# **Logbook**

The purpose of this document is to keep track of the project on a weekly basis. Each week includes planned objectives, what was achieved and any comments regarding the work done during that week.

# Semester 1:

Semester 1 includes mainly the project research took place, materials required for the project, which project to choose and write the proposals for chosen project and Proposals were published on 25 November 2019 and the project was allocated along with supervisor on 06 December 2019.

First meeting took place on 16 December 2019 with supervisor to get to know supervisor. Supervisor gave some keys points to start Face Recognition project.

# Semester 2:

In semester 2 the project was started on 27 January 2020 right after the exams.

# Week 1 - 27 January 2020

**Objectives:**

* Aim of a 1st week was to watch videos on OpenCV and how to install & setup
* Watch online tutorials on how to work on Python programming language
* Get familiar with Python programming language.
* Install Anaconda on to windows operating system
* Research on what else this project requires

**Results:**

The OpenCV videos were watch on how to install OpenCV along with Python.

Anaconda was simple to install on the windows operating system and the download link for the installation was available on Anaconda’s original website. The link for the installation as follows: <https://www.anaconda.com/distribution/>.

The latest version of Python 3.7 (64-Bit) was installed on the computer with a size of 462 MB,



Figure 1 - Anaconda’s available versions

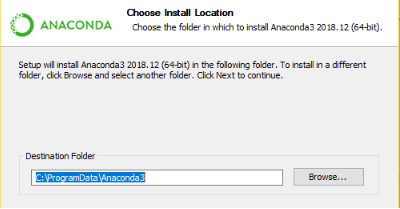


Figure 2 -Choosing location to install

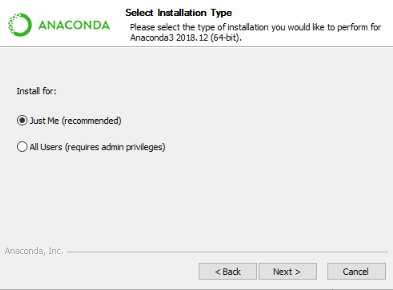


Figure 3 - Selecting installation Type

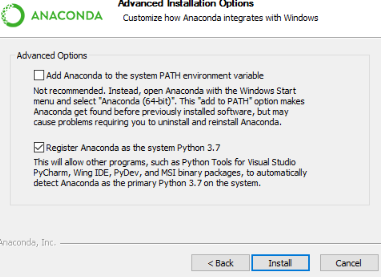


Figure 4 - Installing Anaconda

**Comments:**

The Anaconda was installed properly on the computer and had access to multiple components in the Anaconda’s Package list.

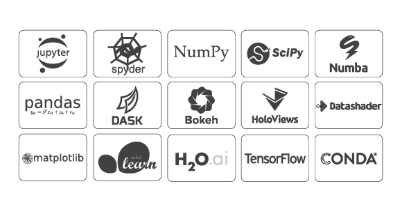


Figure 5 - List of Anaconda Packages

was to get familiar with the Python programming syntax. Spyder runs the latest version of Python so the Spyder was useful tool for improving syntax.

Anaconda Prompt was opened to check the right versions are installed. The commands for checking the Python version and Anaconda version are “python -V” & “Anaconda -V”.

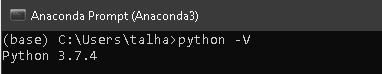


Figure 6 - Python Version is 3.7.4

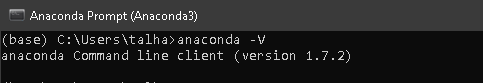


Figure 7 - Anaconda version is 1.7.2

# Week 2 - 03 February 2020

**Objectives:**

* Aim of this week was to install OpenCV library
* Run a program on Python and test the code
* Work on Face Detection

**Results:**

A new environment had to be created in Anaconda Navigator to install OpenCV packages with an older version of Python 3.6. Since the latest 3.7 version was not supporting. Then OpenCV had to be activated in the Anaconda Prompt, which was creating a lot of issues.

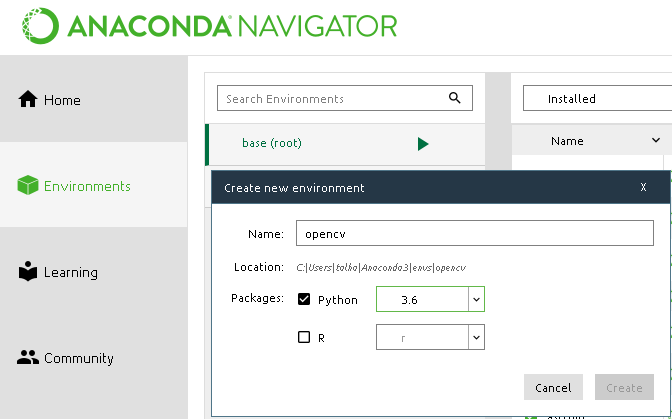


Figure 8 - New environment created in Anaconda Navigator

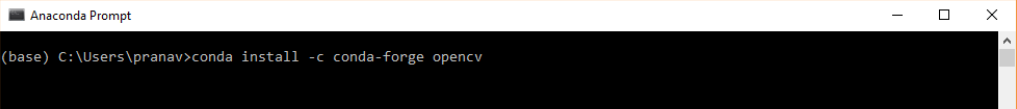


Figure 9 - Installing OpenCV through Anaconda Prompt

Figure 6 shows the code in the prompt for installing OpenCV. However, the program tries to solve the environment as shown in figure 7.

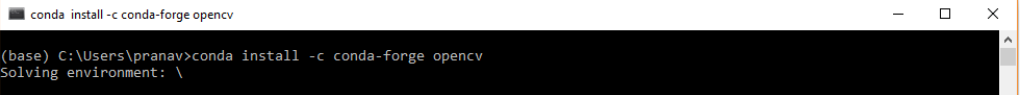


Figure 10 - Solving environment

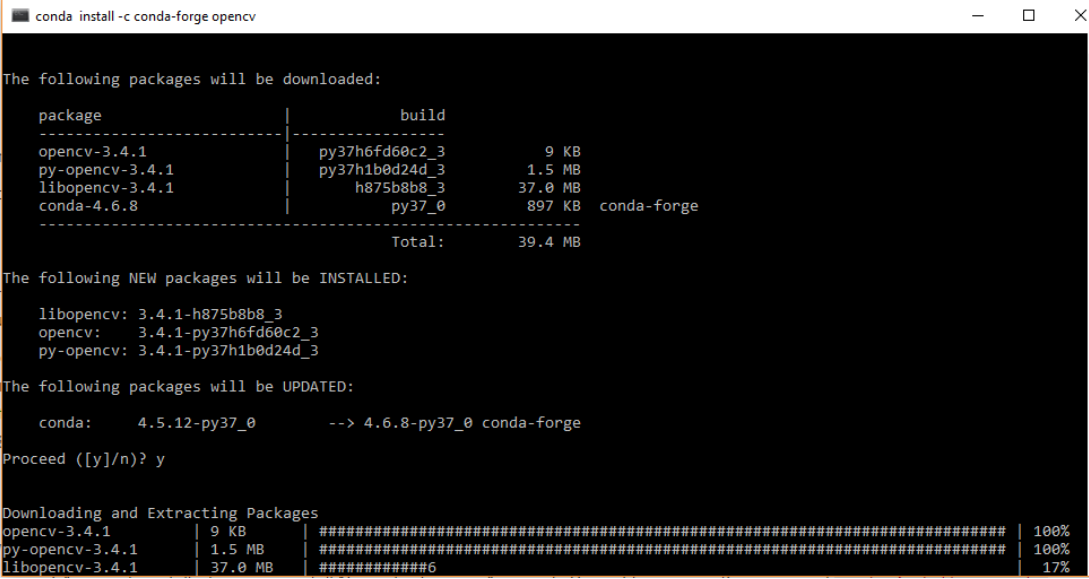


Figure 11 - Downloading and Extracting packages

Downloading packages runs forever and caused a computer to halt and there seems to be lot issues environment is trying to solve after, but it never solves instead it goes in more depth.

**Comments:**

The progress was made on installing OpenCV library, however the process took nearly two days to install the packages which eventually had to cancelled since it was getting nowhere. There are different ways were found to install OpenCV which were also using manual installation methods where the libraries are manually placated in the Anaconda’s directory.

The code couldn’t be run without the OpenCV.

# Week 3 - 10 February 2020

**Objectives:**

* Solving OpenCV installation issue
* Download Haar cascade
* Program a Face detection

**Results:**

OpenCV was a convenient way to detect a face using Python language via Anaconda Navigator and it was a most appropriate way to install OpenCV.

The Anaconda Navigator looked for the OpenCV when it was searched on the Anaconda Navigator search bar and all available OpenCV libraries option were shown and there were three main libraries available for OpenCV to install “libopencv”, “opencv” and “py-opencv” as shown in figure 12. The “opencv” library was installed and It didn’t take much time to install as compared to manually installed. The manual installation of OpenCV had flaws.

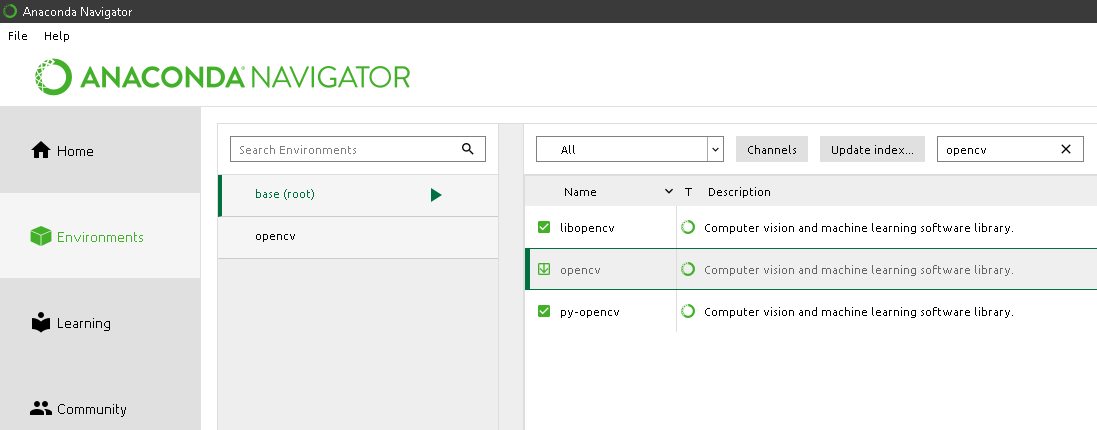


Figure 12 - Available OpenCV library to install in Anaconda Navigator

The OpenCV was imported in the Python console in ‘Spyder’ to check whether the “OpenCV” library was installed properly or not. Figure 11 shows the OpenCV was instated properly without an error.

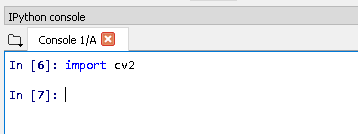


Figure 13 - Importing 'OpenCV' library to check, if it is installed

The 4th line of code defines the face cascade equal to the loading of that cascade “**cv2.CascadeClassifier ('haarcascade\_frontalface\_default.xml')”** and the frontal face haar cascade is downloaded onto the folder where the python files are located. The cascade files were available on the following GitHub link. <https://github.com/opencv/opencv/tree/master/data/haarcascades>

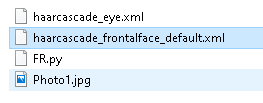


Figure 14 - The Cascades stored in the folder, where the python code is located

**Comments:**

The program for face detection was generated, however, it was not bale to work properly due an issue (0) in line 42, it should 1 representing colour image.

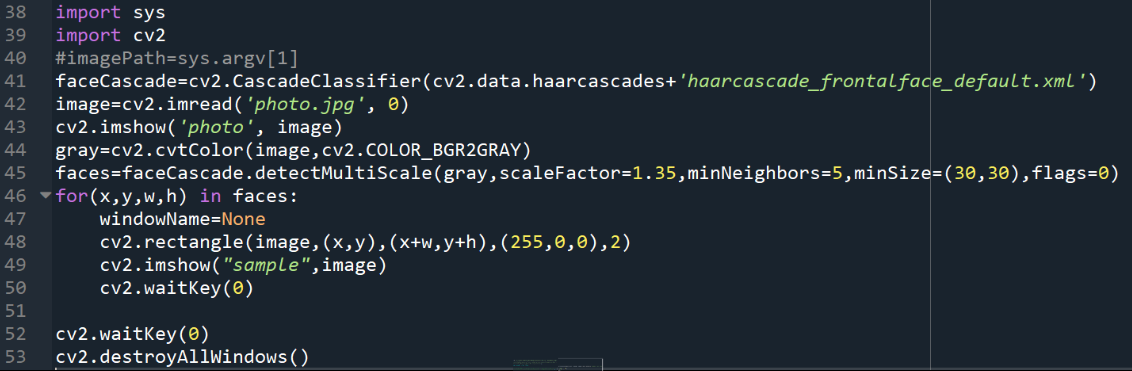


Figure 15 - Face Detection

The program above is not stable as it does not detect faces properly.

# Week 4 - 17 February 2020

**Objectives:**

* Resolve Face Detection program
* Learn the theory behind the Parameters
* Implement Live video capture face detection

**Results:**

The progress on the face Detection code is devolved which detects faces only. The code is running on the Spyder with the latest version Python 3.7.

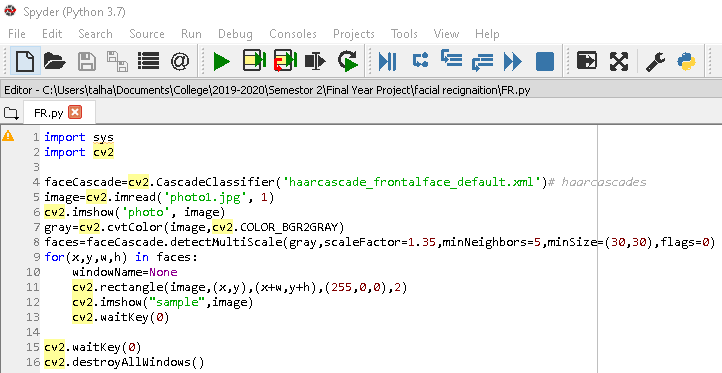


Figure 16 - Face Detection from provided image from the program

MultiScale functions:

* **‘gray’** is the input grayscale image shown in figure ().
* **‘scaleFactor’** is used for creating the scale pyramid and this parameter identifying how much the size of the image is reduced at each image scale.
* **‘minNeighbbors’** is a parameter identifies how many neighbours each applicant rectangle should have to keep it. A higher number provides lower false positives.
* **‘minSise’** is the minimum rectangle size to be considered a face.

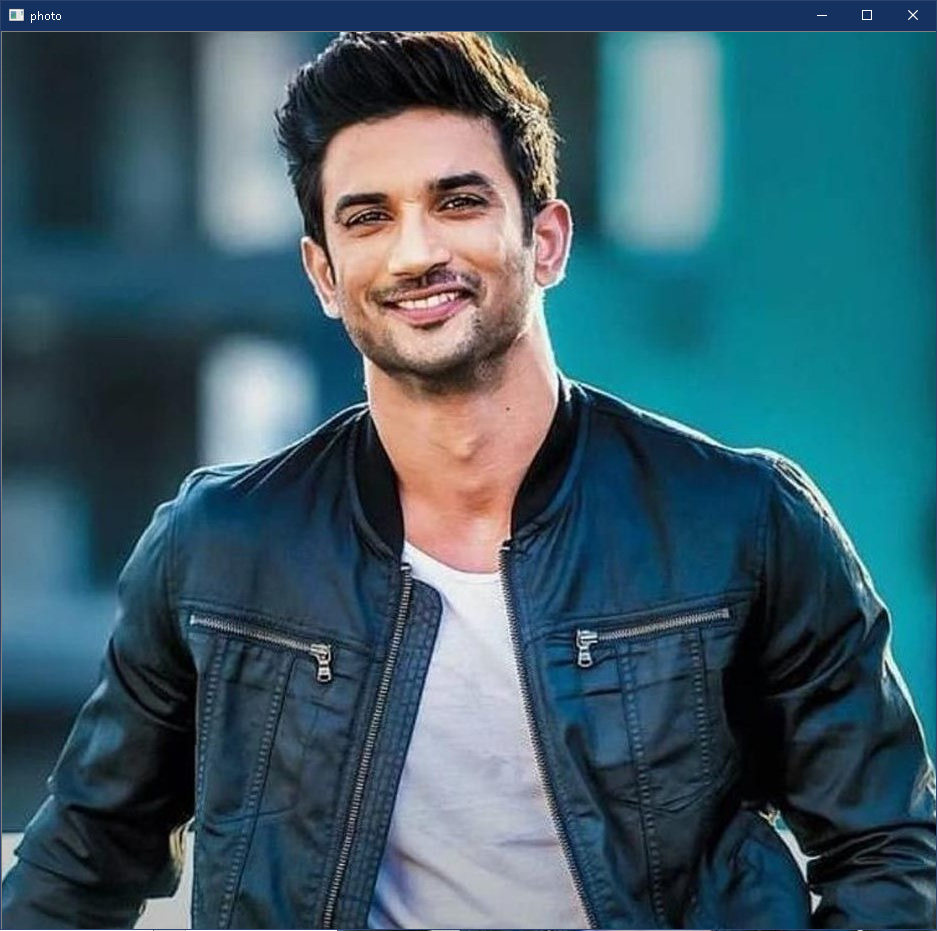


Figure 17 - Sample provided Photo



Figure 18 - Face Detection

**Comments:**

The camera was not working this week due to camera driver issues. Other than that, the program takes in a image and it detects the face in that image.

# Week 5 - 24 February 2020

**Objectives:**

* Implementing a code to detect a face from live video camera
* Learning the process of algorithm

**Results:**

The face detection from a captured video stream was performed in under 25 lines of python code and the result are remarkably shown below as it was detecting the face extremely well even though there are a lot of objects happen to be darker than other at the background.

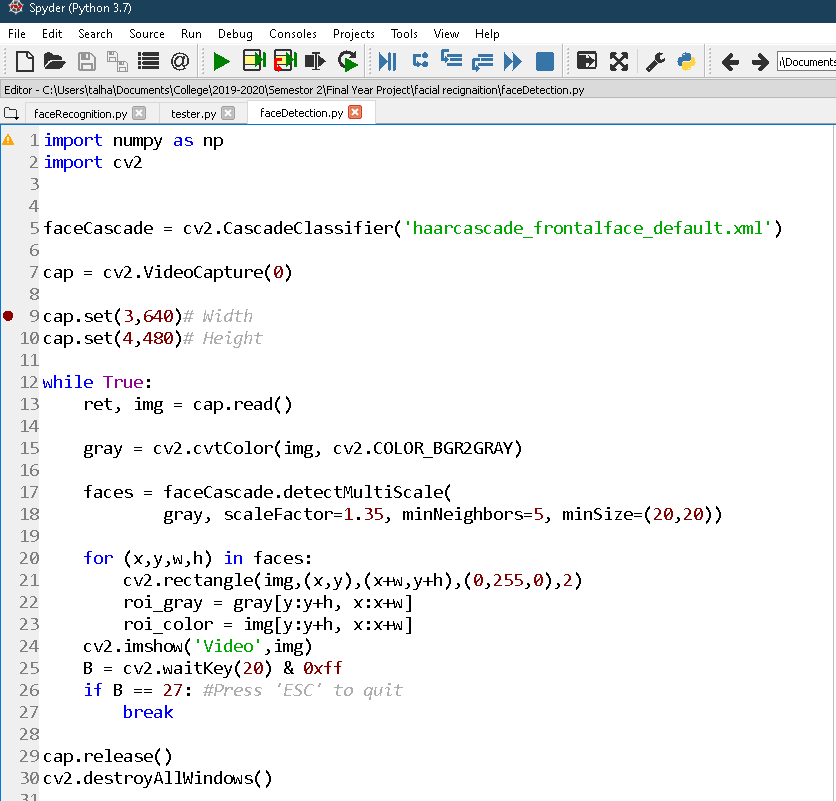


Figure 19 - Face Detection program code from live video camera

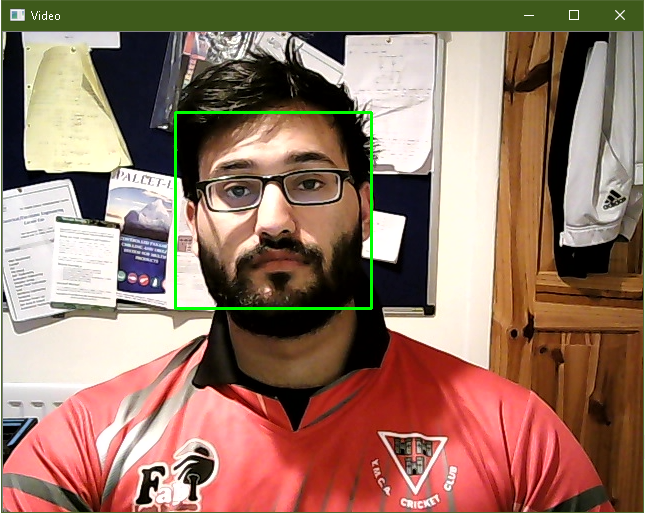


Figure 20 - live face detection from a camera

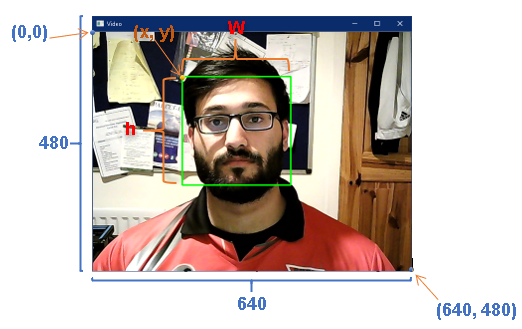


Figure 21 - Coordinates of each point when face to detected

**Comments:**

The problems with the face detection are

* Unknown size of the face - could be big or small
* Very high-resolution image
* Different ethnic groups – young and old people
* People with glasses on

Paul Viola & Michael Jones came up with a classifier that uses very simple features, one bit of an image subtracted by from another bit of an image in thousands and thousands of those features providing a clue after good few stages, maybe there is a face. Basically, it is making very quick decisions about what it is to be a face.

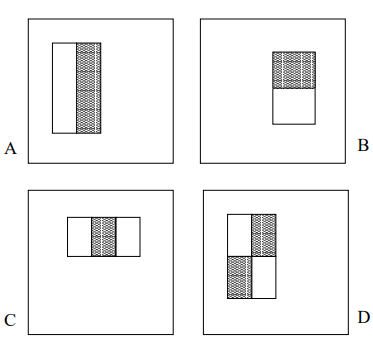


Figure 22 - Rectangle features

# Week 6 - 02 March 2020

**Objectives:**

* Create a directory which stores all the program and features
* Create a database for training images
* Create a test images directory

**Results:**

The directory was required for creating a database where all the training images were stored, testing images samples as an input for reading the images or capturing video from a camera and frontal face cascade. The program had to be stored in the same directory providing easy access to data base, beside with 2 python scripts that were created for the Face Recognition.

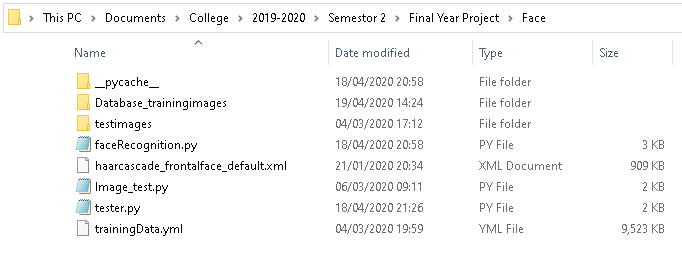


Figure 23 - Directory



Figure 24 - Gathering tanning images in database

100 training images were taken from the camera for user ID = 0 and 40 training images for user ID = 1 were taken from Google images for only testing purposes.

The ID 0 and ID 1 directory was created as shown in figure () which stores training images samples for recognizer.

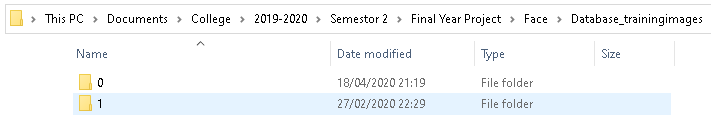


Figure 25 - "Database\_tranningimages" directory ID's for Training images in data base

**Comments:**

The test images are images for inputting an image in the program to be able to detect and recognize.

# Week 7 – 09 March 2020

**Objectives:**

* To be able to recognize faces from test images folder
* The block Diagram
* Explain the Principle of Face Recognition

**Results:**

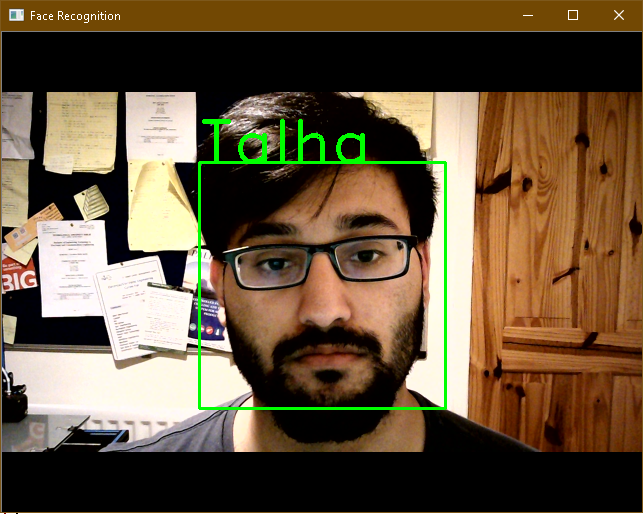


Figure 26 - Face Recognition

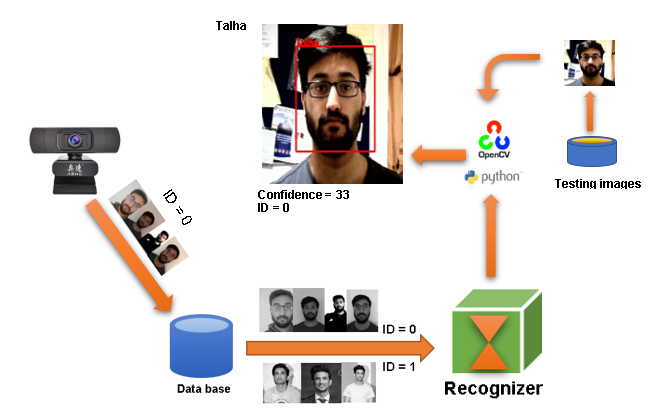


Figure 27 - Bock diagram of the face recognizer

**Comments:**

The program of this code is divided into 2 parts and the code is showing below.

# Week 8 - 16 March 2020

**Objectives:**

* To be able to recognize faces from test images folder
* The block Diagram
* Explain the Principle of Face Recognition

**Results:**

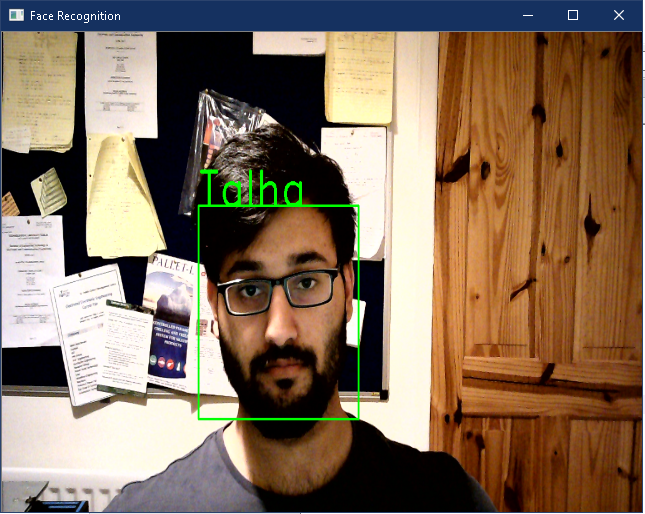


Figure 26 - Face Recognition

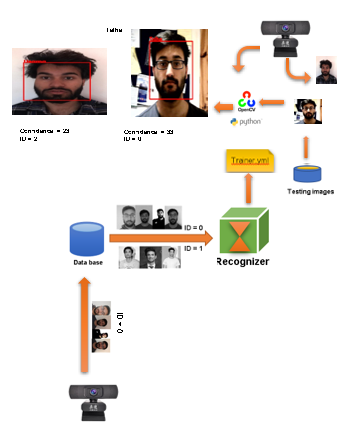


Figure 27 - Bock diagram of the face recognizer

**Comments:**

**The program:**

import os

import cv2

import numpy as np

import faceRecognition as fr

#This module captures images via webcam and performs face recognition

face\_recognizer = cv2.face.LBPHFaceRecognizer\_create()

face\_recognizer.read('trainingData.yml')#Load saved training data

name = {0 : "Talha",1 : "Kangana", 2 : "Usama"}

cap=cv2.VideoCapture(0)

cap.set(3,1000)# Width

cap.set(4,480)# Height

while True:

ret,test\_img=cap.read()# captures frame and returns boolean value and captured image

faces\_detected,gray\_img=fr.faceDetection(test\_img)

for (x,y,w,h) in faces\_detected:

cv2.rectangle(test\_img,(x,y),(x+w,y+h),(0,0,255),thickness=2)

resized\_img = cv2.resize(test\_img, (640, 480))

cv2.imshow('Face Detection',resized\_img)

cv2.waitKey(10)

for face in faces\_detected:

(x,y,w,h) = face

roi\_gray = gray\_img[y:y+w, x:x+h]

ID,confidence = face\_recognizer.predict(roi\_gray)#predicting the id of given image

print("Confidence: ",confidence)

print("ID: ",ID)

fr.draw\_rect(test\_img, face)

predicted\_name = name[ID]

#cv2.putText(img, str(confidence),(x+5,y+h-5), font, 1, (255,255,0),1)

#fr.put\_confidance(test\_img,confidence,x,y)

if confidence < 40:#If confidence less than 37 then don't print predicted face text on screen

fr.put\_text(test\_img,predicted\_name,x,y)

resized\_img = cv2.resize(test\_img, (640, 480))

cv2.imshow('Face Recognition',resized\_img)

if cv2.waitKey(10) == ord('b'):#wait until 'b' key is pressed

break

cap.release()

cv2.destroyAllWindows

### FaceRecognition.py

import cv2

import os

import numpy as np

def faceDetection(test\_img):

gray\_img=cv2.cvtColor(test\_img,cv2.COLOR\_BGR2GRAY)

face\_haar\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

faces=face\_haar\_cascade.detectMultiScale(gray\_img,scaleFactor=1.32,minNeighbors=5,minSize=(30,30),flags=0)

return faces,gray\_img

def ID\_for\_training\_database(directory):

faces=[]

faceID=[]

for path,subdirnames,filenames in os.walk(directory):

for filename in filenames:

id=os.path.basename(path)

img\_path = os.path.join(path, filename)

print("img\_path:",img\_path)

print("id:",id)

test\_img = cv2.imread(img\_path)

if test\_img is None:

print("Image not loaded properly")

continue

faces\_rect,gray\_img = faceDetection(test\_img)

if len(faces\_rect)!=1:

continue

(x,y,w,h)=faces\_rect[0]

roi\_gray=gray\_img[y:y+w,x:x+h]

faces.append(roi\_gray)

faceID.append(int(id))

return faces,faceID

def train\_classifier(faces,faceID):

face\_recognizer=cv2.face.LBPHFaceRecognizer\_create()

face\_recognizer.train(faces,np.array(faceID))

return face\_recognizer

def draw\_rect(test\_img,face):

(x,y,w,h)=face

cv2.rectangle(test\_img,(x,y),(x+w,y+h),(0,0,255),thickness=10)

def put\_text(test\_img,text,x,y):

cv2.putText(test\_img,text,(x,y),cv2.FONT\_HERSHEY\_DUPLEX,2,(0,0,255),thickness=10)