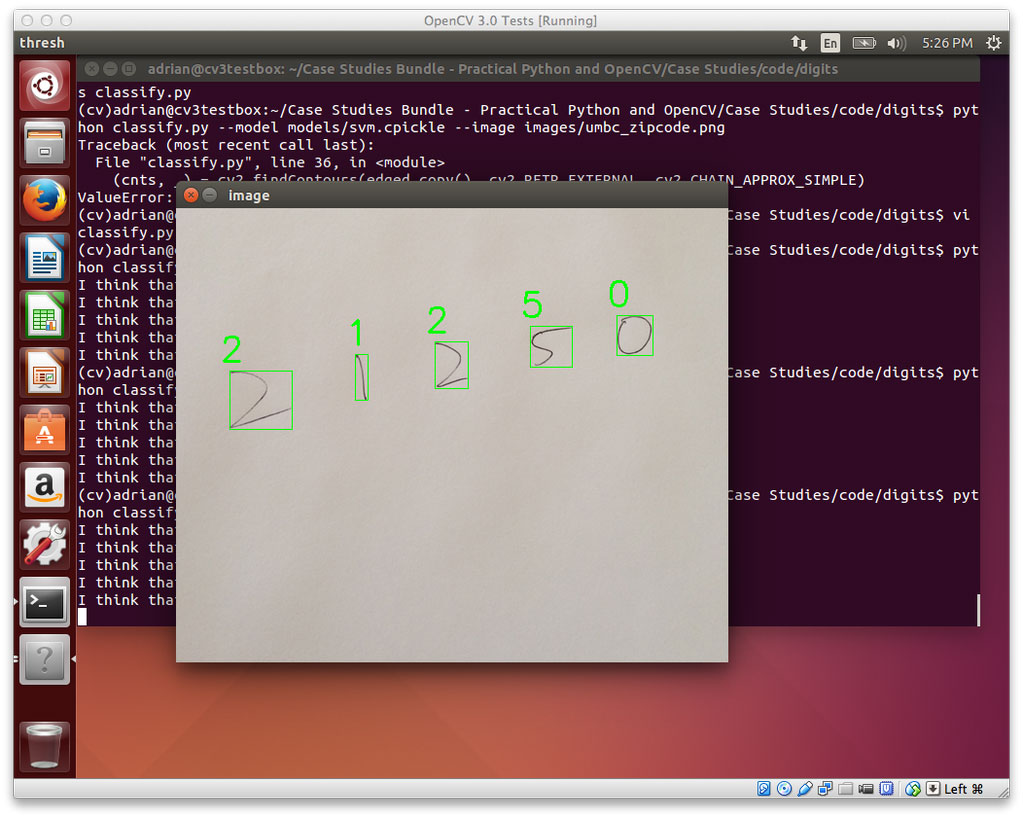
**Install OpenCV 3.0 and Python 2.7+ on Ubuntu**

by [**Adrian Rosebrock**](http://www.pyimagesearch.com/author/adrian/) on June 22, 2015 in [**OpenCV 3**](http://www.pyimagesearch.com/category/opencv-3/), [**Tutorials**](http://www.pyimagesearch.com/category/tutorials/)

[0](https://twitter.com/share?original_referer=http%3A%2F%2Fwww.pyimagesearch.com%2F&source=tweetbutton&text=Install+OpenCV+3.0+and+Python+2.7%2B+on+Ubuntu&url=http%3A%2F%2Fwww.pyimagesearch.com%2F2015%2F06%2F22%2Finstall-opencv-3-0-and-python-2-7-on-ubuntu%2F&via=pyimagesearch)

[12](https://www.linkedin.com/cws/share?url=http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/)

[](https://www.pyimagesearch.com/practical-python-opencv/?src=opencv3-install)

Last week we kicked-off the OpenCV 3.0 install fest by detailing [how to install OpenCV 3.0 and Python 2.7+ on the OSX platform](http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/).

Today we are going to continue the OpenCV 3.0 install instruction series by moving over to the Ubuntu operating system.

In the remainder of the post I will provide instructions on how to configure and install OpenCV 3.0 and Python 2.7+ on Ubuntu. I have *personally tested* these instructions on Ubuntu 14.04, but they should pretty much work on any Debian-based operating system.

**A quick note before we get started:** Yes, OpenCV 3.0 is indeed compatible with Python 3+. However, the install instructions are slightly different between Python 2.7+ and Python 3+. In an effort to keep each article self-contained and easy to follow, I am creating separate OpenCV 3.0 install tutorials for Python 2.7 and Python 3+. If you would like to use OpenCV 3.0 and Python 3+ on your Ubuntu system, please keep an eye on this blog — I will be posting OpenCV 3.0 and Python 3+ install instructions later this month. But for the time being, let’s stick with Python 2.7.

**How to Install OpenCV 3.0 and Python 2.7+ on Ubuntu**

**UPDATE:**The tutorial you are reading now covers how to install OpenCV 3.0 with Python 2.7+ bindings on ***Ubuntu 14.04***. This tutorial ***still works perfectly***, but if you want to install OpenCV on the newer ***Ubuntu 16.04*** with OpenCV 3.1 and Python 2.7 (or Python 3.5) bindings, please use this freshly updated tutorial:

<http://www.pyimagesearch.com/2016/10/24/ubuntu-16-04-how-to-install-opencv/>

This is the second article in the OpenCV 3.0 install-fest series. Last week we covered how to [install OpenCV 3.0 and Python 2.7+ on OSX](http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/). Today we are going to perform the same OpenCV 3.0 and Python 2.7 installation, only on the Ubuntu operating system. In general, you should find installing OpenCV 3.0 and Python 2.7+ on Ubuntu much easier than installing on OSX.

**Step 1:**

Open up a terminal and update the apt-get  package manager followed by upgrading any pre-installed packages:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2 | $ sudo apt-get update  $ sudo apt-get upgrade |

**Step 2:**

Now we need to install our developer tools:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install build-essential cmake git pkg-config |

The pkg-config  is likely already installed, but be sure to include it just in case. We’ll be usinggit  to pull down the OpenCV repositories from GitHub. The  cmake  package is used to configure our build.

**Step 3:**

OpenCV needs to be able to load various image file formats from disk, including JPEG, PNG, TIFF, etc. In order to load these image formats from disk, we’ll need our image I/O packages:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-dev |

**Step 4:**

At this point, we have the ability to load a given image off of disk. But how do we display the actual image to our screen? The answer is the GTK development library, which the highgui  module of OpenCV depends on to guild Graphical User Interfaces (GUIs):

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install libgtk2.0-dev |

**Step 5:**

We can load images using OpenCV, but what about processing video streams and accessing individual frames? We’ve got that covered here:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev |

**Step 6:**

Install libraries that are used to optimize various routines inside of OpenCV:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install libatlas-base-dev gfortran |

**Step 7:**

Install pip , a Python package manager:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



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

**Step 8:**

Install [virtualenv](https://virtualenv.pypa.io/en/latest/) and [virtualenvwrapper](https://virtualenvwrapper.readthedocs.org/en/latest/). These two packages allow us to create ***separate Python environments*** for each project we are working on. While installing virtualenv  andvirtualenvwrapper  is ***not a requirement***to get OpenCV 3.0 and Python 2.7+ up and running on your Ubuntu system, ***I highly recommend it***and the rest of this tutorial ***will assume you have them installed!***

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2 | $ sudo pip install virtualenv virtualenvwrapper  $ sudo rm -rf ~/.cache/pip |

Now that we have virtualenv  and virtualenvwrapper  installed, we need to update our~/.bashrc  file:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2  3 | # virtualenv and virtualenvwrapper  export WORKON\_HOME=$HOME/.virtualenvs  source /usr/local/bin/virtualenvwrapper.sh |

This quick update will ensure that both virtualenv  and virtualenvwrapper  are loaded each time you login.

To make the changes to our ~/.bashrc  file take effect, you can either (1) logout and log back in, (2) close your current terminal window and open a new one, or preferably, (3) reload the contents of your ~/.bashrc  file:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ source ~/.bashrc |

Lastly, we can create our cv  virtual environment where we’ll be doing our computer vision development and OpenCV 3.0 + Python 2.7+ installation:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ mkvirtualenv cv |

**Step 9:**

As I mentioned above, this tutorial covers how to install OpenCV 3.0 and Python 2.7+ (I’ll have a OpenCV 3.0 + Python 3 tutorial available later this month), so we’ll need to install our Python 2.7 development tools:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ sudo apt-get install python2.7-dev |

Since OpenCV represents images as multi-dimensional NumPy arrays, we better install [NumPy](http://www.numpy.org/) into our cv  virtual environment:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ pip install numpy |

**Step 10:**

Our environment is now all setup — we can proceed to change to our home directory, [pull down OpenCV from GitHub](https://github.com/Itseez/opencv), and checkout the 3.0.0  version:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2  3  4 | $ cd ~  $ git clone https://github.com/Itseez/opencv.git  $ cd opencv  $ git checkout 3.0.0 |

***Update (3 January 2016):****You can replace the 3.0.0  version with whatever the current release is (as of right now, it’s 3.1.0 ). Be sure to check*[*OpenCV.org*](http://opencv.org/category/news)*for information on the latest release.*

As I mentioned [last week](http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/), we also need the [opencv\_contrib repo](https://github.com/itseez/opencv_contrib) as well. Without this repository, we won’t have access to standard keypoint detectors and local invariant descriptors (such as SIFT, SURF, etc.) that were available in the OpenCV 2.4.X version. We’ll also be missing out on some of the newer OpenCV 3.0 features like text detection in natural images:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2  3  4 | $ cd ~  $ git clone https://github.com/Itseez/opencv\_contrib.git  $ cd opencv\_contrib  $ git checkout 3.0.0 |

Again, make sure that you checkout the ***same version*** for opencv\_contrib  that you did foropencv  above, otherwise you could run into compilation errors.

Time to setup the build:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | $ cd ~/opencv  $ mkdir build  $ cd build  $ 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 .. |

***Update (3 January 2016):****In order to build OpenCV 3.1.0 , you need to set -DINSTALL\_C\_EXAMPLES=OFF  (rather than ON ) in the cmake  command. There is a bug in the OpenCV v3.1.0 CMake build script that can cause errors if you leave this switch on. Once you set this switch to off, CMake should run without a problem.*

Notice how compared to [last week](http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/) our CMake command is substantially less verbose and requires less manual tweaking — this is because CMake is able to better automatically tune our install parameters (at least compared to OSX).

Now we can finally compile OpenCV:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

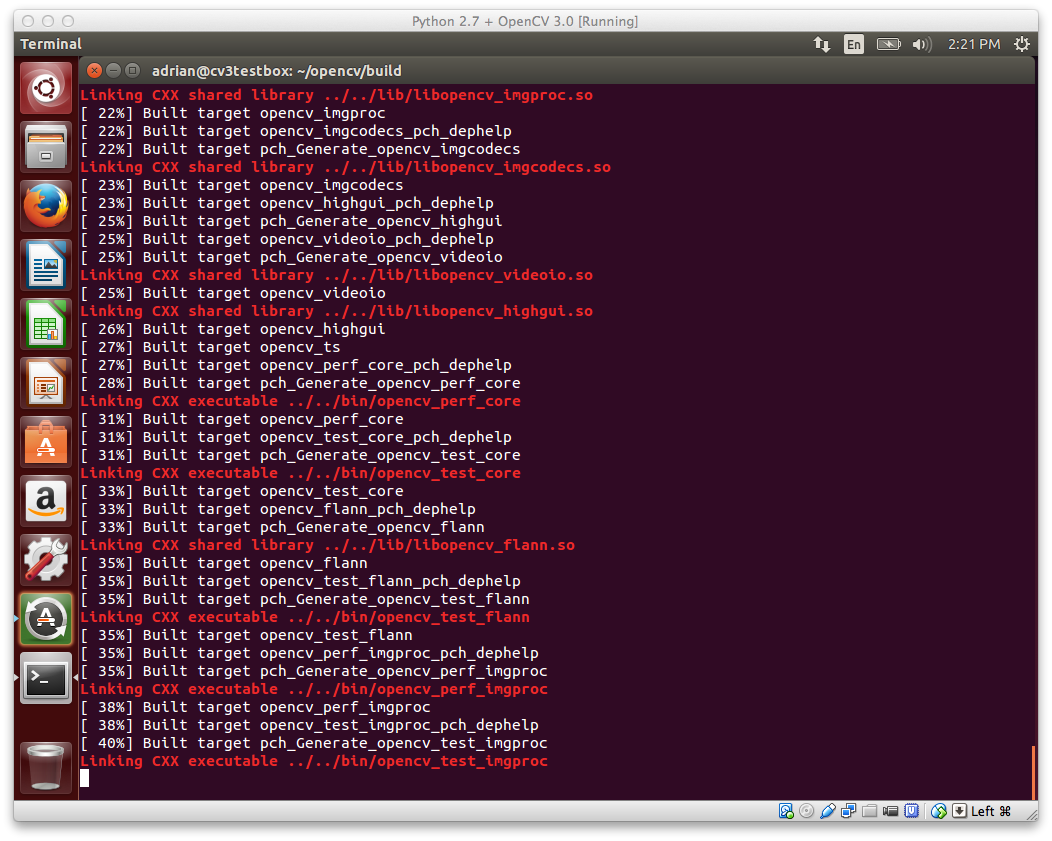
Shell



|  |  |
| --- | --- |
| 1 | $ make -j4 |

Where you can replace the *4* with the number of available cores on your processor to speedup the compilation.

Here’s an example of OpenCV 3.0 compiling on my system:

[](http://www.pyimagesearch.com/wp-content/uploads/2015/06/ubuntu_compiling_opencv3.png)

**Figure 1:** OpenCV 3.0 with Python 2.7+ support compiling on my Ubuntu 14.04 system.

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

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2 | $ sudo make install  $ sudo ldconfig |

**Step 11:**

If you’ve reached this step without an error, OpenCV should now be installed in /usr/local/lib/python2.7/site-packages

However, our cv  virtual environment is located in our home directory — thus to use OpenCV within our cv  environment, we first need to sym-link OpenCV into the site-packages  directory of the cv  virtual environment:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2 | $ cd ~/.virtualenvs/cv/lib/python2.7/site-packages/  $ ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so |

**Step 12:**

**Congratulations! You have successfully installed OpenCV 3.0 with Python 2.7+ bindings on your Ubuntu system!**

To confirm your installation, simply ensure that you are in the cv  virtual environment, followed by importing cv2 :

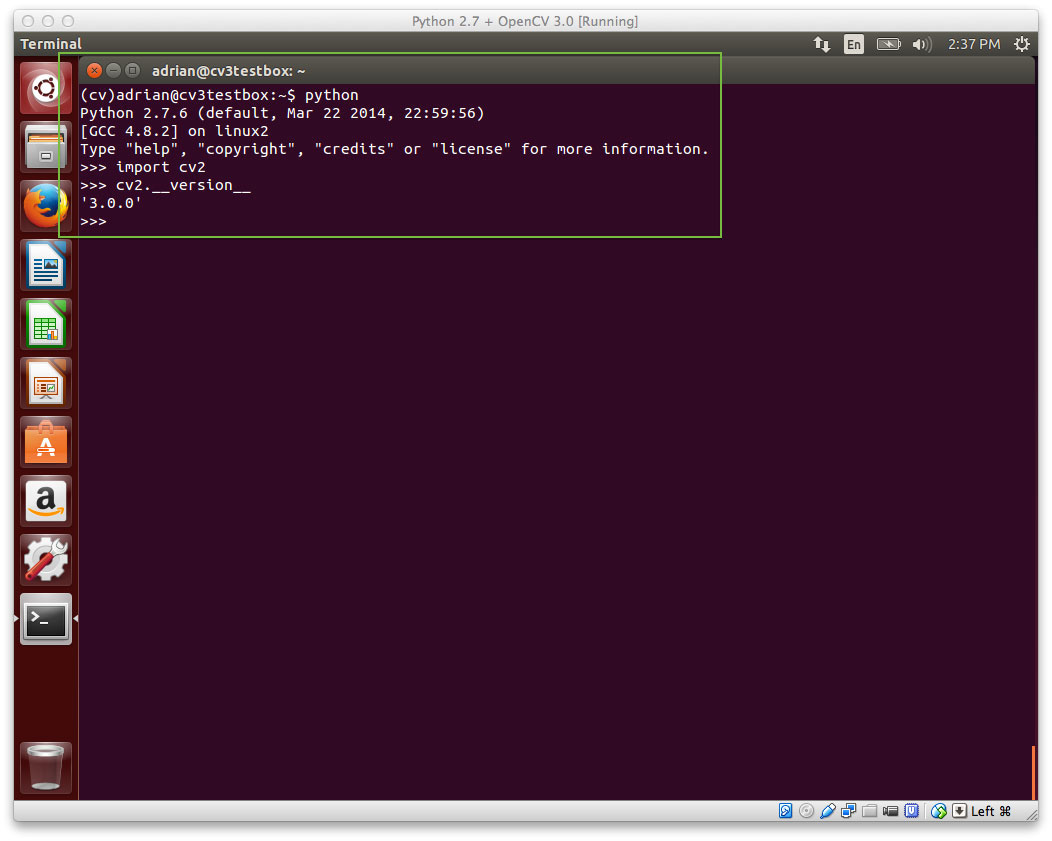
Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1  2  3  4  5 | $ workon cv  $ python  >>> import cv2  >>> cv2.\_\_version\_\_  '3.0.0' |

Here’s an example of demonstrating the OpenCV 3.0 and Python 2.7+ install on my own Ubuntu machine:

[](http://www.pyimagesearch.com/wp-content/uploads/2015/06/ubuntu_opencv3_installed.jpg)

**Figure 2:** OpenCV 3.0 with Python 2.7+ bindings has been successfully installed on Ubuntu!

**Step 13:**

Now that OpenCV has been configured and installed, let’s build a quick Python script to detect the ***red*** game cartridge in the image named games.jpg  below:

[](http://www.pyimagesearch.com/wp-content/uploads/2015/06/games.jpg)

**Figure 3:** Our goal is to detect the red game cartridge (on the *right*) in this image.

Open up your favorite editor, create a new file, name it find\_game.py , and insert the following code:

Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Python



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | # import the necessary packages  import numpy as np  import cv2    # load the games image  image = cv2.imread("games.jpg")    # find the red color game in the image  upper = np.array([65, 65, 255])  lower = np.array([0, 0, 200])  mask = cv2.inRange(image, lower, upper)    # find contours in the masked image and keep the largest one  (\_, cnts, \_) = cv2.findContours(mask.copy(), cv2.RETR\_EXTERNAL,  cv2.CHAIN\_APPROX\_SIMPLE)  c = max(cnts, key=cv2.contourArea)    # approximate the contour  peri = cv2.arcLength(c, True)  approx = cv2.approxPolyDP(c, 0.05 \* peri, True)    # draw a green bounding box surrounding the red game  cv2.drawContours(image, [approx], -1, (0, 255, 0), 4)  cv2.imshow("Image", image)  cv2.waitKey(0) |

You’ll also need to [**download the games.jpg image**](http://www.pyimagesearch.com/wp-content/uploads/2015/06/games.jpg) and place it in the same directory as yourfind\_game.py  file. Once the games.jpg  file has been downloaded, you can execute the script via:

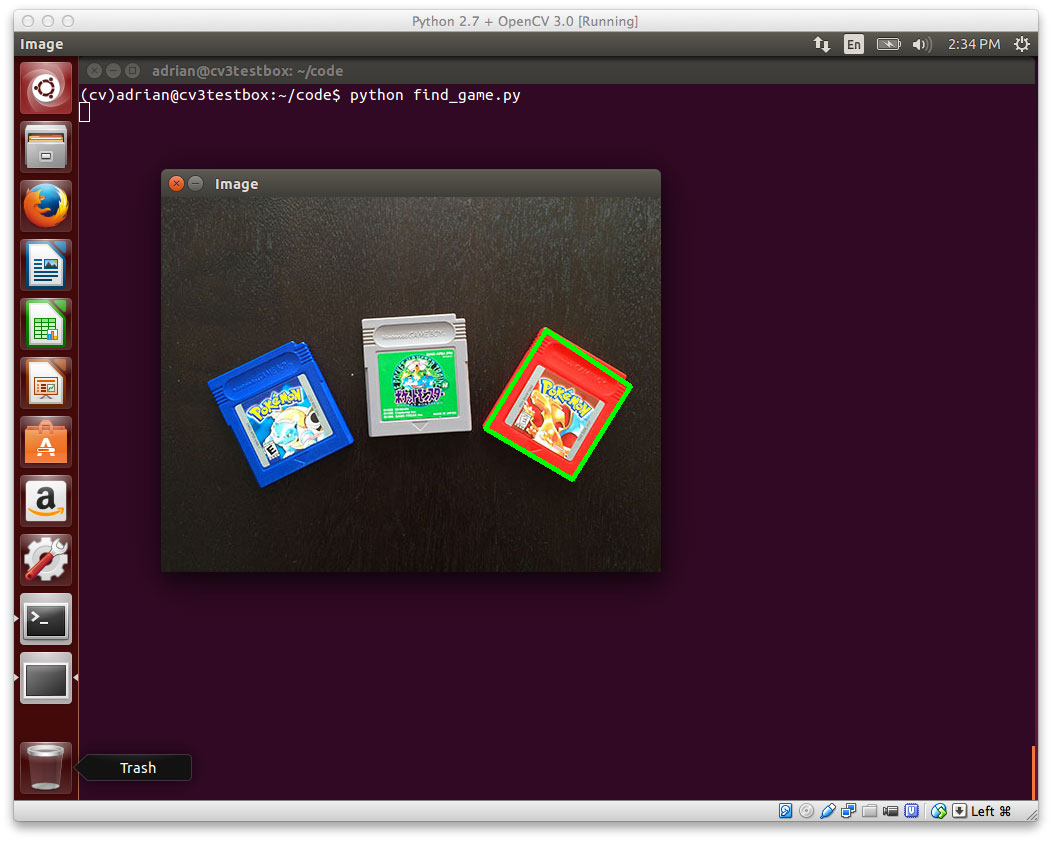
Install OpenCV 3.0 and Python 2.7+ on Ubuntu

Shell



|  |  |
| --- | --- |
| 1 | $ python find\_game.py |

Assuming that you have downloaded the games.jpg  image and placed it in the same directory as our find\_game.py  script, you should see the following output:

[](http://www.pyimagesearch.com/wp-content/uploads/2015/06/ubuntu_found_game.jpg)

**Figure 4:** We have successfully detected the red game cartridge in the image!

Notice how our script was able to successfully detect the red game cartridge in the right portion of the image, followed by drawing a green bounding box surrounding it.

Obviously this isn’t the most exciting example in the world — but it has demonstrated that we have OpenCV 3.0 with Python 2.7+ bindings up and running on our Ubuntu system!