Anyone successfully install openCV 3.1 for Python2 and Python 3 on separate virtual environments?

psmedina **Aug 10**

I seem to be having problems with this... I can successfully install a version of openCV3 with Python 3.5.1 in one virtual environment, however if I try to install openCV3 with Python 2 in a different environment, I can no longer access openCV3 in the Python3 environment. I think it has something to do with the cv2.so file being corrupted when I install openCV3 with Python 2... I am working on a Mac OSX El Capitan.

Does anyone have thoughts regarding this? I know it can be done?

Thanks!

RayFerr **Aug 10**

Hi @psmedina

> psmedina:
>
> I can successfully install a version of openCV3 with Python 3.5.1 in one virtual environmen

At first, I can see that you already have a virtual environment named such as "gurus1". If you have made every step ok on this, then you have installed OpenCV3 after you created "gurus1". I mean that you have installed OpenCV3 inside "GURUS1".

> psmedina:
>
> if I try to install openCV3 with Python 2 in a different environment

Now, if you want to install again OpenCV3, I think you have created another "virtual environment" named such as "gurus2", but "note" before you want to created a new virtual environment, we must go-out (exit) "gurus1", and after that inside this new virtual environment we should install OpenCV3
Then, we have now two "virtual environment".

> psmedina:
>
> I can no longer access openCV3 in the Python3 environment. I think it has something to do with the cv2.so file being corrupted when I install openCV3 with Python 2..

As I said, if you have created two virtual environment such as "gurus1" and "gurus2", and you have done everythig good, this error can't happen, because, you first create "gurus1" and inside there you downloaded OpenCV3.zip + extract that, and install that inside "gurus1". Then you should go-out (exit) gurus1, and create "gurus2", and again, inside there you should donwload OpenCV3.zip + extract that + install.

If you have done that I said above, everything should work well.

> psmedina:
>
> Does anyone have thoughts regarding this? I know it can be done?

Of course you can make this configuration. I see you have set OpenCV3 + Python3. Congratulations. I have never done that, 😊 I was trying but I couldn't.
You are making nice job. Please go-back setting and be sure for donwloading OpenCV3.zip twice, one inside "gurus1" and another one for "gurus2", and again extract+install in each case.

I think you know already all I said, but I hope this remember could help you.

Have a nice day.

Regards

Ray Ferr

---

psmedina **Aug 10**

[**@RayFerr**] Thanks Ray, I will try this out. I have a feeling it's just the order of the steps that I am doing it in... something minor really... I will try it out again later.

With regards to your issue of OpenCV3 and Python3, are you installing Python3 with vtk version 7? This was a major problem for me and it doesn't appear to have been resolved by the opencv peeps. Once I corrected the vtk problem, the installation was flawless (with running vtk 7). Let me know if it is, and I will shoot you my fix.

---

hilman **Aug 10**

I have installed OpenCV 3.1 + Python 2 with VTK 7. But can't seems to get the OpenCV viz module worked correctly on Mac. It is good in Ubuntu though.

RayFerr **Aug 10**

Hi **@psmedina**

> psmedina:
>
> Let me know if it is, and I will shoot you my fix.

Thanks a lot. Be sure I'm gonna tell you for this later. 🙂
For now I use my computer with Linux (OS) and it has Python2.7 + OpenCV2.4.13. This computer is only to use for "Gurus Course". I'm a beginner student and I'm happy for learning a lot in this course that **@Adrian** has written.
As I said, I don't use virtual environment.

> psmedina:
>
> I have a feeling it's just the order of the steps that I am doing it in...

Sure, sometimes a little thing makes us a big problem. 😉

Please, when I get more level on OpenCV, I'd like to ask you for showing me how to set Python3.x + OpenCV3.x

Have a nice day

Enjoy ☀️

Cheers

Ray Ferr

---

godreau **Aug 17**

@psmedina - I've got a parallel install working; CV3.1.0 + python2.7.12 / 3.5.2, ubuntu 16.04.1 LTS

The trick is that the cv2.so symbolic link:

> ~/.//lib/python<2/3>/site-packages/cv2.so

Needs to reference the same library

> /usr/local/lib/python<2/3>/site-packages/cv2.so

regardless of the version of python being used.

When you run make install, it'll pick whatever /usr/local/lib/pythonX is specified in your path. From then on, any virtualenv you create must reference that same .so file. That means python3 will be referencing the site-packages libraries from python2 (if you make/installed to python2; or vice-versa).

Then, creation of new envs is as simple as:

(For python 2.7)

> $ mkvirtualenv -p /usr/bin/python2.7 cv27
> $ ln -s /usr/local/lib/python2.7/site-packages/cv2.so ~/.virtualenvs/cv27/lib/python2.7/site-packages/cv2.so

And conversely, for python 3.5 (note that I reference the *same* source cv2.so)

> $ mkvirtualenv -p /usr/bin/python3.5 cv35
> $ ln -s /usr/local/lib/python2.7/site-packages/cv2.so ~/.virtualenvs/cv27/lib/python3.5/site-packages/cv2.so

I believe you can also manually switch around symlinks for parallel installed versions of OpenCV as well but I haven't tried it yet.

HTH

---

hilman **Aug 18**

Means that if I have already installed a OpenCV with Python 2, I can use the exact same cv2.so with Python 3? Just need to `ln -s` it?

Can this somehow cause a weird problems (since the OpenCV is originally compiled with Python 2 support)?

---

godreau **Aug 18**

Time will tell, I'm navigating those waters now...

---

psmedina **Aug 18**

@godreau , @hillman: So it seems to me that there are two options at present:
1) **@RayFerr** : Download and install opencv 3 on two separate virtual environments
2) **@godreau** : Install opencv 3 in the main system and then link each virtual environment to the resulting cv2.so file.

I think option (1) is guaranteed to work, while it appears option (2) is still being tested out?

I will probably stick with option 1 for now, and if **@godreau** can keep us informed of any problems with option 2 then I will probably change my system to that in time.

Thanks!

---

godreau **Aug 18**

@psmedina - I'm not 100% sure, but I believe that, even if you make/install the opencv and opencv-contrib libraries when you're in an activated env; it'll still install in your system path /usr/local/lib and not in your env. I'm not a super expert with environment management just yet, but it seems the make/install process circumvents pip. Virtualenv is 'hooked into' pip, such that it can isolate changes it (pip) makes, but the make/install process completes without 'notifying' virtualenv, and thus ends up in the system paths. Perhaps somebody a little smarter than myself can confirm this, but that's the behavior I seem to be demonstrating.

RayFerr **Aug 18**

Hey Patrick **@psmedina**

How you doing?
Hope you're ok.

Thanks a lot for your helping.

> psmedina:

> 1) @RayFerr : Download and install opencv 3 on two separate virtual environments

I use Linux Ubuntu 14, and I don't have a virtual machine or virtual environment. As I said, I would ask you for setting OpenCv3.x + Python3.x., be sure about it. 😄

You're so right, the best way is to set a virtual environment for each version, either python or opencv. Your advice is the best way. Such as "gurus1" for OpenCv2.4.x + Python2.7, and "gurus2" for OpenCv3.x+Python3.x , where gurus1 and gurus2 are virtual environments.
Using this, as you said we won't have problems.

Thanks.

Have a nice day.

Adrian Chief PyImageSearcher **Aug 20**

@godreau 's solution to sym-link in your `cv2.so` file into different Python virtual environments is exactly what I do.

However, there is a subtle problem with this that you might run into, especially if you're unfamiliar with compiling libraries against against existing packages. Let's say you compiled OpenCV with Python 2.7 support. You'll be able to import `cv2` into your Python 2.7 environment just fine. However, if you were to sym-link the `cv2.so` file into your Python 3 environment, you would get a seg-fault error.

The reason?

OpenCV was compiled against Python 2.7 instead of Python 3. The solution is to therefore compile multiple versions of OpenCV.

Inside my `/home/adrian` directory, I have a sub-directory named `opencv`. In this directory I keep the `build` directories for all of my OpenCV compiles. One for Python 2.7 + OpenCV 2.4, another for Python 2.7 + OpenCV 3, and another for Python 3 + OpenCV 3. I also have separate virtual environments for each of them with the correct `cv2.so` file sym-linked in.

---

psmedina **Aug 23**

Thanks @Adrian , that seemed to work. I need to try an additional build of OpenCV 3 + Python 2.7, but I at least have OpenCV 2 + Python 2.7 and OpenCV 3 + Python 3.5 working for now. However, Ph.D. stuff is calling so I will post an update of my success with regards to OpenCV3 + Python 2.7 when I get a chance to sit down and work on it.

---

godreau **Aug 23**

Thanks @Adrian and @psmedina for the confirmation - I've *finally* found an elegant solution that's somewhat of a hybrid of bits of info I've dug up over the past week. The most important thing is that it's a local instead of system-wide install, so you don't have to worry about mucking up your file system and leaving dangling files. Here's the workflow outline:

- Clone one copy of the repo of opencv to your drive (i.e. ~/.virtualenvs/opencvrepo)
- When building (cmake/make), you *must have the virtualenv you want to build for activated*. The cmake process references the python interpreter which is active when you build it (i.e. ~/.virtualenvs//local/bin/python2.7) - this threw me for a loop. Note that you also have to change the target build directories, and you also *do not* 'make install'; you manually copy the libraries (see steps below). Here are the cmake flags I used (hat tip @caioiglesias , I stole / modified your idea)

> cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=$VIRTUAL_ENV/local/ -D INSTALL_C_EXAMPLES=OFF -D INSTALL_PYTHON_EXAMPLES=ON -D OPENCV_EXTRA_MODULES_PATH=../opencv_contrib/modules -D PYTHON_EXECUTABLE=$VIRTUAL_ENV/bin/python -D PYTHON_PACKAGES_PATH=$VIRTUAL_ENV/lib/python2.7/site-packages -D BUILD_EXAMPLES=ON ..

- After the compile process is complete, copy the resultant lib directory to your local library 'repo' (i.e. ~/.virtualenvs/opencv_python2.7_opencv3.1.0) - note that you will have to keep track of the book keeping of what python and opencv version everything's compiled for; hence my naming convention
- Symlink your virtualenv's site-packages library to your local repo, i.e.:

> ln -s /home/user/.virtualenvs/opencv_python2.7_opencv3.1.0/lib/cv2.so /home/user/.virtualenvs/py2.7_ocv3.1.0/lib/python2.7/site-packages/cv2.so

And that should take care of it for that particular permutation of python/CV versioning. Note that, although you *should* be able to symlink to the same library in the future with the same python/CV combination, you *may* [*needs citation*] run into issues as the build references an interpreter that's explicitly in the virtualenv *when you built the package*.

With that said; to re-use the same git clone of openCV to make builds for *other* virtualenvs and versions, you *must* delete 'CMakeCache.txt' (in build folder) AND be in the active virtualenv of the target python interpreter *before* you build the package. After the build is complete, copying the libraries is the same as the process above.

Adrian Chief PyImageSearcher **Aug 23**

> godreau:
>
> Note that you also have to change the target build directories, and you also do not 'make install';

The not needing `make install` is a critical tip and one that I think new Unix users don't understand. By understanding how paths work in Unix, you can avoid these types of confusing errors -- however, I don't know of an *easy* way to teach this.

---

godreau **Aug 23**

@Adrian - agreed; it was a solid 10-15h of #struggle. L33t folks who build from source generally have been doing this for years and may not 'pave the way' with documentation for dummies like myself to learn.

I think it'd be worth the time to write a bash script for ubuntu for this. The script could help explain the process of steps with comments and could be forked/adapted to other distros. I could spearhead this if I get time.

Side note - I love how you've kept the option for a self hosted install in the course. Pls don't get rid of it!

Adrian Chief PyImageSearcher **Aug 24**

> godreau:
>
> Side note - I love how you've kept the option for a self hosted install in the course. Pls don't get rid of it!

No worries, that won't be going anywhere 🙂

In the past I've tried to create "auto install" scripts for Ubuntu, but ultimately they ended up failing for one reason or another. I would then have to track down *why* it failed instead of just investigating the output of CMake like I normally would. In the end, it took longer to debug the scripts than just actually compile and install OpenCV by hand.

---

hilman **Sep 2**

If you managed to do it, that would be awesome!

But I guess it is better to just teach the people here how to install it by hand. It can give us a really good grasp on how actually Mac and Ubuntu work behind the scene.

---

hilman **Sep 7**

May I ask a few questions as I want to install the OpenCV 3 for my Python 3 (already installed a system-wide OpenCV 3 for my Python 2)

1. Since we will not `make install`, isn't the `CMAKE_INSTALL_PREFIX` flag will become kind of unimportant? Is it just as a safe precaution in setting it to a local file?
2. I am thinking of just using the same git clone I've used to install the OpenCV 3 for my Python 2. Why do I need to delete the `CMakeCache.txt` file in the build folder? Doesn't the build folder itself is dependent on

its own and will not interfere with the OpenCV git clone?

Adrian Chief PyImageSearcher **Sep 8**

Once you run `cmake` from within the `build` directory, your `build` directory becomes populated with necessary configurations to compile your project. The `CMakeCache.txt` file simply caches these configurations so if you modify parts of the build, the entire `cmake` command does not have to be executed again (although I've found this to be very buggy in certain circumstances). This makes updating changes to your build (before running `make`) easier.

If you've *already* compiled OpenCV using a given `build` directory, you can delete the `build` directory and re-create it, or blow away the cache.