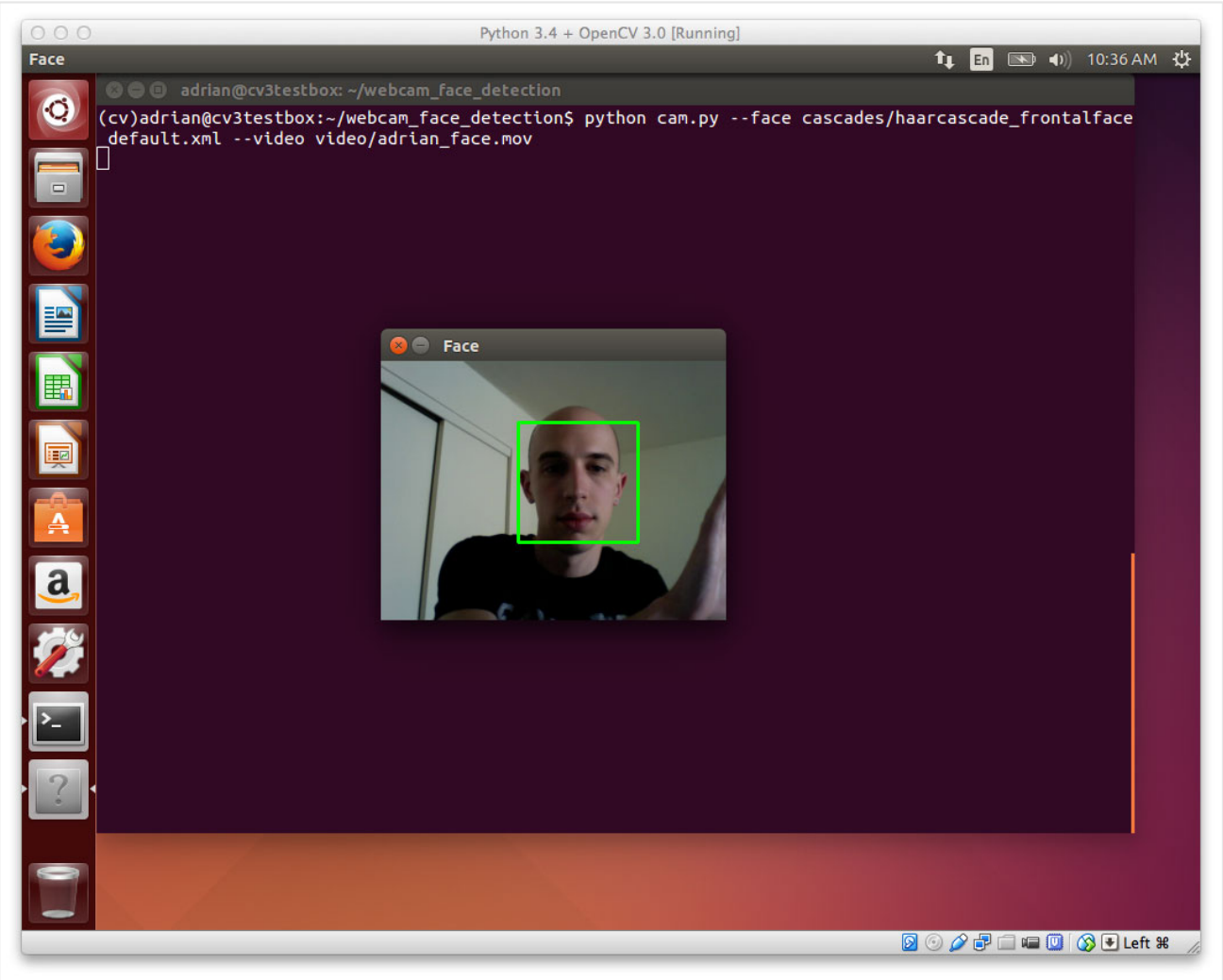
pyimagesearch
be awesome at building image search engines

Install OpenCV 3.0 and Python 3.4+ on Ubuntu

by **Adrian Rosebrock** on July 20, 2015 in **OpenCV 3**, **Tutorials**



A couple weeks ago I provided step-by-step install instructions to setup OpenCV 3.0 and Python 2.7+ on your Ubuntu machine.

However, one of the huge benefits of migrating to OpenCV 3.0 is the new Python 3.4+ support. In the previous 2.4.X releases of OpenCV, only Python 2.7+ was supported. But now, we

can *finally* leverage Python 3.4+ in our new projects.

In the remainder of this blog post, I'll detail how to install OpenCV 3.0 with Python 3.4+ bindings on your Ubuntu 14.04+ system. If you have followed along from the previous tutorial, you'll notice that many of the steps are the same (or at least very similar), so I have condensed this article a bit. That said, be sure to pay special attention when we start working with CMake later in this tutorial to ensure you are compiling OpenCV 3.0 with Python 3.4+ support!

How to Install OpenCV 3.0 and Python 3.4+ on Ubuntu

A few weeks ago I covered how to install OpenCV 3.0 and Python 2.7+ on Ubuntu, and while this was a great tutorial (since many of us are still using Python 2.7), I think it's really missing out on one of the major aspects of OpenCV 3.0 — ***Python 3.4+ support!***

That's right, up until the v3.0 release, OpenCV *only* provided bindings to the Python 2.7 programming language.

And for many of us, that was okay. As scientific developers and researchers, it's a pretty standard assumption that we'll be sequestered to Python 2.7.

However, that's starting to change. Important scientific libraries such as NumPy, SciPy, and scikit-learn are now providing Python 3 support. And now OpenCV 3.0 joins the ranks!

In general, you'll find this tutorial very similar to the previous one on installing OpenCV 3.0 and Python2.7 on Ubuntu, so I'm going to condense my explanations of each of the steps as necessary. If you would like to full explanation of each step, please refer to the previous OpenCV 3.0 article. Otherwise, simply follow along with this tutorial and you'll have OpenCV 3.0 and Python 3.4+ installed on your Ubuntu system in less than 10 minutes.

Step 1: Install prerequisites

Upgrade any pre-installed packages:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ sudo apt-get
2 $ sudo apt-get upgrade
```

Install developer tools used to compile OpenCV 3.0:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ sudo apt-get install build-essential cmake git pkg-config
```

Install libraries and packages used to read various *image formats* from disk:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
```

```
$ sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-dev
```

Install a few libraries used to read *video formats* from disk:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

Install GTK so we can use OpenCV's GUI features:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
$ sudo apt-get install libgtk2.0-dev
```

Install packages that are used to optimize various functions inside OpenCV, such as matrix operations:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
$ sudo apt-get install libatlas-base-dev gfortran
```

Step 2: Setup Python (Part 1)

Let's get `pip`, a Python package manager, installed for Python 3:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python3 get-pip.py
```

Note that I have ***specifically indicated*** `python3` when installing `pip`. If you do not supply `python3`, then Ubuntu will attempt to install `pip` on your Python 2.7 distribution, which is not our desired intention.

Alright, so I've said it before on the PyImageSearch blog, and I'll see it again. ***You should really be using virtual environments for Python development!***

We'll be using virtualenv and virtualenvwrapper in this tutorial. These packages allow us to create ***entirely separate and independent Python environments***, ensuring that we don't junk up our system Python install (and more importantly, so we can have a separate Python environment for each of our projects).

Let's use our fresh `pip3` install to setup `virtualenv` and `virtualenvwrapper`:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
$ sudo pip3 install virtualenv virtualenvwrapper
```

Again, notice how I am specifying `pip3` instead of just `pip` — I'm just making it explicitly obvious that these packages should be installed for Python 3.4.

Now we can update our `~/.bashrc` file (place at the bottom of the file):

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
# virtualenv and virtualenvwrapper
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
export WORKON_HOME=$HOME/.virtualenvs
```

```
4 source /usr/local/bin/virtualenvwrapper.sh
```

Notice how I am pointing `VIRTUALENVWRAPPER_PYTHON` to where our Python 3 binary lives on our Ubuntu system.

To make these changes take affect, you can either open up a new terminal or reload your `~/.bashrc` file:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ source ~/.bashrc
```

Finally, let's create our `cv` virtual environment where we'll be doing our computer vision development using OpenCV 3.0 and Python 3.4:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ mkvirtualenv cv
```

Step 2: Setup Python (Part 2)

We're halfway done setting up Python. But in order to compile OpenCV 3.0 with Python 3.4+ bindings, we'll need to install the Python 3.4+ headers and development files:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ sudo apt-get install python3.4-dev
```

OpenCV represents images as NumPy arrays, so we need to install NumPy into our `cv` virtual environment:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ pip install numpy
```

If you end up getting a *Permission denied* error related to pip's `.cache` directory, like this:

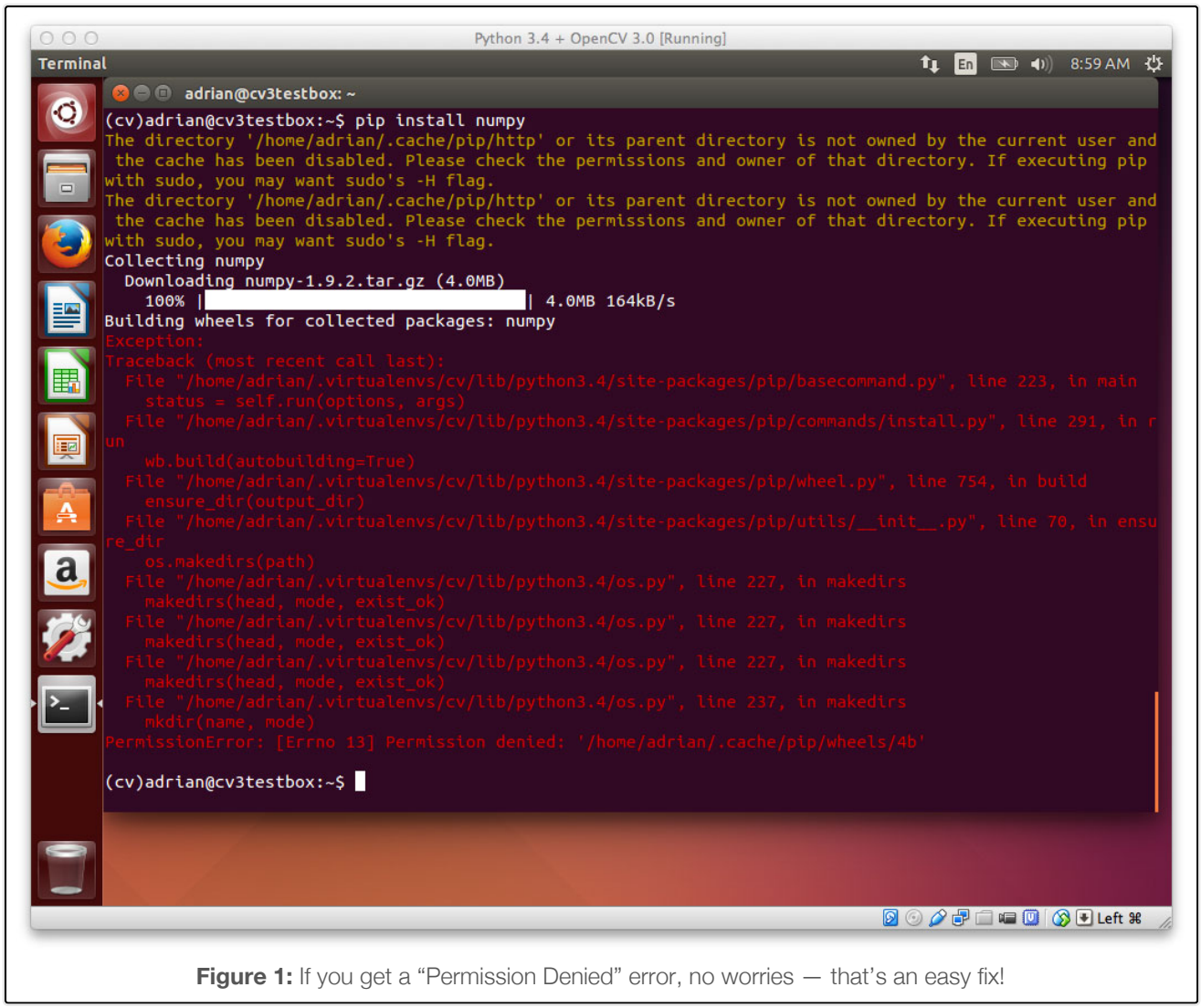


Figure 1: If you get a "Permission Denied" error, no worries — that's an easy fix!

Then simply delete the cache directory and re-run the NumPy install command:

Install OpenCV 3.0 and Python 3.4+ on Ubuntu — Shell

```
1 $ sudo rm -rf ~/.cache/pip/
2 $ pip install numpy
```

And you should now have a nice clean install of NumPy:

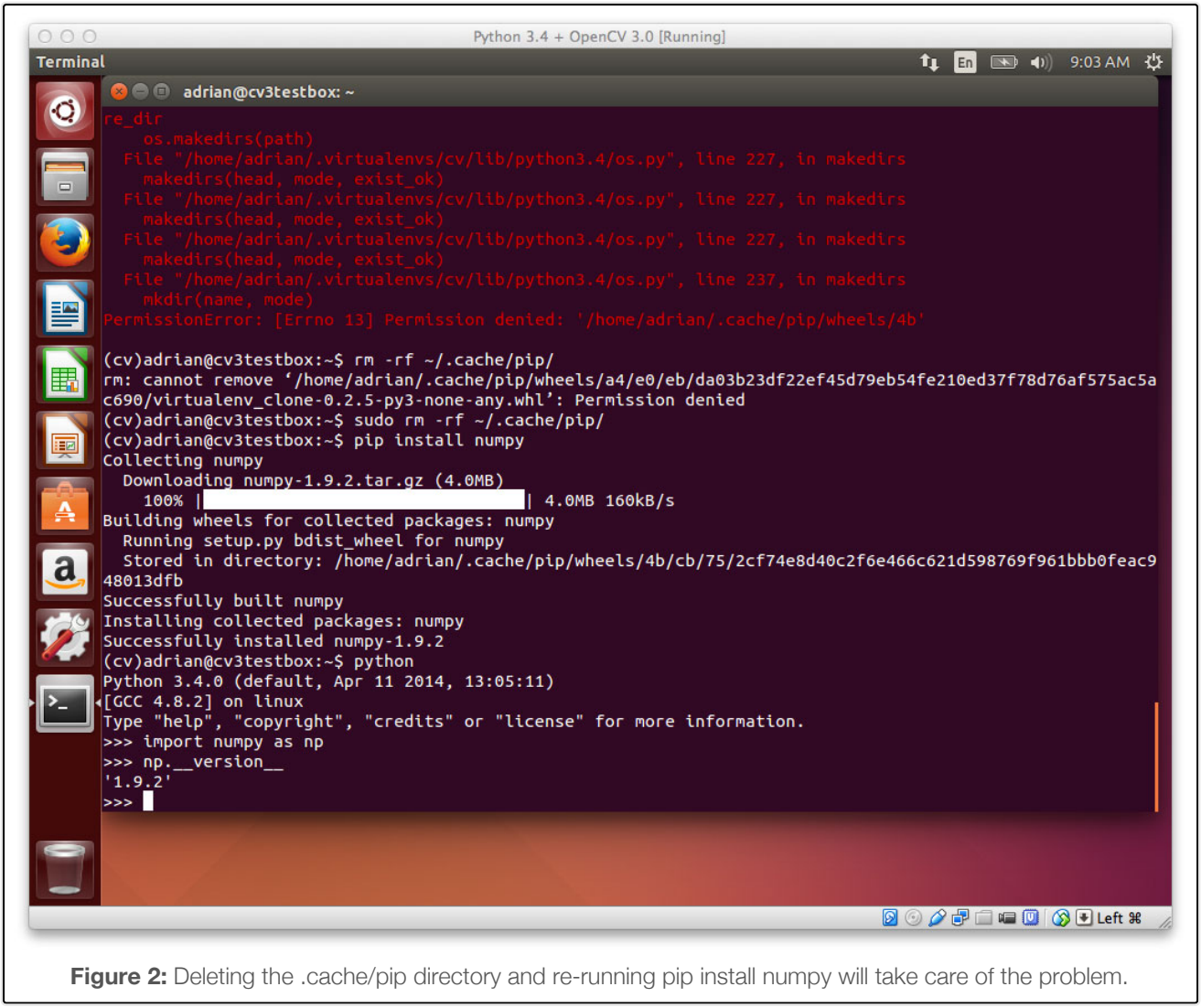


Figure 2: Deleting the .cache/pip directory and re-running pip install numpy will take care of the problem.

Step 3: Build and install OpenCV 3.0 with Python 3.4+ bindings

Alright, our system is all setup now! Let's pull down OpenCV from GitHub and checkout the `3.0.0` version:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ cd ~
2 $ git clone https://github.com/Itseez/opencv.git
3 $ cd
4 $ git checkout 3.0.0
```

We'll also need to grab the opencv_contrib repo as well (for more information as to why we need `opencv_contrib`, take a look at my previous OpenCV 3.0 Ubuntu install post):

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ cd ~
2 $ git clone https://github.com/Itseez/opencv_contrib.git
3 $ cd opencv_
4 $ git checkout 3.0.0
```

Time to setup the build:

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

Let's take a second to look at my CMake output:

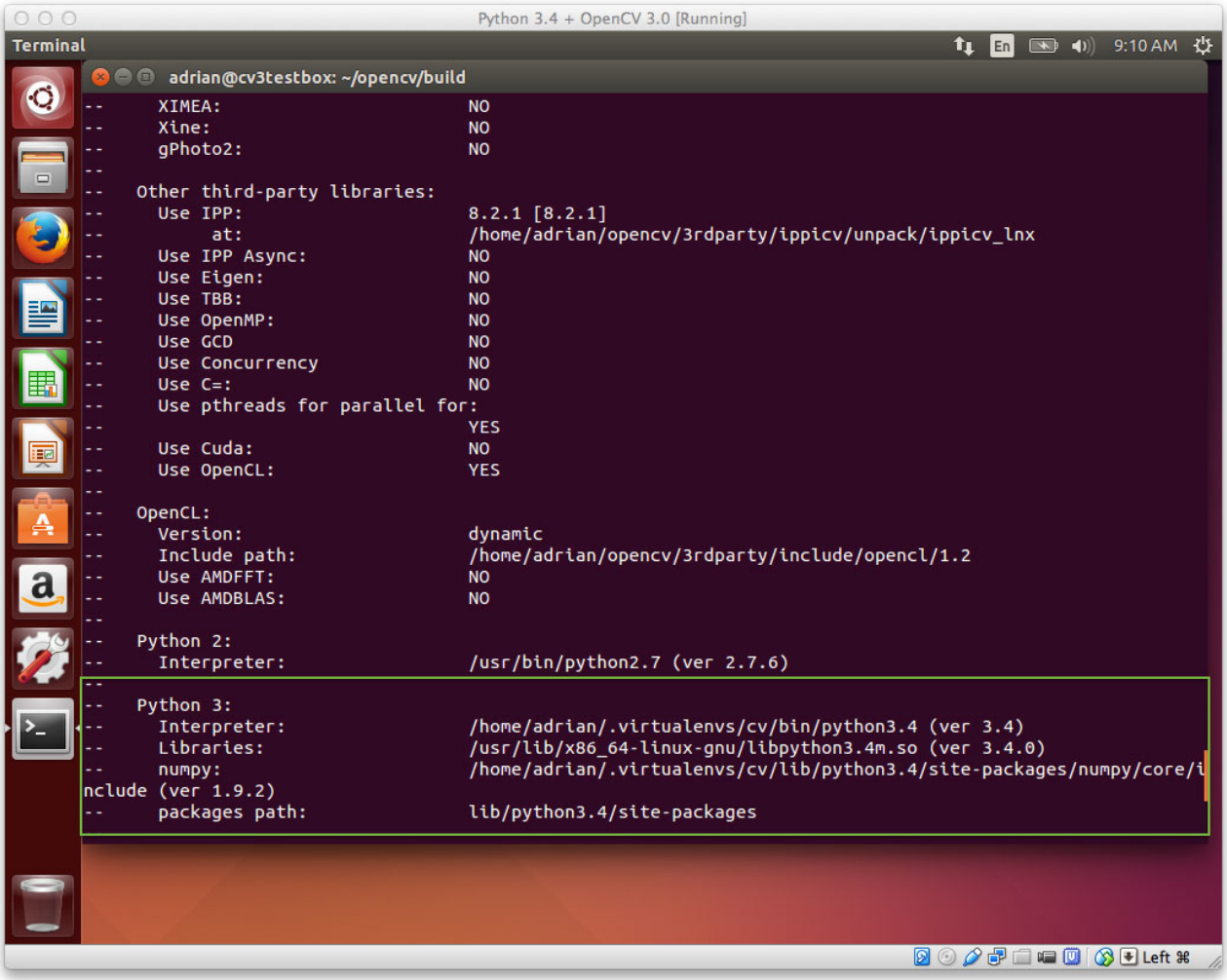


Figure 3: It's a good idea to inspect the output of CMake to ensure the proper Python 3 interpreter, libraries, etc. have been picked up.

Notice how CMake has been able to pick up our Python 3 interpreter! This indicates that OpenCV 3.0 will be compiled with our Python 3.4+ bindings.

Speaking of compiling, let's go ahead and kickoff the OpenCV compile process:

Install OpenCV 3.0 and Python 3.4+ on Ubuntu — Shell

```
$ make -j4
```

Where the *4* can be replaced with the number of available cores on your processor to speedup the compilation time.

Assuming OpenCV 3.0 compiled without error, you can now install it on your system:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ sudo make
2 $ sudo ldconfig
```

Step 4: Sym-link OpenCV 3.0

If you've reached this step, OpenCV 3.0 should now be installed in `/usr/local/lib/python3.4/site-packages/`

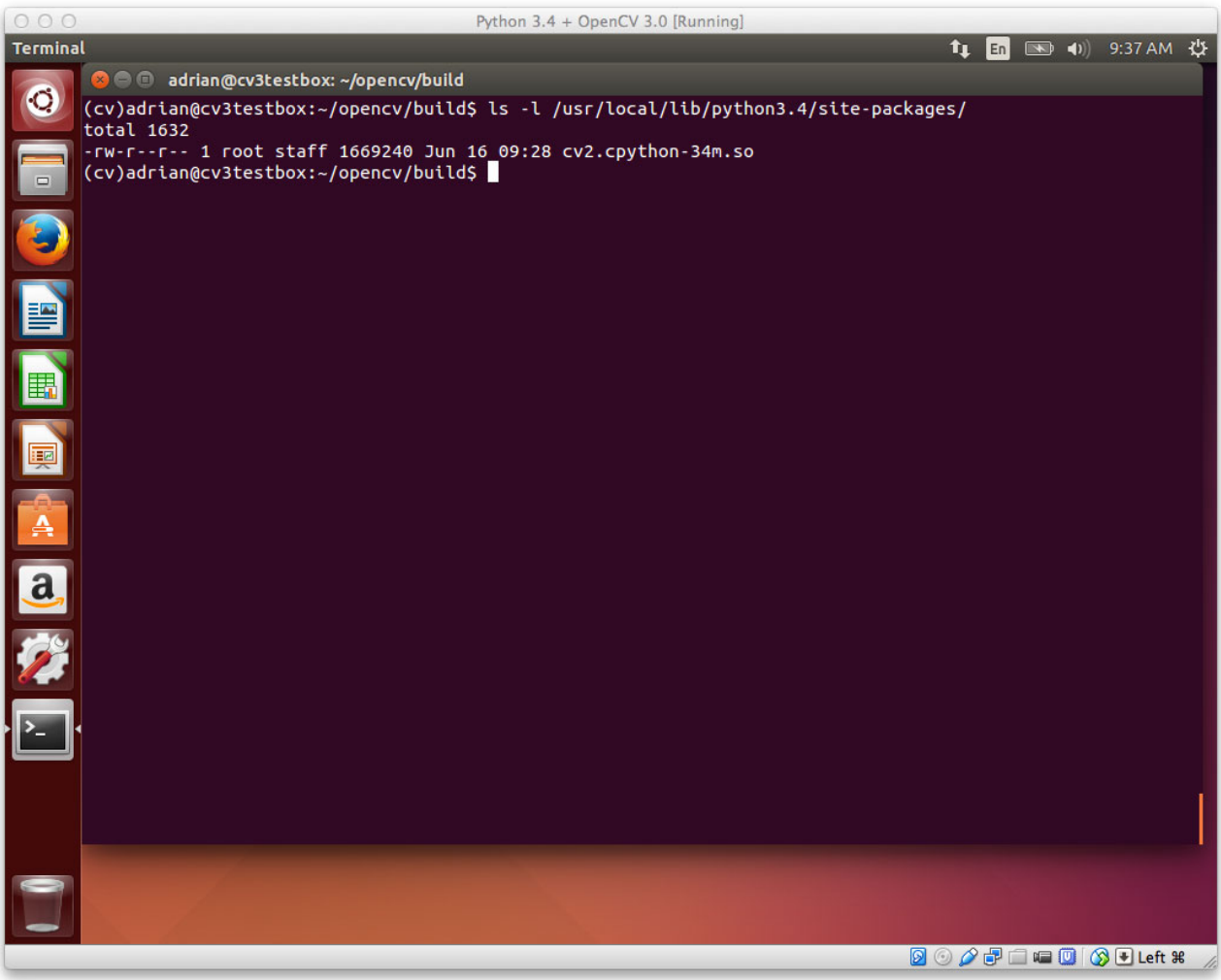


Figure 4: The Python 3.4+ OpenCV 3.0 bindings are now installed in /usr/local/lib/python3.4/site-packages/

Here, our OpenCV bindings are stored under the name `cv2.cpython-34m.so`

Be sure to take note of this filename, you'll need it in just a few seconds!

However, in order to use OpenCV 3.0 within our `cv` virtual environment, we first need to sym-link OpenCV into the `site-packages` directory of the `cv` environment, like this:

```
Install OpenCV 3.0 and Python 3.4+ on Ubuntu                    Shell
1 $ cd ~/.virtualenvs/cv/lib/python3.4/site-packages/
2 $ ln -s /usr/local/lib/python3.4/site-packages/cv2.cpython-34m.so cv2.so
```

Notice how I am changing the name from `cv2.cpython-34m.so` to `cv2.so` — this is so Python can import our OpenCV bindings using the name `cv2` .

So now when you list the contents of the `cv` virtual environment's `site-packages` directory, you'll see our OpenCV 3.0 bindings (the `cv2.so` file):

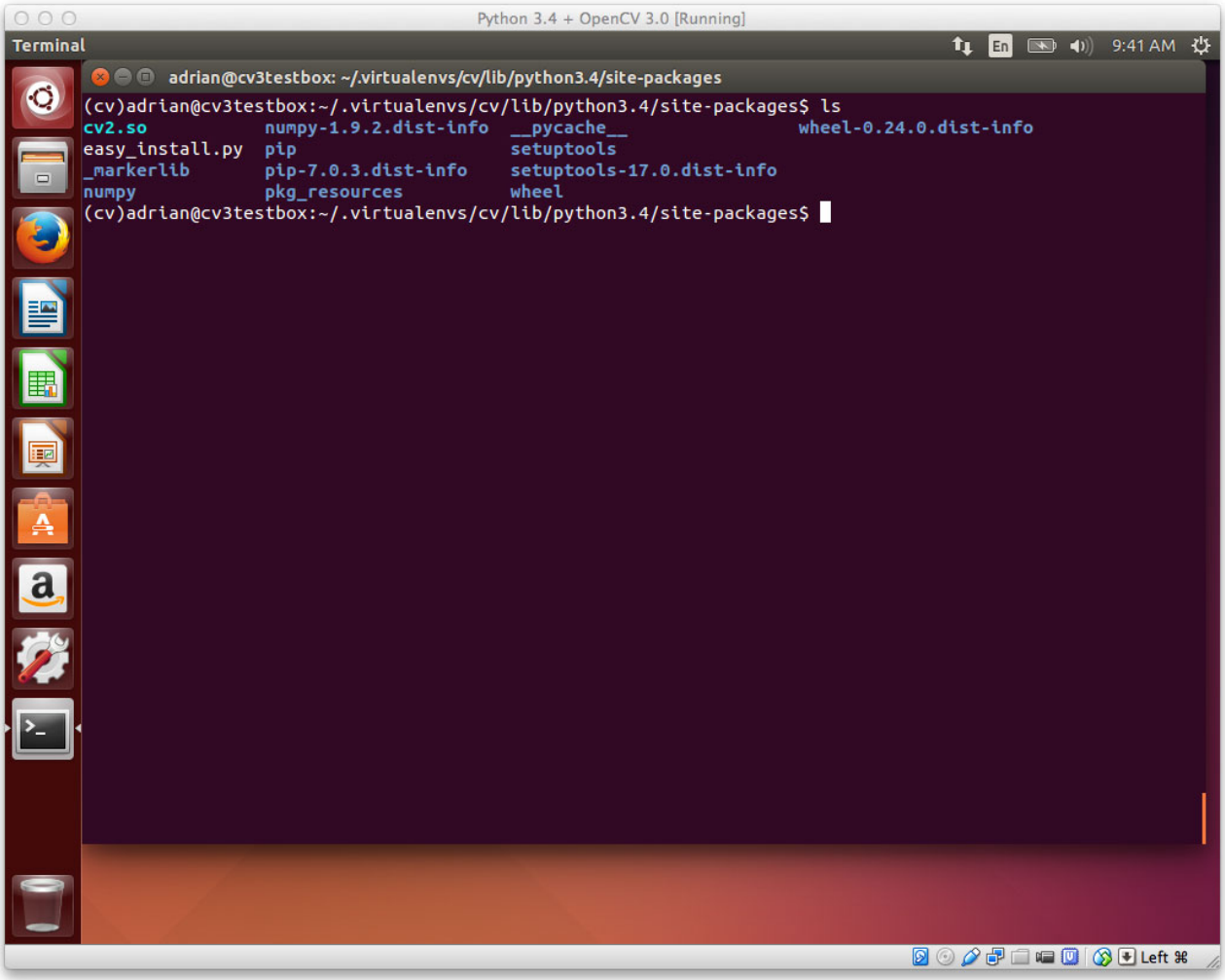


Figure 5: In order to access the OpenCV 3.0 bindings from our Python 3.4+ interpreter, we need to sym-link the cv2.so file into our site-packages directory.

Again, this is a very important step, so be sure that you have the `cv2.so` file in your virtual environment, otherwise you will not be able to import OpenCV in your Python scripts!

Step 5: Test out the OpenCV 3.0 and Python 3.4+ install

Nice work! You have successfully installed OpenCV 3.0 with Python 3.4+ bindings (and virtual environment support) on your Ubuntu system!

But before we break out the champagne and beers, let's confirm the installation has worked. First, ensure you are in the `cv` virtual environment, then fire up Python 3 and try to import `cv2`:

Install OpenCV 3.0 and Python 3.4+ on Ubuntu — Shell

```
$ workon
$ python
>>> import cv2
```

```
>>> cv2.__version__
'3.0.0'
```

Here's an example of me importing OpenCV 3.0 using Python 3.4+ on my own Ubuntu system:

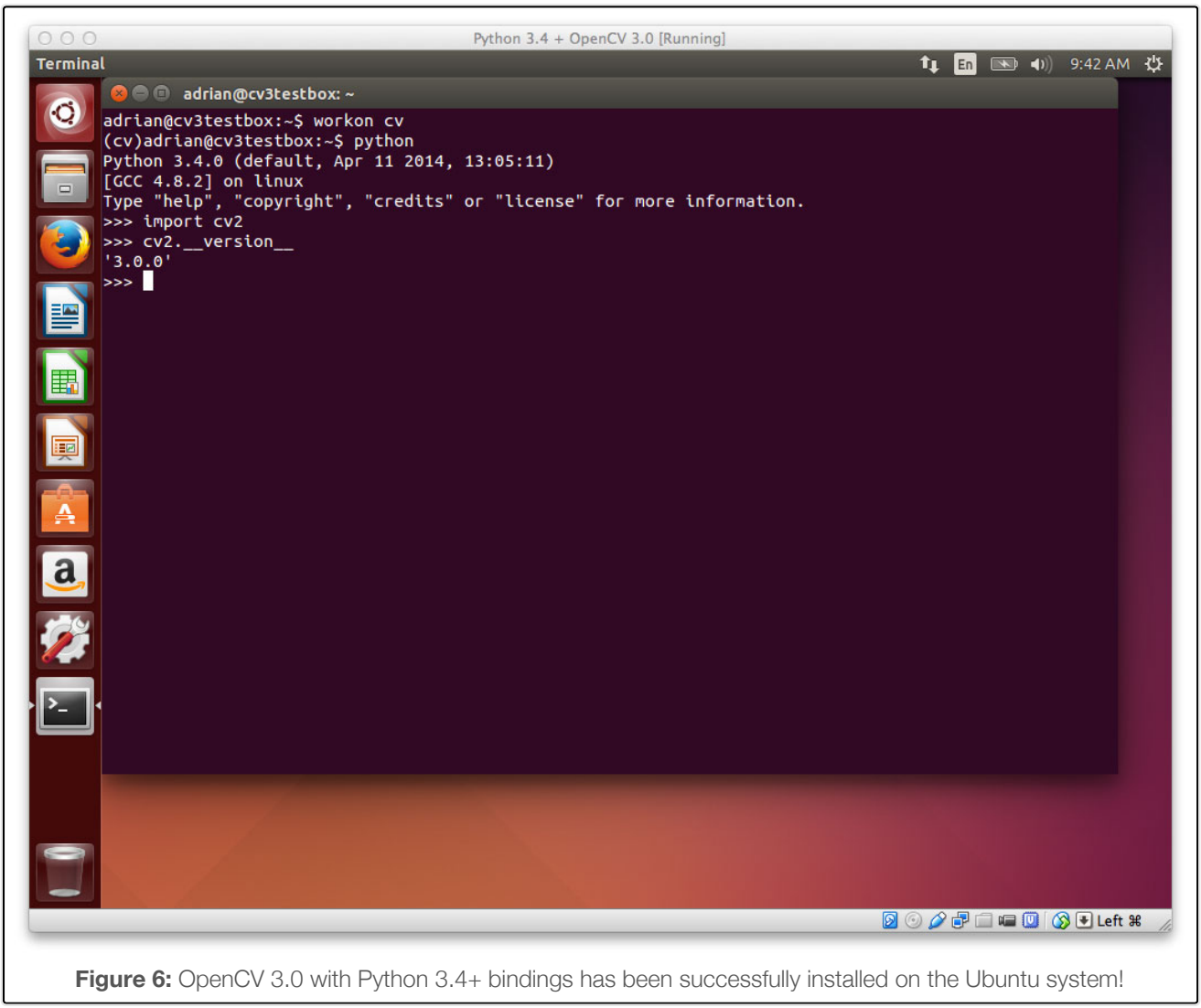


Figure 6: OpenCV 3.0 with Python 3.4+ bindings has been successfully installed on the Ubuntu system!

As you can see, OpenCV 3.0 with Python 3.4+ bindings has been successfully installed on my Ubuntu system!

Summary

In this tutorial I have demonstrated how to install OpenCV 3.0 with Python 3.4+ bindings on your Ubuntu system. This article is very similar to our previous tutorial on installing OpenCV 3.0 and Python 2.7 on Ubuntu, but takes advantage of OpenCV 3.0's new Python 3+ support, ensuring that we can use the Python 3 interpreter in our work.

While having Python 3.4+ support is really awesome and is certainly the future of the Python programming language, I would also advise you to take *special care* when considering migrating from Python 2.7 to Python 3.4. For many scientific developers, the move from Python 2.7 to 3.4 has been a slow, arduous one. While the big Python packages such as NumPy, SciPy, and scikit-learn have made the switch, there are still other smaller libraries that are dependent on Python 2.7. That said, if you're a scientific developer working in computer vision, machine

learning, or data science, you'll want to be careful when moving to Python 3.4 as you could easily pigeonhole your research.

Over the coming weeks the OpenCV 3.0 install-fest will continue, so if you would like to receive email updates when new install tutorials are released (such as installing OpenCV 3.0 with Homebrew, installing OpenCV 3.0 on the Raspberry Pi, and more), please enter your email address in the form below.

install, **opencv 3**, **python 3.4**, **tutorials**

‹ Where did SIFT and SU

3+ on your Raspberry Pi 2 ›

52 Responses t *on Ubuntu*

Sébastie

REPLY

Hi Adrien,

Thanks a lot! I follov nCV3 with Python3.4 on my Ubuntu VM.

I have a related que "export VIRTUALEN so python3.4 will be

Is it still possible to

Adrian Rosebrock July 21, 2015 at 6:58 pm #

REPLY

Hey Sébastien, you can still create virtual environments using Python 2.7. Using a command like this should work:

```
$ mkvirtualenv foo --python python2.7
```

Sébastien Vincent July 27, 2015 at 3:14 pm #

REPLY

Thanks!
Last question: is it safe to delete the opencv/build directory after install? Or must we keep it forever. Its size is 2.9Gb…

Adrian Rosebrock July 28, 2015 at 6:38 am #

REPLY

As long as you have ran `sudo make install`, you can safely delete the `build` directory.

Sébastien Vincent July 28, 2015 at 2:10 pm #

ok, thanks.

Ferdi Güler July 26, 2015 at 3:41 pm #

REPLY

Hi Adrian,
Such a good tutorial, would you mind explaining the installation steps if I choose to use standard virtual environment coming with Python3.4 (Python3 -m venv foo) instead of virtualenv?

Thanks

Adrian Rosebrock July 26, 2015 at 4:48 pm #

REPLY

Hey Ferdi, I'm actually unfamiliar with the virtual environment that comes with Python 3.4 (I'm just migrating to Python 3 myself — previously all I could use was Python 2.7). Do you have a link where I can read more about it?

Ferdi Güler July 27, 2015 at 4:41 am #

REPLY

Hi Again, thanks for your reply. More information about Python's native virtual environment can be found in the following link. It would be nice if can use it since it comes with Python 3.4 by default and there is no need to install any other 3rd party tools

https://docs.python.org/3/library/venv.html

Adrian Rosebrock July 27, 2015 at 6:05 am #

REPLY

Awesome, thanks for passing along the link I'll be sure to read up on it.

chetan July 31, 2015 at 2:59 am #

REPLY

Sir i am getting error as shown below. I have installed as per your instructions but still unable to install correctly. pls help me.

root@chetan-VirtualBox:~# workon cv
workon: command not found
root@chetan-VirtualBox:~# python
Python 2.7.6 (default, Jun 22 2015, 18:00:18)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
Traceback (most recent call last):
File "", line 1, in
ImportError: No module named cv2

Adrian Rosebrock July 31, 2015 at 7:06 am #

REPLY

Hey Chetan — if you are getting an error related to the `workon` command, then it's most likely that your `~/.bash_profile` file was not updated corrected or was not reloaded via the `source` command. Go back to the "Step 2: Setup Python (Part 1)" section and ensure you have updated your `~/.bash_profile` file to reflect the changes I suggested.

Meareg August 3, 2015 at 4:38 am #

REPLY

Hello adrian, very nice step-by-step tutorial. Thanks man!! 🙂
I had to spend some time before figuring out I had to comment out export 'PYTHONPATH=$PYTHONPATH:/usr/lib/local/python2.7/dist-packages' (if such line exists in your ~/.bashrc file) and replace it with

'PYTHONPATH=$PYTHONPATH:/usr/lib/local/python3.4/dist-packages' or simply replace 2.7 with 3.4
This will allow our openCV3.0 compilation to choose the python3.4 interpreter specifically and to include the opencv3.0's bindings for python3.4 as shown above in step 3's cmake output.

Adrian Rosebrock August 3, 2015 at 6:42 am #

REPLY

Nice catch!

Mike Ream August 20, 2015 at 9:41 pm #

REPLY

OK…here is what I did wrong….I had to exit in the middle of this process and when I came back into my terminal I was not on the cv environment when I did the cmake command. I tried running it again with no luck. I then removed the entire build directory and started again making sure I entered "workon cv" to be sure I was in the virtual environment. I followed the steps and all seems OK now.

Adrian Rosebrock August 21, 2015 at 7:17 am #

REPLY

Nice, I'm glad it's working for you Mike!

And yes, to all other readers: **if you leave your computer, restart it, open up a new terminal, etc. be sure to use the workon command to re-enter your virtual environment.**

Tony August 20, 2015 at 1:52 pm #

REPLY

thanks.

Mike Ream August 20, 2015 at 10:05 pm #

REPLY

Sorry for all the comments. I also get the following output when I type cv2.__version__: '3.0.0-dev' . Are other people getting the dev version? I was surprised to get a different output.

Adrian Rosebrock August 21, 2015 at 7:18 am #

REPLY

After pulling down the repository from GitHub did you run `git checkout 3.0.0`?

It seems like you might have forgotten the checkout command.

Mike Ream August 24, 2015 at 10:18 am #

REPLY

Your suspicions were correct. I could have sworn that I ran the “git checkout 3.0.0″command, but I must have forgot it. After I followed the steps again, being more careful I do get the correct version installed.

Adrian Rosebrock August 24, 2015 at 11:11 am #

REPLY

Fantastic, I’m glad it was just a git checkout issue 🙂

Tafadzwa August 25, 2015 at 4:56 am #

REPLY

i’ve encountered an error whilst running “sudo pip3 install virtualenv virtualenvwrapper”…
The response,”Cannot fetch base index……”

Adrian Rosebrock August 25, 2015 at 6:36 am #

REPLY

That sounds like an an issue with pip or your machine.

Tafadzwa September 3, 2015 at 9:46 am #

REPLY

Thanks a lot…..After running the command “cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D INSTALL_C_EXAMPLES=ON \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
-D BUILD_EXAMPLES=ON ..” i have encountered this error,”CMake Error at 3rdparty/ippicv/downloader.cmake:75 (message):
ICV: Failed to download ICV package: ippicv_linux_20141027.tgz. Status=22;”HTTP response code said error”

How can that be resolved??

Adrian Rosebrock September 3, 2015 at 12:21 pm #

REPLY

That sounds like an error related to your internet connection and not OpenCV. The ICV package needs to be downloaded and for whatever reason, the the download is failing — likely due to a connectivity issue (either on your end or OpenCV's).

Nitin September 10, 2015 at 11:10 am #

REPLY

If anyone found dist-package instead of site-package, then use these commands and i guess it will work…

```
1 $ ls -l  /usr/local/lib/python3.4/dist-packages/cv2.cpython-34m.so
2 $ ln -s /usr/local/lib/python3.4/dist-packages/cv2.cpython-34m.so cv2.so
3 $ workon cv
```

Adrian Rosebrock September 11, 2015 at 6:34 am #

REPLY

Thanks for sharing Nitin!

Shiva September 30, 2015 at 7:59 am #

REPLY

Oops, looks like I'd forgotten to update the .bashrc file! It works fine now.

My other query still stands!

Jonathan October 7, 2015 at 3:47 am #

REPLY

Hello,

I've run through the compile and make processes with no warnings or errors. When I run sudo ldconfig from the build directory, there is no output. And when I try to look for the opencv *.so files, I cannot find them (neither in /usr/local/lib/python3.4/dist-packages or site-packages directories. Do you know what might be causing the problem ? What information can I provide to help you understand the issue ? Thanks in advance – and for the great tutorial !

Adrian Rosebrock October 7, 2015 at 6:28 am #

REPLY

Hey Jonathan — be sure to check for `.so` files in the `build/lib` directory. This has happened to me a few times before and then I just manually moved it to the `site-`

`packages` directory.

Milos October 21, 2015 at 11:11 am #

REPLY

Nice, works out of the box. Although, I installed virtual environment with just “virtualenv env –python python3.4”.

Adrian Rosebrock October 22, 2015 at 6:20 am #

REPLY

Awesome, I’m glad the tutorial worked for you Milos! 🙂

Almazi October 27, 2015 at 3:47 pm #

REPLY

Thanks for this great tutorial which worked pretty good. But everytime I start terminal the “workon command: not found” then I have to go thru exporting the virtualenv and virtualenvwrapper. Why is it happening and is it normal? Is there any way to make a virtual terminal which will work for cv directly?
Thanks

Adrian Rosebrock November 3, 2015 at 10:43 am #

REPLY

Hey Almazi — just to clarify: only the `workon` command is not found? The `mkvirtualenv` and `rmvirtualenv` commands work?

Andrea November 12, 2015 at 10:10 am #

REPLY

Hi Adrian, thanks a lot for your tutorial.
I’m having the same problem as Almazi, and I get also the same message when executing mkvirtualenv and rmvirtualenv, how can a deal with it?
Thanks in advance

Adrian Rosebrock November 12, 2015 at 12:09 pm #

REPLY

Hey Andrea — you might want to take a look at the “Troubleshooting” section of this post. While the instructions are for the Raspberry Pi, you can apply the same logic to fix the virtualenv commands.

peemji November 1, 2015 at 1:36 pm #

REPLY

I'm working on a SIFT project that need matplotlib as prerequisite but I can't build(หรือ compile) it. Could you give me any example or instruction?

Adrian Rosebrock November 3, 2015 at 10:19 am #

REPLY

Please see my reply to Sommai above.

Sommai Dev November 2, 2015 at 4:27 am #

REPLY

I'm working on a SIFT project that need matplotlib as prerequisite but I can't build it to virualane . Could you give me any example or instruction?

Adrian Rosebrock November 3, 2015 at 10:17 am #

REPLY

Please see this tutorial on how to install matplotlib into the `cv` virtual environment.

Evan November 5, 2015 at 11:57 pm #

REPLY

The best, most straightforward installation procedure I've ever read. THANKS!!!

Adrian Rosebrock November 6, 2015 at 6:19 am #

REPLY

Thanks so much Evan, I'm glad it worked for you! 🙂

Baptiste November 8, 2015 at 12:16 pm #

REPLY

Works great in Terminal, but i can't import cv2 in idle or spyder ? how do configure them to work in your virtual cv ? Thanks !

Adrian Rosebrock November 8, 2015 at 3:19 pm #

REPLY

I'm not familiar with the Sypder IDE, but if it's anything like PyCharm, you just need to set the proper virtual environment for the project. Please see this post for more information.

Baptiste November 10, 2015 at 1:38 am #

REPLY

It works !!! with Eclipse or Pycharm … I don't understand why not with my spyder ? I Have yet some trouble to solve with matplotlib… (I will try your patch of your blog) – Thank you very much !

waspinator November 18, 2015 at 10:51 am #

REPLY

have you heard of Conda/Anaconda? It makes managing your packages and environments easier. I think it would be a better fit for computer vision researchers and practitioners than pip/virtualenv which are designed more for python programmers.

https://www.continuum.io/downloads

Adrian Rosebrock November 18, 2015 at 6:54 pm #

REPLY

Yes, I've heard of and used conda before. They actually used to include OpenCV in their pre-compiled packages but it was removed awhile back. I also don't talk about it publicly, but the Continuum company and myself had a pretty big falling out (at least on my end of things). You won't see me use their products on the blog.

saborae December 9, 2015 at 9:22 pm #

REPLY

Hey, I did as you instructed and I get the similar output on the cmake but after the packages path: site-packages, there's another line that says python(for build): 2.7

Also, compiling with -j4, I get several warnings all .cpp or .hpp files. Anyway, the install works, with my camera, video files, all fine but now, how do I use opencv on my regular python install? should I redo all the steps without virtualenv?

Adrian Rosebrock December 10, 2015 at 6:52 am #

REPLY

You can ignore the "for build" section, that part of the CMake script is buggy. As for using OpenCV in your regular Python install, just the `cv2.so` file into your regular `site-packages` directory (if it's not already there). From there, you'll be able to import OpenCV

outside of the virtual environment.

Hacklavya December 17, 2015 at 1:42 pm #

REPLY

I want to keep both
install-opencv-3-0-and-python-3-4-on-ubuntu/
and
install-opencv-3-0-and-python-2-7-on-ubuntu/
will this tutorial keep older one intact ?

Actually I want all following on my system, so that I can use according to need:-
1. opencv 3.0 & python 3.4
2. opencv 3-0 & python 2.7
3. opencv 2.4 & python 2.7
How can I have all the three ?

Adrian Rosebrock December 18, 2015 at 5:56 am #

REPLY

It certainly is possible, but it's non-trivial. The trick is to compile and install OpenCV 3.0 (for either Python version first). But that is the only time you run `make install`. Afterwards, you create separate Python virtual environments for the remaining two installs. Then you compile your respective Python versions. But only keep the `build` directory after compiling. You can then sym-link in the `cv2.so` bindings from the respective `build/lib` directories into the correct Python virtual environments.

Like I said, it's not easy to do — but it's absolutely possible.

Jack December 7, 2015 at 8:02 pm #

REPLY

Hello,
I'm looking to connect my "cv" virtualenv that was created in this tutorial to PyCharm (as per one of your other tutorials) but I can't figure out where the "cv" virtualenv is stored. As per this guide, where do I look to find that virtual environment? (also what is it's final extension?)
Thanks a million,
Jack

Ahh, of course. I try to dig around as long as I can before posting stupid comments. I found the .virtualenvs dir inside my home directory then all I had to do was point the PyCharm interpreter at .virtualenvs/cv/bin/python3.4 (as reference for anyone else in my position).
Thanks for the guide!
-Jack

Adrian Rosebrock December 8, 2015 at 6:28 am #

REPLY

I'm glad you figured it out Jack. Also, for future reference, I have a tutorial dedicated to setting up PyCharm with virtual environments.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

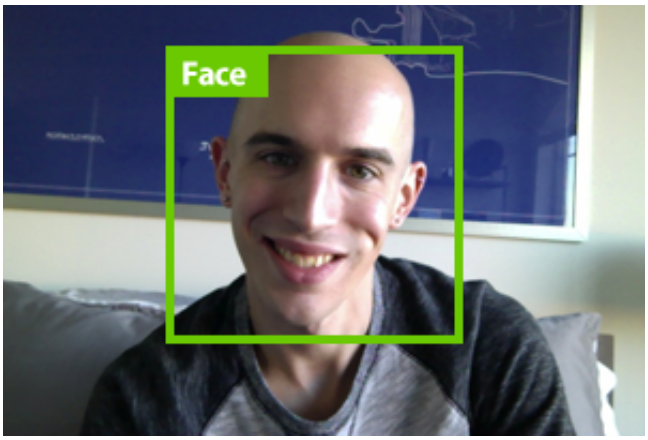
Resource Guide (it's totally free).

Enter your email address below to get my **free 11-page Image Search Engine Resource Guide PDF**. Uncover **exclusive techniques** that I don't publish on this blog and start building image search engines of your own!

Your email address

DOWNLOAD THE GUIDE!

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 new members every 2-4 weeks.

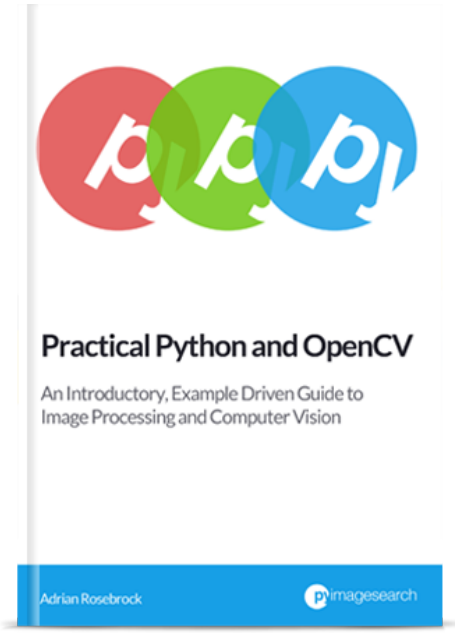
PyImageSearch Gurus enrolls new members every 2 to 4 weeks. Seats inside the course are limited, so I suggest you ***claim your spot in line*** when the doors open again and the next enrollment session starts.

CLICK HERE TO CLAIM YOUR SPOT!

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.



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 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

Accessing the Raspberry Pi Camera with OpenCV and Python
MARCH 30, 2015

Install OpenCV 3.0 and Python 2.7+ on Ubuntu
JUNE 22, 2015

Installing OpenCV 3.0 for both Python 2.7 and Python 3+ on your Raspberry Pi 2
JULY 27, 2015

Install OpenCV 3.0 and Python 2.7+ on OSX
JUNE 15, 2015

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

Search

Search...

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.