Face Recognition

The objective of the program given is to recognize object of interest(face) in real time. Face detection using haarcascades is a machine learning based approach where a cascade function is trained with a set of input data. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc.

There are three steps used for recognizing faces.

Data Gathering: Gather face data (face images in this case) of the persons you want to identify.
Train the Recognizer: Feed that face data and respective names of each face to the recognizer so that it can learn.
Recognition: Feed new faces of that people and see if the face recognizer you just trained recognizes them.

The Libraries we will be using are: -

- **cv2:** This is the OpenCV module for Python used for face detection and face recognition.
- **os:** We will use this Python module to read our training directories and file names.
- **numpy:** This module converts Python lists to numpy arrays as OpenCV face recognizer needs them for the face recognition process.
- **PIL :** This is the Pillow module for Python used for importing image

Data Gathering (face_image_storer.py)

1) Firstly, we initialize cascade which is just an XML file that contains the data to detect faces. We read the image and convert BGR frames to grayscale. (RGB to gray scale is done to reduce complexity as the image dimension change from 3 to single dimension).

2) To capture a user id, “input command” was added, that should be an integer number (1, 2, 3, etc).

3) The detectMultiScale function is a general function that detects objects. Since we are calling it on the face cascade, that’s what it detects.

- The first option is the grayscale image.
- The second is the scaleFactor. Since some faces may be closer to the camera, they would appear bigger than the faces in the back. The scale factor compensates for this.

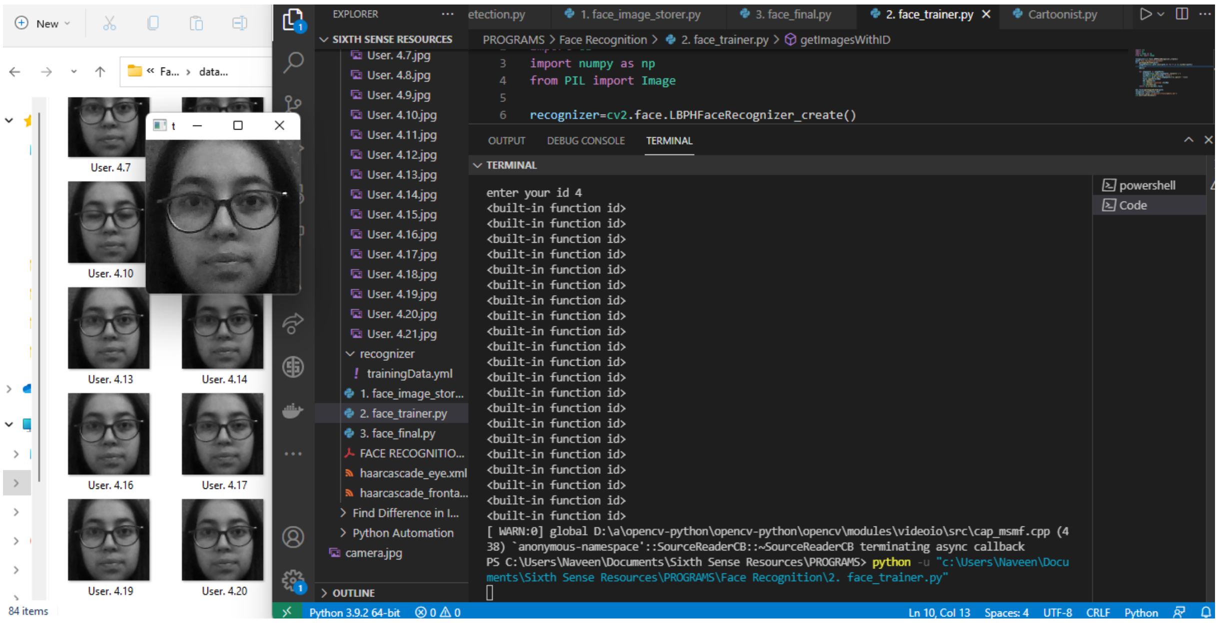
- The detection algorithm uses a moving window to detect objects. minNeighbors defines how many objects are detected near the current one before it declares the face found. minSize, meanwhile, gives the size of each window.

4) For each one of the captured frames, we should save it as a file on a “dataSet” directory. Each file’s name will follow the structure:

```
User.face_id.count.jpg
```

5) Run the Python script by pressing F5.

OUTPUT:



```python
import numpy as np
from PIL import Image

recognizer=cv2.face.LBPHFaceRecognizer_create()
```

```
enter your id 4
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
<built-in function id>
[ WARN:0] global D:\a\opencv-python\opencv-python\opencv\modules\videoio\src\cap_msmf.cpp (4
38) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
PS C:\Users\Naveen\Documents\Sixth Sense Resources\PROGRAMS> python -u "c:\Users\Naveen\Docu
ments\Sixth Sense Resources\PROGRAMS\Face Recognition\2. face_trainer.py"
```

Training(face_trainer.py)

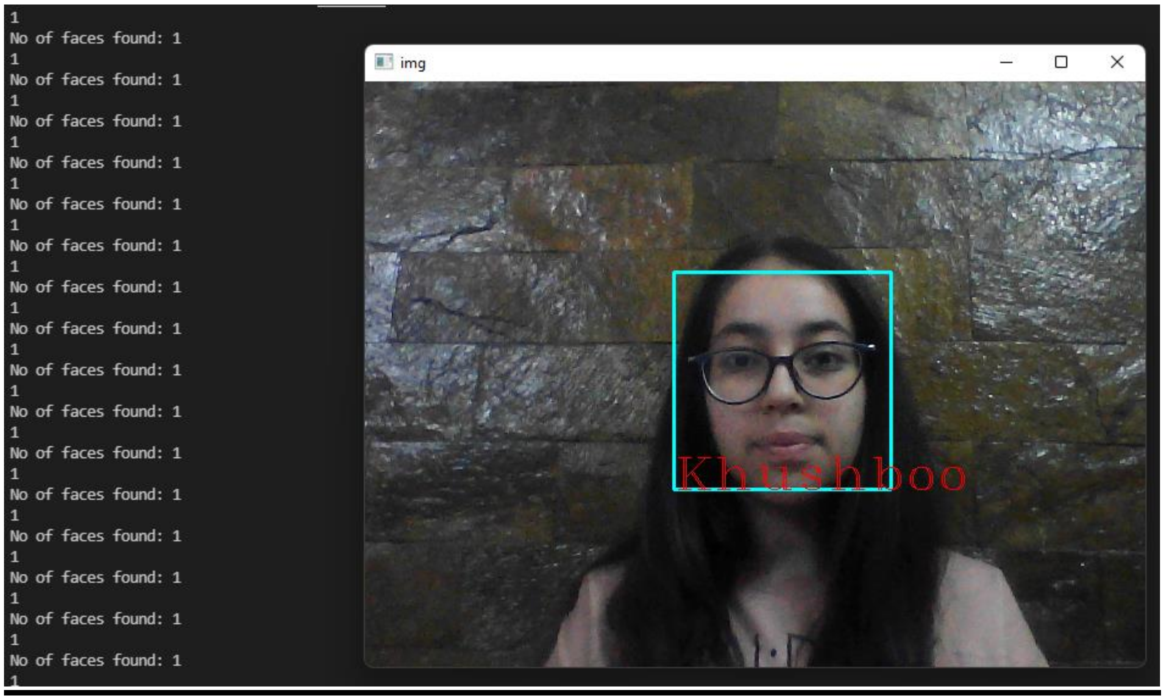
1) We have initialized our **LBPH** face recognizer and we also have prepared our training data in dataset folder, it's time to train. We will do that by calling the methodtrain(faces-vector, labels-vector) of face recognizer.
2) Instead of passing vector labels directly to face recognizer, we are first converting it to numpy array because OpenCV expects labels vector to be a numpyarray.
3) The function “*getImagesWithID (path)”,* will take all photos on directory: “dataset/”, returning 2 arrays: “Ids” and “faces”. With those arrays as input, we will “train our recognizer”.
4) As a result, a file named “*trainer.yml*” will be saved in the recognizer folder that was previously created by us.
5) Run the Python script by pressing F5.

Recognizition (face_final.py)

1) This is where we get to see if our algorithm is recognizing our individual faces or not.
2) The first function *draw_rectangle* draws a rectangle on the image based on given coordinates. It uses OpenCV's built in function cv2.rectangle(img, topLeftPoint, bottomRightPoint, rgbColor, lineWidth) to do so.
3) It basically read the test image , detect the face from test image and then recognize the face by calling face recognizer's predict(face) method. This method will return a label.
4) Finally we need to draw a rectangle around the detected face and draw the name of the predicted individual above the face rectangle.
5) Run the Python script by pressing F5.

Note :- all the haarcascade file should be in the same folder in which the code is saved.

Output



```
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
No of faces found: 1
1
```