***Face Recognition***

At this one we are gonna be showing you how to implement open CV with Python to do facial recognition and facial identification. So what facial recognition is is really just pulling out faces in an image or a video , this technique that we're gonna show you works for both of them and since open CV is so powerful.

It's actually really easy to implement so like we're not actually gonna have to do a whole lot of work , there's some understanding on how to actually get it setup but once you do it's gonna be very useful and easy to get going to recognize faces.



Fig (1)

Now that's not the same as identifying faces identifying faces as what you see now that's a bounding box around face in **fig(1)** with the name on it up that actually shows that we can actually identify faces in a video or image as well using open CV this takes a little bit extra training that we have to do so at the end of this.

The recognizing or the identifying of a face those things can absolutely be

advanced but open CV has some built-in features to make these things a lot easier.

When I say advanced I mean by using like a deep learning libraries such as tensor-flow or PI torch those can advance our recognition feature for the identification portion now if all this stuff sounds a little confusing you might want to go back and learn a little bit more about open CV but if you know open CV and you have it .

*Now we are going to start with the first step which is installation :*

All resources being in this repository : <https://github.com/engmarioghaly/Face_Recogintion_Pyra.git>

At first you must have python , if your computer doesn't have python you should install it form : <https://www.python.org/downloads/> **DON’T FORGET TO MAKE THE PATH ON INSTALLATION**

Installing libraries :

1\_ open-CV using command : pip install opencv-python

2\_ Face-recognition : to install this library you should install dlib at first .

To install dlib you have to open the repository then download this file

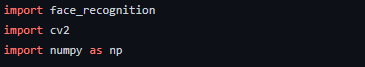
Then use a command : pip install “\*Path of this file on the computer “

Now you installed the dlib the you can use a command : pip install face-recognition it will install face-recognition and numpy libraries .

Note : Make sure that your pip version is the latest version .

Code :

At first we start to write the libraries that I’ll use in this application ….



We used face\_recognition library to load image file , compare faces and encode faces .

Cv2 is the open-cv library that used to capture video , recognize and positioning faces .

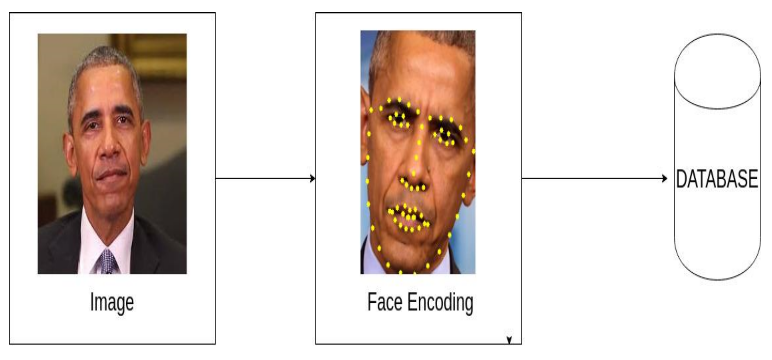
After this, The start of the code :

At first we have to load a sample picture to learn how to recognize it …..



This lines used to load image file then encode the image

Face encoding is used to specify some points in the face like eye , nose ,lips and the frame of face



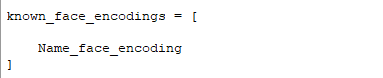
Then we have to start capturing using a camera …



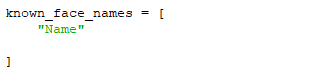
This line used to define a video capture object for the camera

NOTE : if you want to setup your mobile camera you can use a specified program to host the mobile camera’s live video to windows system … you can download this program form app store or google play it’s called **Droid** cam you can download it from : <https://www.dev47apps.com/>

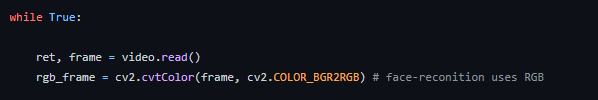
Now the camera seen some people that it recognize some of them and some of other people aren’t known to this camera , So we have to make a list with the encoded images to compare it with the video that captured by the camera .



So , you have also to make list with the name of the Name of the person on the image , this name will appear on the frame.

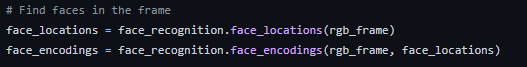


* Set up an infinite while loop and use the **read()**method to read the frames using the above created object.
* Then we use **cv2.cvtColor** to recognize face using RGB



* Now we want to positioning the face and make it inside a frame , so we use a command

**face\_recognition.face\_locations** to know the location of the face on the frame then we use face\_recognition.face\_encodings to encode the image inside the frame



* We will make a for loop to see if the face is match to known face or not , so we will compare the frame body of the encoded face and face location with the image .

If it matches it will output the user’s name … if it doesn’t match it will out “not in this class “

To compare the captured image we use **face\_recognition.compare\_faces ,** then , you have two ways to compare images …

First: is if the match true with we will make the name of the sample image on the image that captured by the camera , but it’s not accurate because a very small change on the appearance of the human , this code couldn't define him ,

So instead , we use the known face with the smallest distance to the new face that will give a variety of some changes on human appearance

* Then , When the device see a face ( known face or unknown face ) , we have to put the face in a frame.

So we have to use cv2.rectangle to draw frame around the face we used the 4 dimension and center the frame on the face of the user using this command : **(frame, (left, top), (right, bottom), (0, 0, 255), 2)**

* Now we want to draw a label with the name of the user below the frame , first we have to make a filled rectangle to type a name on it

**cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)**

* **To control colors we have to know the colors :**

**Red : (0,0,255)**

**Green : (0,255,0)**

**White : (255,255,255)**

So , if we want to to change the frame’s color we have to change ((frame, (left, top), (right, bottom)**, (0, 0, 255), 2) )**

Then we have to type with a font so you can specify any font you want using this command

**font = cv2.FONT\_HERSHEY\_DUPLEX**

* Then you can put text below the frame using command :

**cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)**

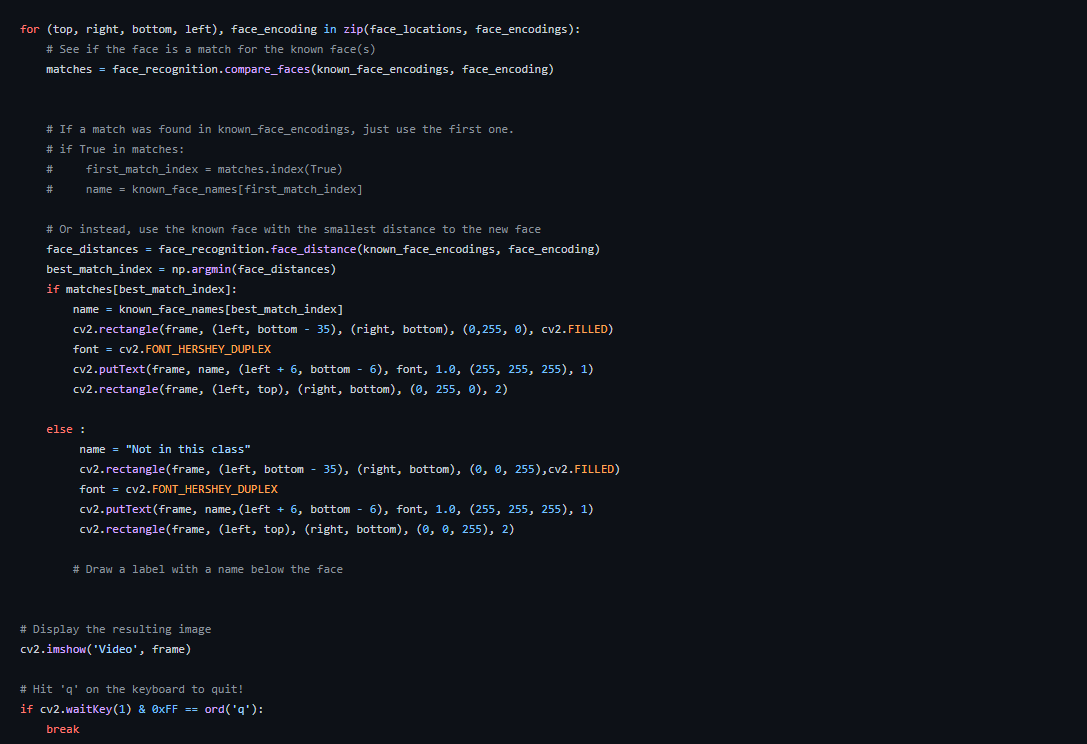
* Finally, we are going to display the final image result using command :

**cv2.imshow('Video', frame)**

* **Note :** if you want to hit ‘q’ to close the program you can use :

**if cv2.waitKey(1) & 0xFF == ord('q'):**

**break**



* **Finally ,** you have to handle to the camera to make the best capture you can use commands to handle the camera :

**video.release()**

**cv2.destroyAllWindows()**

