Click here to download the source code to this post

with OpenCV and Python

by **Adrian Rosebrock** on March 30, 2015 in **Raspberry Pi**, **Tutorials**

Over the past year the PyImageSearch blog has had a lot of popular blog posts. Using k-means clustering to find the dominant colors in an image was (and still is) hugely popular. One of my personal favorites, building a kick-ass mobile document scanner has been the most popular PyImageSearch article for months. And the first (big) tutorial I ever wrote, Hobbits and Histograms, an article on building a simple image search engine, still gets a lot of hits today.

But ***by far***, the most popular post on the PyImageSearch blog is my tutorial on installing OpenCV and Python on your Raspberry Pi 2 and B+. It's really, *really* awesome to see the love you and the PyImageSearch readers have for the Raspberry Pi community — and I plan to continue writing more articles about OpenCV + the Raspberry Pi in the future.

Anyway, after I published the Raspberry Pi + OpenCV installation tutorial, many of the comments asked that I continue on and discuss ***how to access the Raspberry Pi camera using Python and OpenCV.***

module. And

Click here to download the source code to this post

Read on to find out how…

IMPORTANT: *We'll be building off my original tutorial on installing OpenCV and Python on your Raspberry Pi. If you do not already have OpenCV + Python configured and installed correctly on your Raspberry Pi, please take the time now to review the tutorial and setup your own Raspberry Pi with Python + OpenCV.*

Looking for the source code to this post?
Jump right to the downloads section.

OpenCV and Python versions:
This example will run on **Python 2.7/Python 3.4+** and **OpenCV 2.4.X/OpenCV 3.0+**.

Step 1: What do I need?

To get started, you'll need a Raspberry Pi camera board module.

I got my 5MP Raspberry Pi camera board module from Amazon for under $30, with shipping. It's hard to believe that the camera board module is almost as expensive as the Raspberry Pi itself — but it just goes to show how much hardware has progressed over the past 5 years. I also picked up a camera housing to keep the camera safe, because why not?

Assuming you already have your camera module, you'll need to install it. Installation is very simple and instead of creating my own tutorial on installing the camera board, I'll just refer you to the official Raspberry Pi camera installation guide:

Assuming your camera board and properly installed and setup, it should look something like this:

[Click here to download the source code to this post]

Figure 1: Installing the Raspberry Pi camera board.

Step 2: Enable your camera module.

Now that you have your Raspberry Pi camera module installed, you need to enable it. Open up a terminal and execute the following command:

Accessing the Raspberry Pi Camera with OpenCV and Python — Shell

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

This will bring up a screen that looks like this:

[Click here to download the source code to this post]

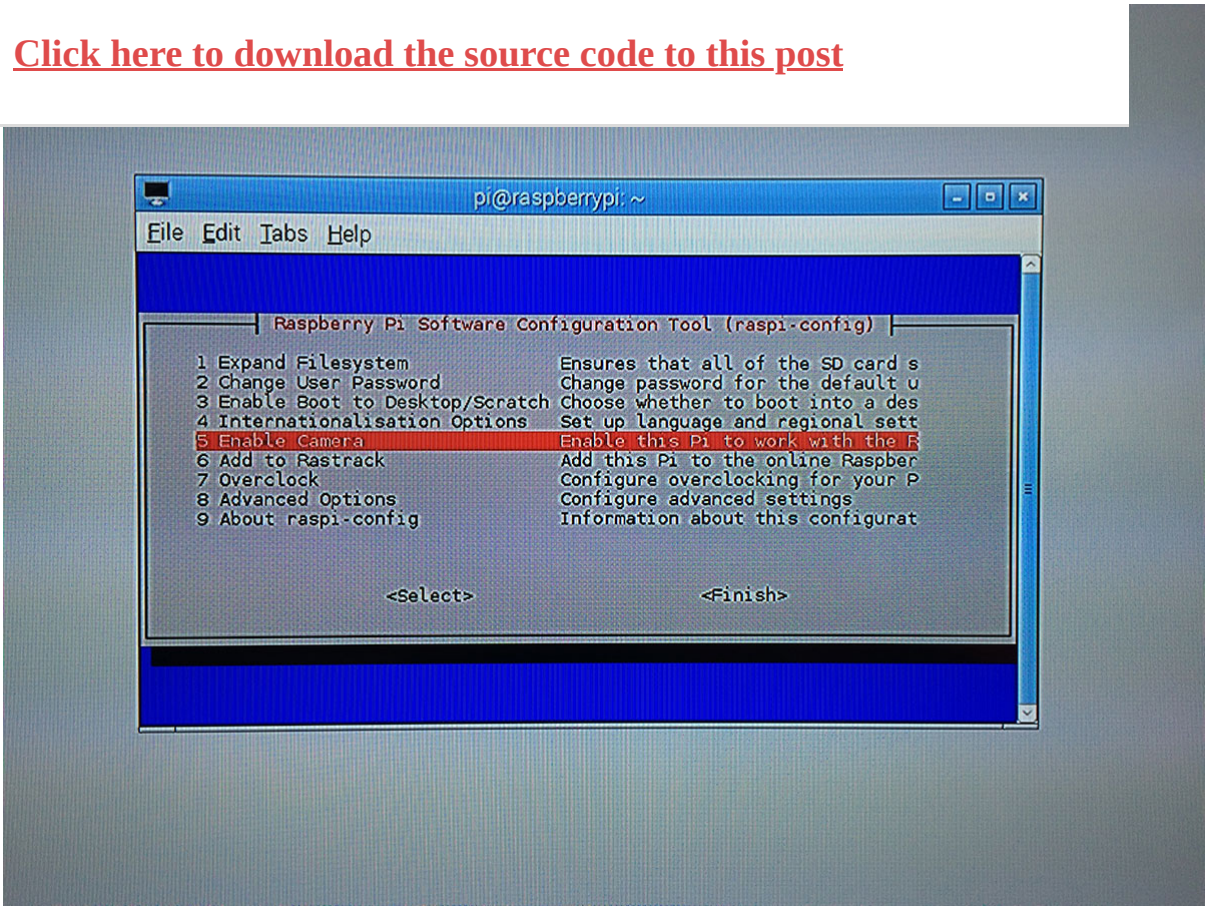

Figure 2: Enabling the Raspberry Pi camera module using the raspi-config command.

Use your arrow keys to scroll down to **Option 5: Enable camera**, **hit your enter key** to enable the camera, and then arrow down to the **Finish** button and hit enter again. Lastly, **you'll need to reboot your Raspberry Pi** for the configuration to take affect.

Step 3: Test out the camera module.

Before we dive into the code, let's run a quick sanity check to ensure that our Raspberry Pi camera is working properly.

Note: *Trust me, you'll want to run this sanity check before you start working with the code. It's* ***always good*** *to ensure that your camera is working prior to diving into OpenCV code, otherwise you could easily waste time wondering when your code isn't working correctly when it's simply the camera module itself that is causing you problems.*

Anyway, to run my sanity check I connected my Raspberry Pi to my TV and positioned it such that it was pointing at my couch:

Click here to download the source code to this post

Figure 3: Example setup of my Raspberry Pi 2 and camera.

And from there, I opened up a terminal and executed the following command:

Accessing the Raspberry Pi Camera with OpenCV and Python — Shell

```
$ raspistill -o output.jpg
```

This command activates your Raspberry Pi camera module, displays a preview of the image, and then after a few seconds, snaps a picture, and saves it to your current working directory as `output.jpg` .

Here's an example of me taking a photo of my TV monitor (so I could document the process for this tutorial) as the Raspberry Pi snaps a photo of me:

Click here to download the source code to this post

Figure 4: Sweet, the Raspberry Pi camera module is working!

And here's what `output.jpg` looks like:

Figure 5: The image captured using the raspi-still command.

me more

[Click here to download the source code to this post]

Step 4: Installing picamera.

So at this point we know that our Raspberry Pi camera is working properly. But how do we interface with the Raspberry Pi camera module using Python?

The answer is the picamera module.

Remember from the previous tutorial how we utilized `virtualenv` and `virtualenvwrapper` to cleanly install and segment our Python packages from the the system Python and packages?

Well, we're going to do the same thing here.

Before installing `picamera`, be sure to activate our `cv` virtual environment:

```
Accessing the Raspberry Pi Camera with OpenCV and Python                Shell
1 $ source ~/.profile
2 $ workon cv
```

By sourcing our `.profile` file, we ensure that we have the paths to our virtual environments setup correctly. And from there we can access our `cv` virtual environment.

Note: *If you are installing the the `picamera` module system wide, you can skip the previous commands. However, if you are following along from the previous tutorial, you'll want to make sure you are in the `cv` virtual environment before continuing to the next command.*

And from there, we can install `picamera` by utilizing pip:

```
Accessing the Raspberry Pi Camera with OpenCV and Python                Shell
1 $ pip install "picamera[array]"
```

IMPORTANT: *Notice how I specified `picamera[array]` and not just `picamera`.*

Why is this so important?

While the standard `picamera` module provides methods to interface with the camera, we need the (optional) `array` sub-module so that we can utilize OpenCV. Remember, when using Python bindings, OpenCV represents images as NumPy arrays — and the `array` sub-module allows us to obtain NumPy arrays from the Raspberry Pi camera module.

Assuming that your install finished without error, you now have the `picamera` module (with NumPy array support) installed.

Step 5: Accessing a single image of your Raspberry Pi using Python and OpenCV.

Alright, now we can finally start writing some code!