Raspy Install

OS and Internet

Flash Raspian Stretch to SD Card
Raspian Stretch download: https://www.raspberrypi.org/downloads/raspbian/
Flashing Tool: https://etcher.io/

Connect raspi to monitor via hdmi with mouse and keyboard
`sudo nano /etc/dhcpcd.conf`

Edit file and enter the following:
```
interface eth0
static ip_address=192.168.137.100/24
static routers=192.168.137.1
static domain_name_servers=8.8.8.8
```

PC Ethernet Adapter 192.168.137.1/24
Share WiFi Adapter with Ethrnet-Adapter

Putty to Raspi via SSH

Default user pi, pw raspberry

Expand file system:
`sudo raspi-config` "Advanced Options" "Expand filesystem" and reboot system

Dependencies

Update and upgrade any existing packages:
`sudo apt-get update && sudo apt-get upgrade`

Intall some developer tools, including CMake, which helps us configure the OpenCV build process:
`sudo apt-get install build-essential cmake pkg-config`

Install image and video I/O packages that allow us to load various image file formats from disk. Examples of such file formats include JPEG, PNG, TIFF, etc.:
```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
sudo apt-get install libxvidcore-dev libx264-dev
```

The OpenCV library comes with a sub-module named highgui which is used to display images to our screen and build basic GUIs. In order to compile the highgui module, we need to install the GTK development library:
`sudo apt-get install libgtk2.0-dev libgtk-3-dev`

Many operations inside of OpenCV (namely matrix operations) can be optimized further by installing a few extra dependencies:
`sudo apt-get install libatlas-base-dev gfortran`

Lastly, let's install both the Python 2.7 header files so we can compile OpenCV with Python bindings:
`sudo apt-get install python2.7-dev`

Download OpenCV

Download opencv from git

```
cd ~
wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip
unzip opencv.zip
```

and opencv_contrib for full install of OpenCV 3

```
wget -O opencv_contrib.zip
https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip
unzip opencv_contrib.zip
```

install pip, a Python package manager:

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
```

It's standard practice in the Python community to be using virtual environments of some sort:

```
sudo pip install virtualenv virtualenvwrapper
sudo rm -rf ~/.cache/pip
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
echo -e "\n# virtualenv and virtualenvwrapper" >> ~/.profile
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.profile
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.profile
```

Creating Python virtual environment

IMPORTANT: Any Python packages in the *global* site-packages directory *will not* be available to the cv virtual environment. Similarly, any Python packages installed in site-packages of cv *will not* be available to the global install of Python. Keep this in mind when you're working in your Python virtual environment and it will help avoid a lot of confusion and headaches.

```
mkvirtualenv cv -p python2
```

If you ever reboot your Raspberry Pi; log out and log back in; or open up a new terminal, you'll need to use the workon command to re-access the cv virtual environment.

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

Tipp: `deactivate` to exit (cv)

To continue installing… following commands in (cv) environment:

```
pip install numpy
```

Compile and Install OpenCV

To continue installing… following commands in (cv) environment:

```
cd ~/opencv-3.3.0/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE \
       -D CMAKE_INSTALL_PREFIX=/usr/local \
       -D INSTALL_PYTHON_EXAMPLES=ON \
       -D INSTALL_C_EXAMPLES=ON \
       -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules \
       -D BUILD_EXAMPLES=ON ..
```

Output should look something like this:

Finally we can now compile OpenCV - This will take approx.. 4h!

```
make
```

Once this is finished we can install OpenCV on our Raspi

```
sudo make install
sudo ldconfig
```

OpenCV should now be installed in /usr/local/lib/python2.7/site-packages

Final Step is to sym-link OpenCV into cv virtual environment for Python 2.7:

```
cd ~/.virtualenvs/cv/lib/python2.7/site-packages/
ln -s /usr/local/lib/python2.7/dist-packages/cv2.so cv2.so
```

To check if everything worked out open a new terminal:

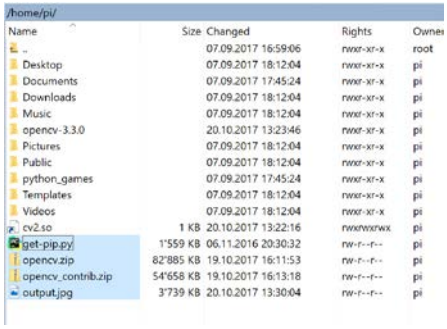
```
source ~/.profile
workon cv
python
import cv2
>>> cv2.__version__
'3.3.0'
>>>
```

Tipp: `exit()` to exit Python

And if your sure everything worked out, delete downloaded files to free up some space:

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

Connected via WinSCP SFTP and deleted:



Camera

Enable your camera module

`sudo raspi-config` “Enable Camera” and reboot Raspi

Check if Camera is connected correctly:

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

 WUHU!

To use camera from python. 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.

```
source ~/.profile
workon cv
pip install "picamera[array]"
```

Und Thadaa!

Now we only have to save the Image as backup:

https://sourceforge.net/projects/win32diskimager/