCV in projects. If you are looking for a more structured, linear path to studying OpenCV I would recommend my book, Practical Python and OpenCV as well as the PyImageSearch Gurus course.

Microsoft Support UK July 4, 2018 at 6:46 pm # REPLY

I don't know how to configure and install the OpenCV in ubuntu. But after reading the post now I can install it in Ubuntu. I also need to install python for using the OpenCV.

Adrian Rosebrock July 5, 2018 at 6:17 am # REPLY

The method detailed in this post will show you how to install Python and OpenCV on your Ubuntu system.

clamytoe July 13, 2018 at 9:48 am # REPLY

Hi Adrian,

I'm using Anaconda with Linux Mint 19. and Just wanted to let others know that I was able to build OpenCV from source successfully by modifying the your instructions as follows:

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=~/anaconda3/envs/cv \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D INSTALL_C_EXAMPLES=OFF \
-D OPENCV_EXTRA_MODULES_PATH=~/Downloads/opencv_contrib-3.4.2/modules \
-D PYTHON_EXECUTABLE=~/anaconda3/envs/cv/bin/python \
-D BUILD_EXAMPLES=ON ..
make -j4
make install
ldconfig -n ~/anaconda3/envs/cv/lib
```

This of course assumes that Anaconda was installed in the default location. I put all of your instructions into a script, but I have not tested it.

https://gist.github.com/clamytoe/3424d0201bba0073ff1313e80ffc6328

Adrian Rosebrock July 17, 2018 at 8:10 am # REPLY

Thanks so much for sharing!

c_r_p July 21, 2018 at 3:19 pm # REPLY

Thank you clamytoe. This helped me to compile for Miniconda3 on Ubuntu 18.04.

Javier Hagua July 17, 2018 at 7:36 am # REPLY

Hi again Adrian,
Seems like it was some problem in the cmake part. I repeat the entire process and now it seems ok.
Thanks for your private reply
BR

Adrian Rosebrock July 17, 2018 at 8:11 am # REPLY

I'm glad it worked for you Javier!

Abdulrhman Alkhodiry July 21, 2018 at 9:32 am # REPLY

Thanks, Adrian for this guide I was able to build OpenCV using cmake params.

using Anaconda envs

Gist: https://gist.github.com/zeroows/4b85151f11536ffb6bafe10ebc6dc325

Adrian Rosebrock July 25, 2018 at 8:35 am # REPLY

Thank you for sharing, Abdulrhman!

Alison August 5, 2018 at 12:25 am # REPLY

hey how exactly are you supposed to edit the ~/.bashrc if workon command is not found? which part are you supposed to change?

and if i chose not to use virtual, then what should i change?

Adrian Rosebrock August 7, 2018 at 6:55 am # REPLY

You can use your favorite text editor to edit your .bashrc file. As far as changing it goes, without knowing what commands you've run on what errors you've encountered it's hard for me to give any suggestions. Try to ensure your .bashrc file matches what I've included on the PyImageSearch blog. If you choose to not use virtual environments you can skip all 'workon' and 'mkvirtualenv' commands.

Abhishek Singh August 14, 2018 at 12:05 am # REPLY

In step 5, I am getting an error as unknown directory and i dont see any site-packages folder for python3.6
Any help?

Jacob September 17, 2018 at 6:04 pm # REPLY

I am getting the same error.

Jacob September 17, 2018 at 6:12 pm # REPLY

All I have found is a dist-packages folder. Have you solved the issue yet?

Adrian Rosebrock September 17, 2018 at 7:26 pm # REPLY

Hey Jacob, no need to submit three comments to the same post. I went ahead and removed the first, redundant one. If you have a "dist-packages" directory, which it sounds like you do, you can use that one instead.

Anuj Pahade August 19, 2018 at 8:46 am # REPLY

I'm getting this error

mkvirtualenv cv -p python3
ERROR: virtualenvwrapper could not find /usr/local/bin/virtualenv in your path

which python
/home/acer/anaconda3/bin/python

Help needed

Adrian Rosebrock August 22, 2018 at 10:07 am # REPLY

This tutorial was not designed for Anaconda. I would suggest using my exact install instructions.

Jorge Paredes August 23, 2018 at 8:28 pm # REPLY

I compiled opencv 3.4.2 (I need it for your saliency-detection post 🙂), I follow your detailed instructions step by step and it works like a charm.

Super blog!

Thanks, Adrian.

Adrian Rosebrock August 24, 2018 at 8:35 am # REPLY

That's great news! Congrats on getting OpenCV installed, Jorge 🙂

manan August 24, 2018 at 10:56 am # REPLY

thanks for this…..

Hector Hernandez August 24, 2018 at 1:20 pm # REPLY

Great tutorial It saved me another painful nightmarish experience compiling opencv again….

Just for the record, do NOT compile opencv in FAT32 systems, you might end up getting errors like this as the system is not able to set the executable bit on FAT 32 filesystems:

```
Linking CXX shared library ../../lib/libopencv_core.so
CMake Error: cmake_symlink_library: System Error: Operation not permitted
CMake Error: cmake_symlink_library: System Error: Operation not permitted
make[2]: *** [lib/libopencv_core.so.2.4.9] Error 1
make[1]: *** [modules/core/CMakeFiles/opencv_core.dir/all] Error 2
```

Gustavo September 3, 2018 at 3:35 pm # REPLY

How can I launch idle Python in a virtual environment?

Adrian Rosebrock September 5, 2018 at 8:52 am # REPLY

Unfortunately you cannot. Python IDLE does not respect Python virtual environments. You should look into using Jupyter Notebooks instead.

Andrea September 4, 2018 at 5:11 am # REPLY

I had an issue at compile time with xfeatures2d. It threw a fatal error about a missing file “fatal error: opencv2/xfeatures2d/cuda.hpp: No such file or directory”.
I solved it with a brute force approach: removing xfeature2d directory from contrib modules “rm -rf ~/opencv_contrib-3.4.1/modules/xfeatures2d/”.
Maybe not the best way to solve the problem, but it compiles now.
I suppose that xfeatures2d is needed for SIFT/SURF algorithms, so don’t remove this module unless you are not planning to use them!

Eli September 6, 2018 at 11:04 am # REPLY

Conflicts occur with Yolo v3. After googling it seems that it is better installing 3.4.0, but still not succeeding in compiling darknet yolo v3 with opencv. Anyone?

Adrian Rosebrock September 11, 2018 at 8:41 am # REPLY

Just to clarify, you're compiling DarkNet from source but OpenCV itself installed okay?

Abdul September 22, 2018 at 6:14 am # REPLY

Hello.
Thanks for this great write up.
I did follow the instruction without any problems.
But, cv2.cpython-36m-x86_64-linux-gnu.so is not in /usr/local/lib/python3.6/site-packages/

Adrian Rosebrock October 8, 2018 at 1:03 pm # REPLY

Can you check and see if it's in the `build/lib` directory? If so, you can manually copy it over.

Hélder Ribeiro September 26, 2018 at 6:39 am # REPLY

You are just the best Adrian, I hope you are having a great honeymoon! Thanks for everything you

Adrian Rosebrock October 8, 2018 at 12:39 pm # REPLY

Thanks so much Hélder 🙂

TACO_CODER October 1, 2018 at 10:42 am # REPLY

Hello, sir thanks for your tutorial, what if i want to develop using CUDA? Do i need to do it on C++? can you please assist with the set up for CUDA please?

Thanks in advance

Adrian Rosebrock October 8, 2018 at 10:46 am # REPLY

You can follow my guide on compiling OpenCV with CUDA support.

Carlos October 2, 2018 at 4:35 pm # REPLY

Excelent! I followed all the steps and it worked, Thanks!

Adrian Rosebrock October 8, 2018 at 10:32 am # REPLY

Awesome, congrats on getting OpenCV installed on your Ubuntu machine, Carlos! 🙂

Crowmagnus October 7, 2018 at 7:33 pm # REPLY

There’s no site site-packages in my 3.6, there is in 2.7 but it’s empty.

Jho October 9, 2018 at 3:17 pm # REPLY

First I had problem with not having /site-packages/ -folder, but I figured out I need to be installing opencv inside that virtualenv cv.(using ‘workon cv’) and then it worked fine.

Second problem came, when I didn’t have ~/.virtualenvs/ -folder.
It was located in root (/), not at home (~), which confused me when trying to link the cv2.so files. But linking to there it worked.

Adrian Rosebrock October 12, 2018 at 9:26 am # REPLY

Hey Jho — it sounds like you may have accidentally ran all these steps in a root shell versus a user shell, that is likely what the discrepancy is.

Beckyli October 10, 2018 at 8:32 pm # REPLY

I met the same problem.

Adrian Rosebrock October 12, 2018 at 9:09 am # REPLY

Which problem is that, Beckyli?

trico October 12, 2018 at 9:19 am # REPLY

I tried this but its not working. Mainly:
1) For the CMAKE options should we pass python libraries and python include dirs? Cmake complained about not finding them
2) It found the python interpreter I gave but at the end I cannot find the .so file you mentioned. I do not have the site-packages folder anyway but rather a dist-packages one.
3) As I cannot do the linking of the bindings then I cannot use open cv in the virtual environment.

Thank you!

Adrian Rosebrock October 12, 2018 at 9:33 am # REPLY

1. As long as you are in the "cv" Python virtual environment your Python libraries and include dirs should automatically be found cmake. You can pass them in manually but that should be unnecessary.

2. You need the *entire* section filled out correctly, not just the interpreter. The include dirs, libraries, interpreter, and NumPy sections need to be filled out.

3. See above.

Jay October 14, 2018 at 4:10 pm # REPLY

Hi Adrian,
I followed all the steps properly but I am now stuck at the build step. The error is,
"CMake Error: The source directory "/home/jay/opencv-3.4.1″ does not appear to contain CMakeLists.txt."
I tried using the additional option -D for No such directory or file but it doesn't seems to work as I am getting same error with or without additional -D option. If you can help it would be really great.

Adrian Rosebrock October 16, 2018 at 8:39 am # REPLY

You'll want to scroll up through your "cmake" output to find out exactly what the error is. The error you're reporting in the comment is just a message from cmake saying the build files could not be created — the actual error is farther up in the output.

Bartolo October 29, 2018 at 7:31 am # REPLY

There is a typo in the line "cmake -D CMAKE_BUILD_TYPE=RELEASE \". It should be "cmake .. -D CMAKE_BUILD_TYPE=RELEASE \" because the previous command was "cd build".

Bartolo October 31, 2018 at 10:33 am # REPLY

I was wrong. There is ".." at the end of the command.

Jay October 19, 2018 at 2:40 pm # REPLY

Hi Adrian,
That's what the problem is there is no output or anything just the error the build doesn't even gets started. I even tried to start from scratch like setting up the vm again but still the error remains same.

Adrian Rosebrock October 20, 2018 at 7:27 am # REPLY

Are you sure you're in the "build" directory when you execute the "cmake" command? It almost sounds like your directory structure is incorrect. Delete the files and restart from where we download the OpenCV source code to our system.

Rahat October 27, 2018 at 2:19 pm # REPLY

How can i add .virtualenvs on my pycharm ide?i mean how can run program terminal & pycharm in both side?

Adrian Rosebrock October 29, 2018 at 1:31 pm # REPLY

You'll want to go to your project preferences and set then the "Python Interpreter" to point to whatever Python virtual environment you are using.

Peter Lunk October 28, 2018 at 11:32 am # REPLY

Please can you tell me/us ho to include GStreamer support in the opencv installation ?

Patrick November 2, 2018 at 5:05 pm # REPLY

Hello Adrian

Are you aware that we need to add this switch in CMAKE now to get the "non-free" modules ?

-D OPENCV_ENABLE_NONFREE:BOOL=ON \

Best Regards

Adrian Rosebrock November 6, 2018 at 1:31 pm # REPLY

Thanks Patrick, I'll check on this.

Leave a Reply

Name (required)

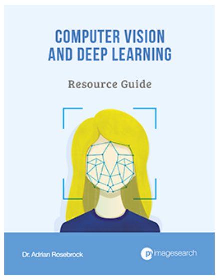
Email (will not be published) (required)

Website

SUBMIT COMMENT

Search...

Resource Guide (it's totally free).



Get your **FREE 17 page Computer Vision, OpenCV, and Deep Learning Resource Guide PDF.** Inside you'll find my hand-picked tutorials, books, courses, and libraries to help you master CV and DL.

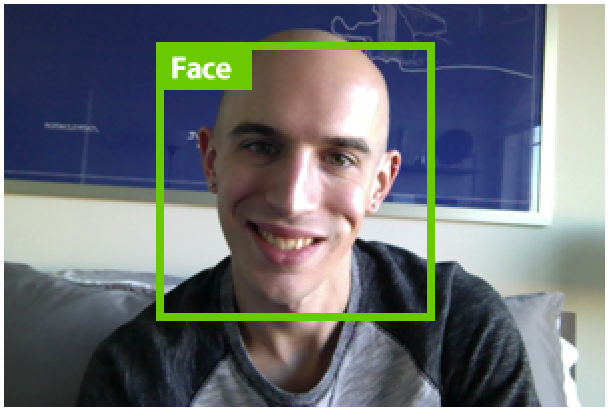
Download for Free!

Deep Learning for Computer Vision with Python Book — OUT NOW!

You're interested in deep learning and computer vision, *but you don't know how to get started.* Let me help. **My new book will teach you all you need to know about deep learning.**

CLICK HERE TO MASTER DEEP LEARNING

You can detect faces in images & video.



Are you interested in **detecting faces in images & video?** But **tired of Googling for tutorials** that *never work?* Then let me help! I guarantee that my new book will turn you into a **face detection ninja** by the end of this weekend. Click here to give it a shot yourself.

CLICK HERE TO MASTER FACE DETECTION

PyImageSearch Gurus: NOW ENROLLING!

The PyImageSearch Gurus course is *now enrolling!* Inside the course you'll learn how to perform:

- Automatic License Plate Recognition (ANPR)
- Deep Learning
- Face Recognition
- *and much more!*

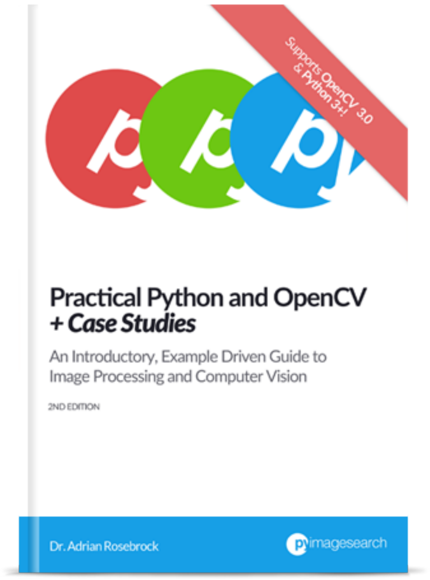
Click the button below to learn more about the course, take a tour, and get 10 (FREE) sample lessons.

TAKE A TOUR & GET 10 (FREE) LESSONS

Hello! I'm Adrian Rosebrock.

I'm an entrepreneur and Ph.D who has launched two successful image search engines, ID My Pill and Chic Engine. I'm here to share my tips, tricks, and hacks I've learned along the way.

Learn computer vision in a single weekend.


Supports OpenCV 3.0 & Python 3+!
Practical Python and OpenCV
+ Case Studies
An Introductory, Example Driven Guide to Image Processing and Computer Vision
2ND EDITION
Dr. Adrian Rosebrock
pyimagesearch

Want to learn computer vision & OpenCV? I can teach you in a **single weekend**. I know. It sounds crazy, but it's no joke. My new book is your **guaranteed, quick-start guide** to becoming an OpenCV Ninja. So why not give it a try? Click here to become a computer vision ninja.

CLICK HERE TO BECOME AN OPENCV NINJA

Subscribe via RSS

Never miss a post! Subscribe to the PyImageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

Install guide: Raspberry Pi 3 + Raspbian Jessie + OpenCV 3
APRIL 18, 2016

Raspbian Stretch: Install OpenCV 3 + Python on your Raspberry Pi
SEPTEMBER 4, 2017

Install OpenCV and Python on your Raspberry Pi 2 and B+
FEBRUARY 23, 2015

Home surveillance and motion detection with the Raspberry Pi, Python, OpenCV, and Dropbox
JUNE 1, 2015

Ubuntu 16.04: How to install OpenCV
OCTOBER 24, 2016

How to install OpenCV 3 on Raspbian Jessie
OCTOBER 26, 2015

Basic motion detection and tracking with Python and OpenCV
MAY 25, 2015

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.
© 2018 PyImageSearch. All Rights Reserved.