Setup OpenCV 3.3.0 on Rapberry Pi 3

OpenCV is an open-source framework for computer vision. It's used for self-driving cars, surveillance, CCTV, traffic flow measurement, autonomous robots, license plate recognition, gesture recognition, and just about any other application where a computer needs to "see".

You'll have seen OpenCV in action whenever you've watched a surveillance scene in a film or documentary showing boxes being automatically drawn around moving people or cars on a computer screen.

OpenCV 3.3.0 runs well on a Raspberry Pi 3 using either the Pi camera module, USB webcam or USB video grabber and CCTV camera. Our examples run in Python 3. We have different code examples for the Pi camera and USB webcam/video grabber.

To start, make sure you have a 16GB or 32GB Micro SD card with the latest Raspbian Stretch Desktop version installed.

Open a terminal window and type:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

If you're using the Raspberry Pi camera, enable the camera module in the Desktop Preferences Menu → Raspberry Pi Configuration → Interfaces → Camera Enable (I also enabled SSH for remote access to the Pi from my PC). Restart your Pi.

Then follow this guide to install and build OpenCV 3 for Python 3 on your Raspberry Pi 3 running Raspbian Stretch:

https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/

When following the guide you'll find these extra tips useful:

1 - Ignore the line near the end where he says to do this:

```
rm -rf opencv-3.3.0 opencv_contrib-3.3.0
```

2 – Just before you compile OpenCV he tells you to edit /etc/dphys-swapfile, but not how:

```
sudo nano /etc/dphys-swapfile
```

3 – Standard instructions worked fine for previous Stretch (2017-11-29) but on the latest build of Stretch (2018-03-13) with my new Pi 3 B+ I had to add an extra line to ./profile, else I got virtualenvwrapper errors when trying to create the cv virtual environment on python 3:

```
# virtualenv and virtualenvwrapper
VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
export WORKON_HOME=/home/pi/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

Following the guide and compiling OpenCV takes about two hours in total on a Pi 3 or Pi 3 B +.

Test everything's working correctly with:

```
source ~/.profile
workon cv
python
import cv2
cv2.__version__
```

Quit Python with Ctrl-D.
Then follow this guide to watch live video in a frame on your desktop from the Pi Camera:

https://www.pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/

If you didn’t follow the link above, make sure you do this:

```
pip install imutils
```

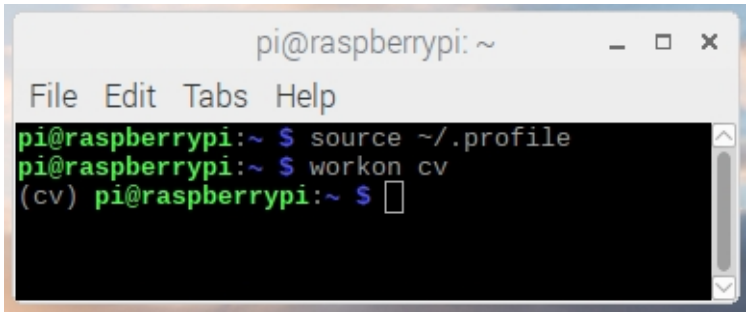
Also, if you didn’t follow the link above and have a Raspberry Pi camera, make sure you do this before using our Raspberry Pi specific python scripts - while in the CV environment:

```
pip install "picamera[array]"
```

Each time you reboot your Pi and open a terminal window remember to issue these commands first:

```
source ~/.profile
workon cv
```

and the prompt will change and have cv at the front.



you can leave the CV virtual environment with:

```
deactivate
```

OpenCV buzzwords.

When you start wanting to know how to do different things with OpenCV, the obvious place to start is Google. It's worth mentioning that OpenCV has been around quite a while and a lot of the examples you'll find are for the older OpenCV 2 / Python 2 and these may throw up errors when run on OpenCV 3 or Python 3.

Thankfully, there are up to date examples for OpenCV 3 at https://docs.opencv.org. All our own downloadable examples all work with Python 3 and OpenCV 3 on both Pi camera and USB webcam.

Here are a few useful OpenCV phrases you might not be familiar with, that will help you when Googling for examples:

ROI or Region of interest – Part of an image you want to isolate using x,y,w,h co-ordinates.

Contours – The part of an image that matches your search criteria or contains movement. You'll often draw a box or circle around the contour. Contours are expressed by co-ordinates x,y,w,h – where x and y are the top left corner and w and h are the width and height of the object. You can choose to ignore contours that are smaller than a certain value, or just draw boxes around contours that are in the correct part of the display.

Haar Cascade – A set of rules used to find a feature in the image. The most common would be a Haar Cascade that finds faces in an image. As well as all the pre-definded Haar Cascades, you can use a Trainer & Detector to create your own. Our examples feature a face detctor Python script.

Background subtraction – we want to isolate any new features present in the image, so we have a background template and look for new features in the current frame. Our examples use the MOG2 method for doing this. In OpenCV 3.3.0 the (older?) MOG and GMG methods aren't available by default – so if you're working from an example you found on the internet try MOG2 instead. MOG2 generates quite a bit of unwanted background noise which is filtered out in our examples using Erosion and Dilation filters.
Example video : https://www.youtube.com/watch?v=T-L9FoH3D9w

Erosion – we use this to get rid of any Contours smaller than a certain size. Our mog.py example does 2 iiterations which then leaves us with objects we're interested in, but cuts out bushes and plants blowing around in the wind.

Dilation – bulks up the objects remaining in the frame, after we've done the erosion.

Our OpenCV Python examples

You can download our OpenCV example Python scripts with

```
wget www.securipi.co.uk/opencv-stuff.zip
unzip opencv-stuff.zip
```

to see the list of files

```
ls *.py
```

Here's what each script does:

test-video-picam.py – shows a live feed from the Pi camera on your Desktop
find-faces-picam.py – displays a box around any face found in the Pi camera feed
find-faces-picam2.py – as above but records time stamped photos to SD card too.
mog-picam-1.py - Uses Mog2 background subtractor, takes a photo of large items on the left.
mog-picam-2-.py – as above but with erosion and dilation filters too.

Also similar examples for USB webcam or USB video grabber, without the picam titles:

test-video.py – shows live vide feed from USB webcam or USB video grabber.
find-faces.py – finds faces in the video feed and draws a box around them.
mog.py - Uses Mog2 background subtractor to find intruders and cars on our CCTV camera.
mog2.py – as above but takes photos of intruders in the right hand side of picture.
mog3.py – Uses Mog2 bacground subtractor. Counts number of cars that have driven past and which direction they were going in the frame. (see photo on next page)
mog4.py – as above but also records each frame with a car in to a photo file. Press c to create a timelapse video from the photos, and then press p to play the timelapse video.

By comparing test-video-picam.py and test-video.py you can see exactly how to modify other Python scripts for Raspberry Pi camera or USB webcam.

To run the script test-video-picam.py you'd type into the terminal:

```
python test-video-picam.py
```

You can quit the script by clicking on the terminal window and pressing Ctrl-C.

This is the live feed from my own CCTV camera. I have the CCTV box rigged up to the Raspberry Pi using the USB video grabber we sell on Amazon UK and US :
https://www.amazon.co.uk/dp/B072Q4MNKM or
https://www.amazon.com/dp/B072Q4MNKM

The script mog.py looks for any cars in the top left of the photo and also looks for intruders on the far right of the photo. It stores photos of any intruders to a time and date stamped photo file on the SD card. I plan to add a proper car counting function soon, along with an air pollution sensor.

Update 24th March – since making example python scripts specifically for Raspberry Pi camera and USB webcam, I've since discovered you can make the Raspberry Pi camera behave like a USB Webcam (visible as /dev/video0 device to the system) by doing this :

```
sudo modprobe bcm2835-v4l2
```

you can see it's worked by doing

```
ls /dev/video*
```

So once you do that, the scripts that I wrote for USB webcam will also work with the Raspberry Pi camera. It also means many of the OpenCV python examples you'll find on Google will work fine with the Raspberry Pi camera too.

Other useful OpenCV resources:

https://pythonprogramming.net/raspberry-pi-camera-opencv-face-detection-tutorial/

https://docs.opencv.org/3.3.0/dc/d2e/tutorial_py_image_display.html

https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html

https://www.pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/

https://docs.opencv.org/3.3.1/d4/d73/tutorial_py_contours_begin.html

https://docs.opencv.org/3.1.0/d3/df2/tutorial_py_basic_ops.html
(ROI region of interest info)

https://stackoverflow.com/questions/46447073/how-to-ignore-part-of-a-image-during-feature-extraction-in-opencv
(more ROI info)

https://pythonprogramming.net/loading-video-python-opencv-tutorial/

https://pythonprogramming.net/drawing-writing-python-opencv-tutorial/?completed=/loading-video-python-opencv-tutorial/

Quick ANPR / ALPR car license plate recognition setup on Raspberry Pi 3.

```
curl -sSL https://get.docker.com | sh

sudo docker build -t openalpr
https://github.com/openalpr/openalpr.git

wget http://plates.openalpr.com/h786poj.jpg

sudo docker run -it --rm -v $(pwd):/data:ro openalpr -c eu
h786poj.jpg

raspistill -w 640 -h 480 -o mm52enw.jpg

sudo docker run -it --rm -v $(pwd):/data:ro openalpr -c eu
mm52enw.jpg
```

If you found this document useful, please follow us on Twitter @securipi

Our eBay shop contains lots of components and project kits for Raspberry Pi and is here:
http://stores.ebay.co.uk/ConvertStuffUK