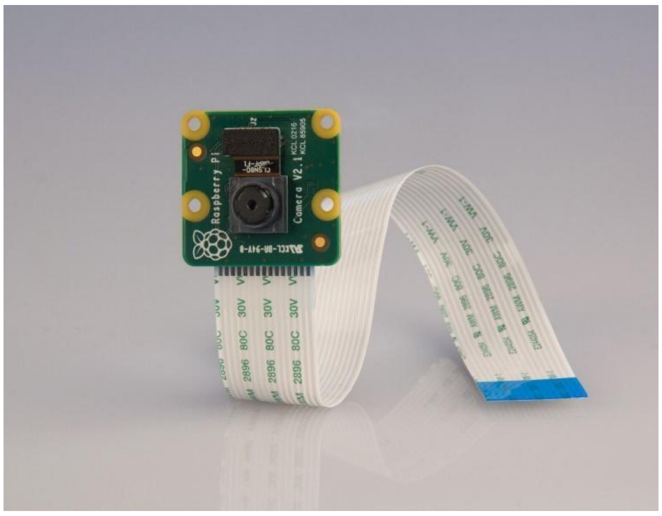
Internet of Things (IoT)
Topic 6: Pi Camera Module

What you will need

- Raspberry Pi Camera Module

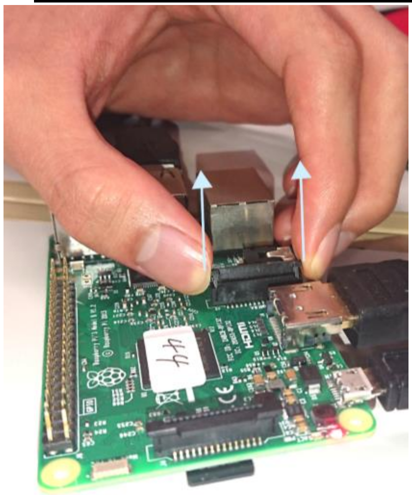


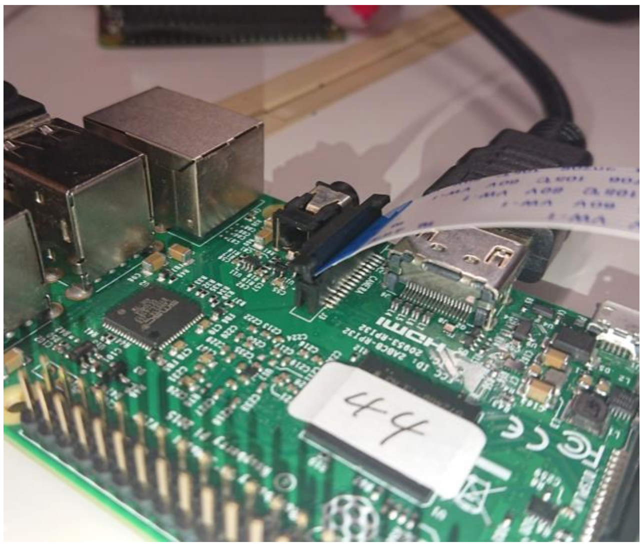
Connect the Camera Module

First of all, please power off the Raspberry Pi!!!

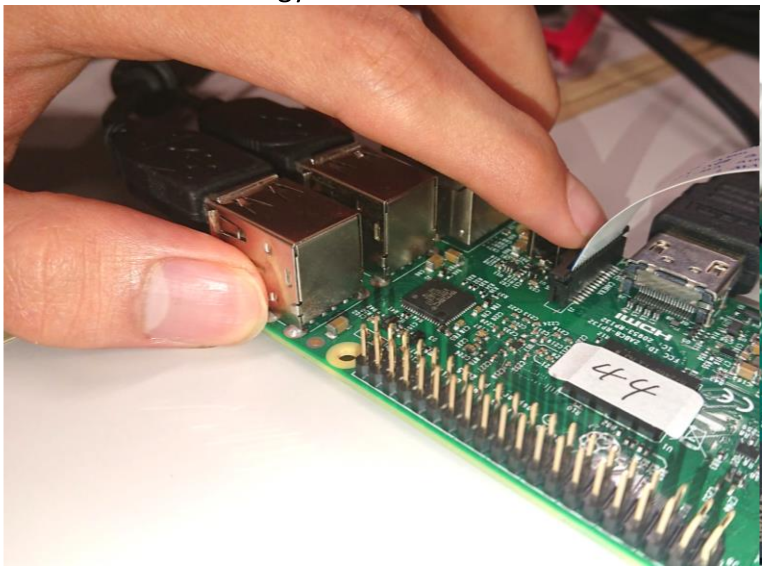
1. Locate the camera port and connect the camera:
   Follow https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/3
   or the steps below

a. **Pull that black thingy up to open the camera slot** b. insert the camera into the connector

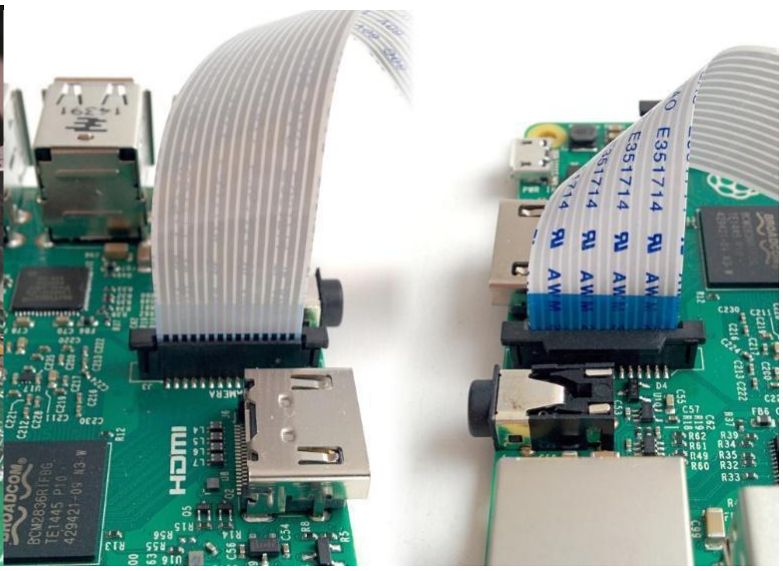




c. Push the black thingy down

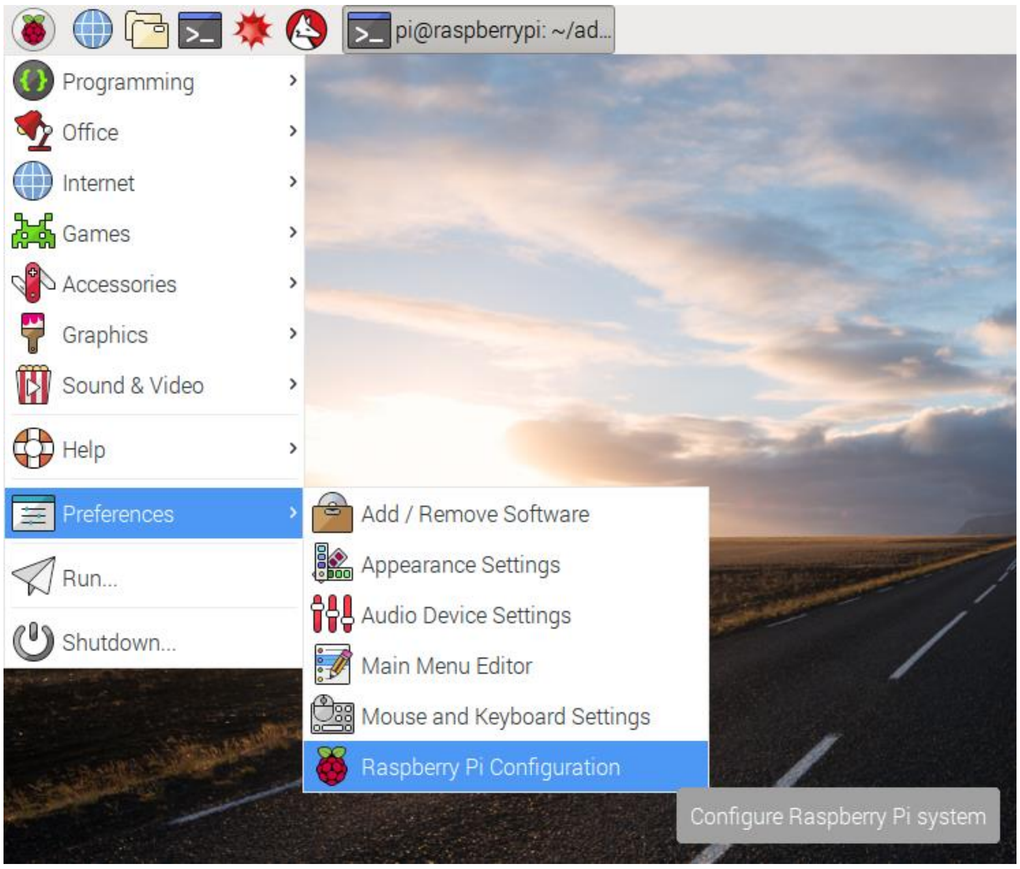


d. Done

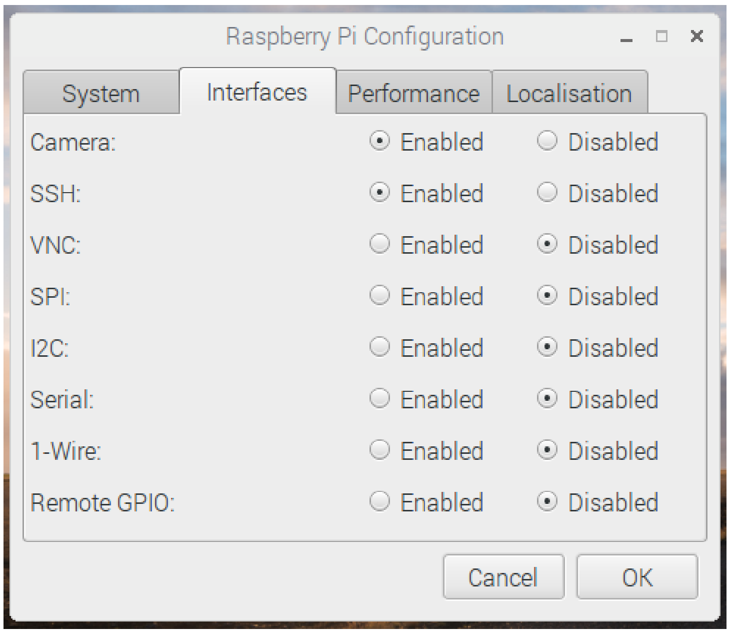


2. Turn on the Raspberry Pi

3. Open the Raspberry Pi Configuration Tool from the main menu:



4. Enable the camera software, and then **reboot** your Raspberry Pi



Camera:	[✓ Enabled]	Disabled
SSH:	[✓ Enabled]	Disabled
VNC:	Enabled	[✓ Disabled]
SPI:	Enabled	[✓ Disabled]
I2C:	Enabled	[✓ Disabled]
Serial:	Enabled	[✓ Disabled]
1-Wire:	Enabled	[✓ Disabled]
Remote GPIO:	Enabled	[✓ Disabled]

Testing the camera

Open a Terminal window and enter the following command:

raspistill -o image1.jpg

Adjust the camera to point to yourself. You should be able to see a camera preview on the screen, and then an image would be captured. Click the file manager icon and open **image1.jpg** to look at your own photo.

Taking a picture with Python

Try and run the following Python code to use the Camera Module for taking still pictures.

camera.py

```
from picamera import PiCamera
import time

camera = PiCamera()

# uncomment the following line to rotate the image if your preview was upside-down!
# camera.rotation = 180
camera.start_preview()
time.sleep(5)
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

Recording video

Now, let’s try to record a video with your Pi camera! Amend the above code (**camera.py**) to replace **capture()** with **start_recording()** and **stop_recording()**.

video.py

```
from picamera import PiCamera
import time

camera = PiCamera()

# uncomment the following line if the preview was upside-down
# camera.rotation = 180
camera.start_preview()
camera.start_recording('/home/pi/video.h264')
time.sleep(10)
camera.stop_recording()
camera.stop_preview()
```

Try to play your video in the Terminal with the following command:

omxplayer video.h264

Note: The video should play, and it may actually play at a faster speed than what has been recorded, due to omxplayer’s faster frame rate.

Facial Recognition with OpenCV and Raspberry Pi

In this exercise, we will run a simple facial recognition application with the use of OpenCV (Open Source Computer Vision) Library [4] on Raspberry Pi. A demo video of this application can be found here:

https://drive.google.com/file/d/192IgDhxHkF-8ahwXwWfrnUuseX1gwC3M/view?usp=sharing

The OpenCV library takes a rather long time to be installed on a Raspberry Pi. To save time, we have installed the OpenCV in your Pi.

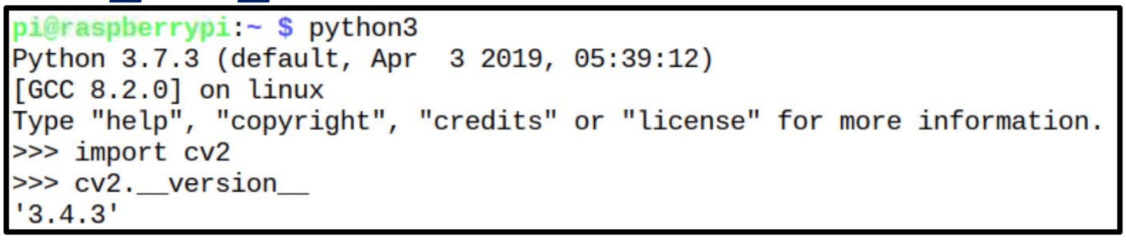
Step 1: Prepare the working environment

Let's first verify OpenCV is properly installed. Open a terminal, and try to import the OpenCV library, and check the version:

python3

>>> ***import cv2***

>>> ***cv2.__version__***



```
pi@raspberrypi:~ $ python3
Python 3.7.3 (default, Apr  3 2019, 05:39:12)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.3'
```

If you see no error and '3.4.3' (or similar numbers) being printed out, OpenCV has been successfully installed.

Regardless of whether you have OpenCV installed or not, exit the Python Interpreter first:

>>> *exit()*

If you see errors, that means OpenCV is not installed properly yet, please try to install it by yourself, by executing these commands in terminal one-by-one, or ask tutors for help:

sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test python3-pyqt5
sudo apt-get install libatlas-base-dev
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng-dev
sudo pip3 install picamera
sudo pip3 install opencv-contrib-python

Now with OpenCV installed properly, we are ready to use it with a simple facial recognition application. First, let's download the codes we are going to use for this exercise from GitHub[5] to Desktop.

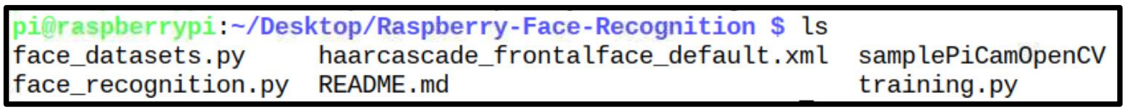
cd ~/Desktop/ #actually you can simply type "**cd ~/De**" and press **[tab]** on keyboard and it will auto complete for you

git clone https://github.com/trieutuanvnu/Raspberry-Face-Recognition

Let's go to the folder with the downloaded codes.

cd Raspberry-Face-Recognition

Here we can see several documents



```
pi@raspberrypi:~/Desktop/Raspberry-Face-Recognition $ ls
face_datasets.py     haarcascade_frontalface_default.xml  samplePiCamOpenCV
face_recognition.py  README.md                            training.py
```

Now let's see how this facial recognition system should work. Consider how we humans can recognize each other's faces, we can break the process down to 3 steps:

1. see a person's face and remember that → [Collect Data]
2. associate that face to that person → [Learn]
3. see that person's face again, and name that person → [Recognize]

This process is quite similar to how the machine would recognize your face, and the 3 documents: *face_datasets.py, training.py and face_recognition.py* are responsible for the above 3 steps respectively.

haarcascade_frontalface_default.xml is to be used as a face detector. (Here we clarify that a face detector tries to tell if something is a face or not, a face recognizer tries to tell if a face it sees is someone.) After all a machine has to first know what a face is, before it can start to recognize faces. This face detector is already built for you so you don't have to worry about it.

Step 2: Data Collection (*face_datasets.py)*

Step 2.1:

When you run *face_datasets.py*, it will try to collect data for 1 person's face. First, let's inspect the code in *face_datasets.py* using idle by entering the following command*:*

idle face_datasets.py &

In line 11 (line number and column number are at the bottom right corner of the idle window), we are giving a unique ID to the person we are going to collect data from. If we are going to collect data from the first person, just go ahead and change line 11 to:

    face_id = 1

Line 29 is a start of a for loop, that is doing: For each detected face, save it.

In line 35 we can see a “count” variable counting the number of images of a person’s face we have taken. For debugging purposes, let’s add the following to line 36:

print(count)

Remember to add 2 tabs before the statement for correct indentation.

Line 38 states that we are going to save the image into a folder called “dataset”. So now let’s create a folder called “dataset”. We go back to the terminal and enter the following command:

mkdir dataset

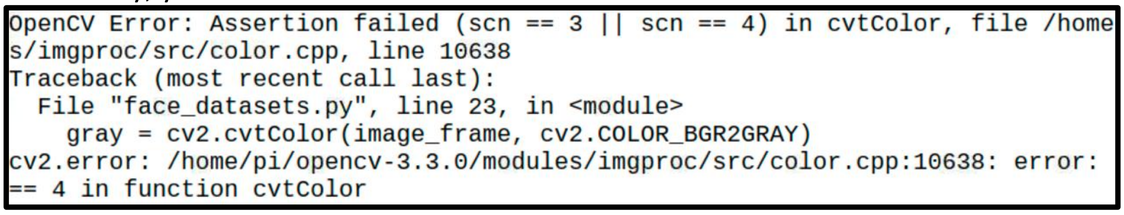
Lines 48-49 terminates the program loop when the number of images collected reaches a certain threshold. Right now, the threshold is 100, you can experiment with different numbers but I’ll just leave it there for now.

Step 2.2:

Now let’s try to run *face_datasets.py*:

python3 face_datasets.py

Most likely, you will encounter this error:



```
OpenCV Error: Assertion failed (scn == 3 || scn == 4) in cvtColor, file /home
s/imgproc/src/color.cpp, line 10638
Traceback (most recent call last):
  File "face_datasets.py", line 23, in <module>
    gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)
cv2.error: /home/pi/opencv-3.3.0/modules/imgproc/src/color.cpp:10638: error:
== 4 in function cvtColor
```

Please try to search online for the solution (You can try to search for “OpenCV Error: Assertion failed (scn == 3 || scn == 4) in cvtColor” on Google). If you really cannot find the solution, please refer to the cheat sheet in the last section.

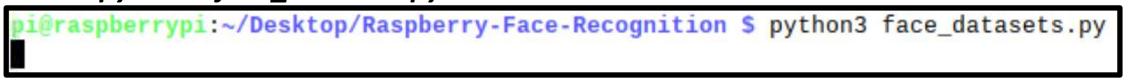
If there is an error saying that the camera cannot be found, try to change line 5 to:

```
vid_cam = cv2.VideoCapture(-1)
```

Step 2.3:

After you have solved that, try to run the *face_datasets.py* again:

python3 face_datasets.py



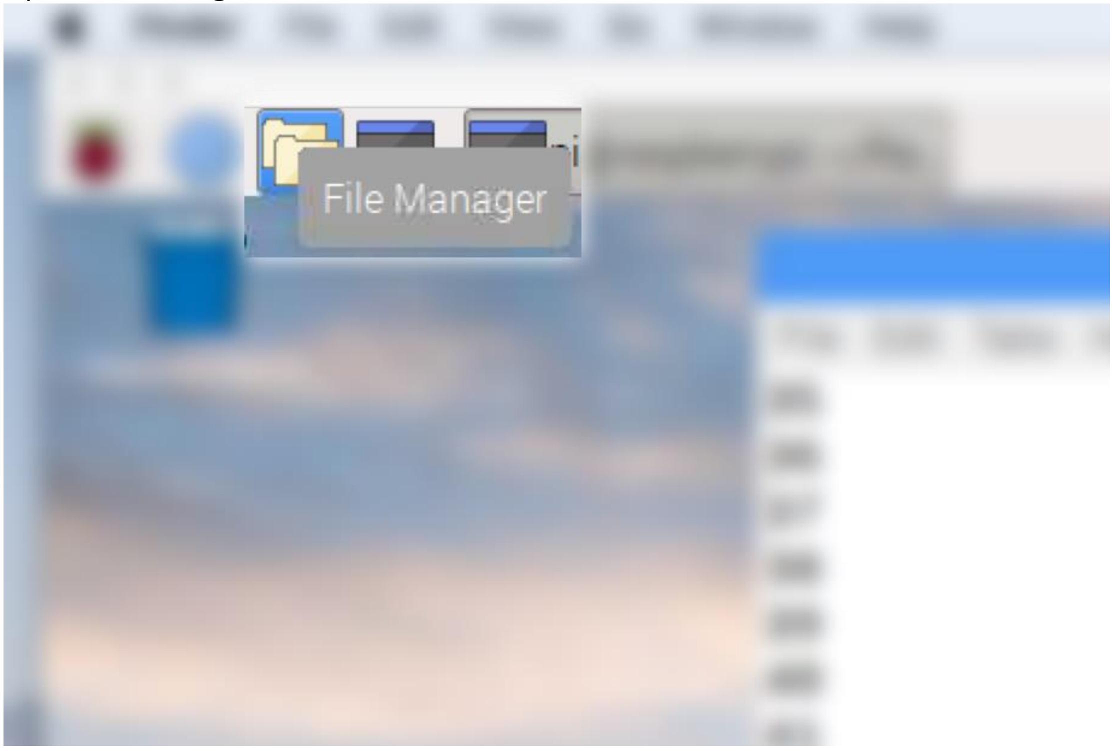
```
pi@raspberrypi:~/Desktop/Raspberry-Face-Recognition $ python3 face_datasets.py
█
```

If this is what you see, you are on the right track. **Go Ahead and turn the camera to yourself and face the camera.** Keep camera as stationary as possible. You should now see a preview on the screen, with a blue square surrounding your face. In the terminal, you should see some numbers popping out, they are indicating the number of images of your face it has captured.

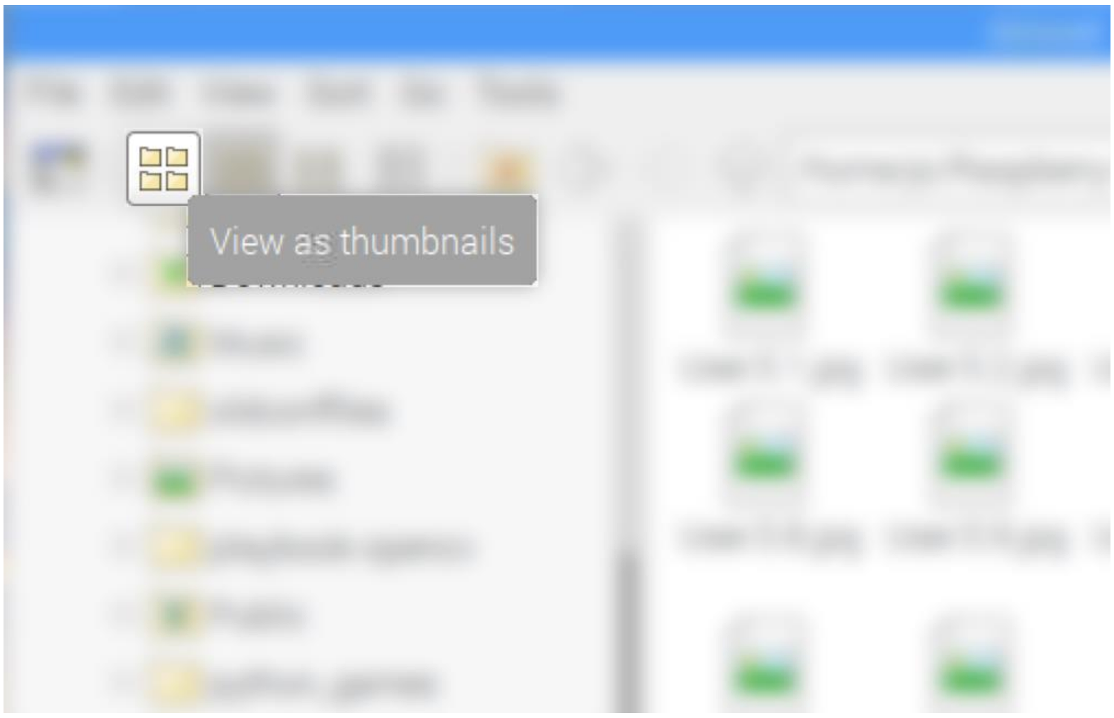
Please continue facing the camera until it has captured enough images (same as the number you specified in Line 49). You can try with different distance between your face and the camera, and also different facial expressions, so as to increase the diversity of the data the machine can collect.

Step 2.4:
After the program terminates, let's see how the data collected look like.

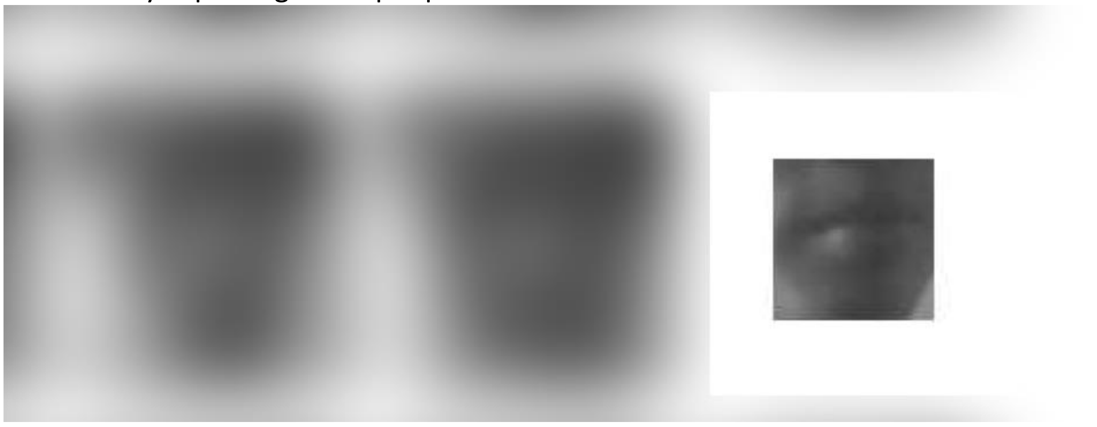
Open **File Manager**



Navigate to */home/pi/Raspberry-Face-Recognition/dataset*, then click on **View as thumbnails**.



Delete any weird photos you see, like the one below, capturing only the mouth, or accidentally capturing other people's face.



The data collection step is now finished.

Step 3: Machine Learning Model Training (training.py)

Step 3.1:

Now we have to train the machine so that next time when it sees a similar face to ones it has collected, it can still recognize it. Let's first inspect *training.py* using idle by entering the following command*:*

idle training.py &

At the very bottom, line 65 states that we are going to save the machine learning model we will train in a folder called "trainer". Let's create a folder called "trainer" by entering the following command in the terminal:

mkdir trainer

Step 3.2:

Now we can try to run *training.py*:

python3 training.py

Most likely you will again encounter some errors. Please try to solve them by yourself. If you really want to give up, just check refer to the cheat sheet in the last section.

Step 3.3:

After you have solved that, try to run *training.py* again. If you see some numbers popping on the terminal screen, that means the training process has begun. It should terminate in a few seconds.

Step 4: Testing (face_recognition.py)

Step 4.1:

At this point, the machine should be able to associate your face with an ID, which is a number, but we also want the machine to know your name! So now let's introduce ourselves to the machine by modifying *face_recognition.py*:

idle face_ recognition.py &

In around line 46-53, you can modify the code so that the machine will know your name.

Step 4.2:

At this stage, you can try to run *face_recognition.py* by

python3 face_recognition.py

and do some debugging. Again, if you really want to give up, please check refer to the cheat sheet in the last section.

Step 4.3:

After you have solved that, try to run *face_ recognition.py* again.

Task1: *Submit to Moodle: a demonstration video & **face_recognition.py***

You can train your model so that it can recognize your classmates as well. The procedures would be similar to the above 3 steps.

Facial Recognition with OpenCV and Raspberry Pi Summary (Cheat Sheet)

Step 0: Install OpenCV https://www.pyimagesearch.com/2018/09/19/pip-install-opencv/

Step 1: Terminal ->

```
cd ~/Desktop/
git clone https://github.com/trieutuanvnu/Raspberry-Face-Recognition
cd Raspberry-Face-Recognition
```

Step 2.1: Terminal ->

```
mkdir dataset
idle face_datasets.py &
```

face_datasets.py ->

line 11 -> ***face_id = 1***

line 36 -> ***print(count)***

Step 2.2: Terminal ->

```
sudo modprobe bcm2835-v4l2
```

Note that it's v4l2 NOT v412. If this doesn't solve the problem, try:

Save all the python files you edited first

Terminal ->

```
sudo reboot
cd ~/Desktop/Raspberry-Face-Recognition
sudo modprobe bcm2835-v4l2
```

Step 2.3: Terminal ->

```
python3 face_datasets.py
```

Step 2.4: Clean the collected data in */home/pi/Raspberry-Face-Recognition/dataset*

Step 3.1: Terminal ->

```
mkdir trainer
idle training.py &
```

Step 3.2: Terminal ->

```
pip3 install Pillow
```

training.py ->

line 13 -> ***recognizer = cv2.face.LBPHFaceRecognizer_create()***

line 65 -> ***recognizer.write('trainer/trainer.yml')***

Step 3.3: Terminal ->

```
python3 training.py
```

Step 4.1: Terminal ->

```
idle face_ recognition.py &
```

face_recognition.py ->

line 47 -> **Id = "<your name>"**

Step 4.2: *face_recognition.py* ->

line 8 -> ***recognizer = cv2.face.LBPHFaceRecognizer_create()***

line 11 -> ***recognizer.read('trainer/trainer.yml')***

(for OpenCV 3.3: ***recognizer = recognizer.load('trainer/trainer.yml')*** *)*

line 46 -> ***if(Id[0] == 1):***

Step 4.3: Terminal ->

```
python3 face_recognition.py
```

Optional (no submission): Streaming Video from Pi Camera to Web Browsers

There are a few modern streaming protocols for web browsers. For example, **HLS** is Apple's choice, so it has great support on Apple devices but not much elsewhere. Another one, called **Fragmented MP4**, is supported by Adobe and Microsoft, but requires browser plugins from these companies on the player computer, so Windows and Mac computers can do it, but Linux and mobile cannot. Therefore, the alternative here we will be using is an older streaming protocol called **Motion JPEG** or **MJPEG**.

So what is **Motion JPEG**? In simpler terms, it's just a stream of individual JPEG pictures, one after another. The downside of MJPEG streams is that they are not as efficient as H.264, which greatly improves quality and reduces size by encoding only the differences from one frame to the next.
The following section guides you to install the **MJPG-Streamer** on the Raspberry Pi, which is a small open source MJPEG streaming server written in C that can be easily compiled for the Raspberry Pi.

1. Install build dependencies

The following command installs the three libraries that MJPG-Streamer uses:

```
sudo apt-get install libjpeg8-dev imagemagick libv4l-dev
```

2. Add missing videodev.h

The ***videodev.h*** header file that MJPG-Streamer needs has been replaced with a videodev2.h, therefore, you have to create a symbolic link with the command:

```
sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h
```

3. Download MJPG-Streamer

The source code for MJPG-Streamer is available at sourceforge.net, open a Terminal window and enter the following command:

```
wget http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-streamer-code-182.zip
```

*Note: sometimes the link above fails to work. If that is the case, please go to the following link to download the file:

http://sourceforge.net/p/mjpg-streamer/code/HEAD/tarball

4. Unzip the MJPG-Streamer source code

The source code downloaded is a compressed zip file. Put the file in your home directory (or a temporary folder, if you prefer) and run the following to extract the files:

```
sudo unzip mjpg-streamer-code-182.zip
```

5. Build MJPG-Streamer

MJPG-Streamer comes with several plugins, but only a couple of them are needed to stream video, enter the following commands in the Terminal window:

```
cd mjpg-streamer-code-182/mjpg-streamer
sudo make mjpg_streamer input_file.so output_http.so
```

6. Install MJPG-Streamer

The following commands copy all the needed files into system directories:

```
sudo cp mjpg_streamer /usr/local/bin
sudo cp output_http.so input_file.so /usr/local/lib/
sudo cp -R www /usr/local/www
```

7. Start the camera

We are almost there! Now it is time to start the camera module:

```
sudo mkdir /tmp/stream
raspistill --nopreview -w 640 -h 480 -q 5 -o /tmp/stream/pic.jpg -tl 100 -t 9999999 -th 0:0:0 &
```

If you encounter the error: **mmal: mmal_vc_component_enable: failed to enable component: ENOSPC**, please try to follow the instructions in this link below. note that if you are not familiar with **nano**, please try to use **leafpad** instead (replace **nano** with **leafpad** in the commands)**:**
http://www.linuxx.eu/2014/07/mmal-mmalvccomponentenable-failed-to.html

8. Start MJPG-Streamer

The camera is now writing images, the last step is to start MJPG-Streamer:

```
LD_LIBRARY_PATH=/usr/local/lib mjpg_streamer -i "input_file.so -f /tmp/stream -n pic.jpg" -o "output_http.so -w /usr/local/www"
```

9. Watch the Stream!

Now you can connect with your web browser and watch the stream live. If you want to watch from within the same Raspberry Pi, you can enter **http://localhost:8080** in the browser's address bar, and then click on the **stream** tab on the left-hand side. If you want to watch from another computer in your network, enter **http://<IP-address>:8080** instead of http://localhost:8080 on the other computers or your mobile phone's web browsers. Demonstrate your work to the TA.

Congratulations! That's the end of this Module! Have fun programming! ☺

References

1. https://projects.raspberrypi.org/en/projects/getting-started-with-picamera
2. https://blog.miguelgrinberg.com/post/how-to-build-and-run-mjpg-streamer-on-the-raspberry-pi
3. https://virtualenv.pypa.io/en/stable/
4. https://opencv.org/
5. https://github.com/trieutuanvnu/Raspberry-Face-Recognition