

Term Project Report on

Face Recognition System

B. Tech. EC
Sem. VII

Submitted by

Name: Gokani Kesha
Roll No.:EC019
ID No.:18ECUOS106

Name: Sahil Katiriya
Roll No.:EC027
ID No.:18ECUOS112



DEPARTMENT OF ELECTRONICS AND COMMUNICATION
FACULTY OF TECHNOLOGY
DHARMSINH DESAI UNIVERSITY, NADIAD



Certificate

This is to certify that the project entitled “title of the project” is a bonafide work of Miss. Gokani Kesha Bhupendrabhai Roll. No.: EC019 Identity No.: 18ECUOS106 of B.Tech. Semester VII in the branch of “Electronics and Communication” during the academic year 2021-2022.

Staff In-Charge

Prof. Nisarg K. Bhatt

Head of the Department,

Dr. Purvang Dalal

Date: 27-11-2021

ACKNOWLEDGEMENT

It is great pleasure for me to express my gratitude to our honourable VICE CHANCELLOR **DR. H.M DESAI**, for giving the opportunity and platform with facilities in accomplishing the project-based laboratory report.

We express the sincere gratitude to our dean **PROF. D.G. PANCHAL** for his administration towards our academic growth.

We express sincere gratitude to HOD-EC **DR. PURVANG DALAL** for his leadership and constant motivation provided in successful completion of our academic semester. We record it as my privilege to deeply thank for providing us the efficient faculty and facilities to make our ideas into reality.

We express my sincere thanks to our project supervisor **PROF. NISARG BHATT** for his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to venture this project successfully.

We express our deep gratitude and affection to our parents who stood behind us in all our endeavours.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

INDEX

Sr. No	Title	Page No
1	Abstract	1
2	Introduction	2
3	Motivation	3
4	Problem Statement	3
5	Survey and Research	4
6	Methodology <ul style="list-style-type: none">➤ Need of an automated system➤ Graphic User Interface➤ Arrangement for system design	5
7	Face Detection from webcam <ul style="list-style-type: none">➤ Capture video from camera➤ Function Used➤ Compare two faces	7
8	Program Codes , Working and Results <ul style="list-style-type: none">➤ Program Code 1➤ Program Code 2➤ Program Code 3➤ Program Code 4	9
9	Trouble Shooting	20
10	Future Scope	21
11	Advantages and Disadvantages	22
12	Conclusion	23

ABSTRACT

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. There are two kinds of methods that are currently popular in developed face recognition pattern namely, Eigenface method and Fisherface method.

Facial image recognition Eigenface method is based on the reduction of face-dimensional space using Principal Component Analysis (PCA) for facial features. The main purpose of the use of PCA on face recognition using Eigen faces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project face detection system with face recognition is Image processing.

we've used machine learning to solve isolated problems that have only one step — estimating the price of a house, generating new data based on existing data and telling if an image contains a certain object. All of those problems can be solved by choosing one machine learning algorithm, feeding in data, and getting the result.

But face recognition is really a series of several related problems:

1. First, look at a picture and find all the faces in it
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc.
4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

This project gives an ideal way of detecting and recognizing human face using OpenCV, and python which is part of deep learning. This report contains the ways in which deep learning an important part of computer science field can be used to determine the face using several libraries in OpenCV along with python. This report will contain a proposed system which will help in the detecting the human face in real time. This implementation can be used at various platforms in machines and smartphones, and several software applications.

Key Words: Python, OpenCV, Deep Learning, Face detection, etc...

INTRODUCTION

Face recognition is the technique in which the identity of a human being can be identified using one's individual face. Such kind of systems can be used in photos, videos, or in real time machines. The objective of this report is to provide a simpler and easy method in machine technology. With the help of such a technology one can easily detect the face by the help of dataset in similar matching appearance of a person. The method in which with the help of python and OpenCV in deep learning is the most efficient way to detect the face of the person. This method is useful in many fields such as the military, for security, schools, colleges and universities, airlines, banking, online web applications, gaming etc. this system uses powerful python algorithm through which the detection and recognition of face is very easy and efficient.

Here, we are using the Python programming language for detecting faces in images and videos. It requires the installation of additional packages such as OpenCV. OpenCV stands for Open Source Computer Vision. In OpenCV, all the images are converted to or from numpy arrays. This makes it easier to integrate it with other libraries that use numpy.

After installing OpenCV, we need to install some specific libraries.

- Such as,
 - dlib
 - face_recognition
- We also need a working webcam. It includes some basic performance tweaks to make things to run faster.
 - only detect faces from every other frame.
- We will use Haar-cascade for face detection.
- For detecting the face from the video, we need a **VideoCapture** object. We will learn about this further.

MOTIVATION

The most useful area in which face recognition is important is the biometrics that is used for authentication process which makes the work more easier. Face recognition is one of the widely used technologies or systems in which it has the potential to perform tasks such as to have records provided in by the dataset in many areas such as the school and colleges attendance systems, it can also be helpful in catching the thieves or the terrorist, can be helpful in the security of common people and the much-needed security areas in the country. Face recognition can be used by the government to verify the voters list, find missing persons, find the population or census, immigration process, also provide security over internet scams protecting Ecommerce and highly used in the medicine and healthcare range. This brings in a very high demand or a real time face recognition system for several uses for the people and government. Providing such excellent systems there would be ease in several activities.

PROBLEM STATEMENT

The main aim or objective of this paper is to provide or develop a system that will use the camera of the computer or the system that would detect and recognize the person's face or the face of the individual using the tool in OpenCV called as the Open Face and python programming language in deep learning domain.

SURVEY & RESEARCH

This section is a basic overview of the major techniques used in the face recognition system that apply mostly to the front face of the human being. The methods include neural networks, hidden Markov model, face matching done geometrically and template matching.

Eigenface is one of the most widely used methods in face recognition and detection which are broadly called as the principle components in mathematical terms. The eigenvectors are ordered to represent different amounts of the variations in the faces.

Neural networks are highly used in the face recognition and detection systems. An ANN (artificial neural network) Was used in face recognition which contained a single layer Which shows adaptiveness in crucial face recognition systems. The face verification is done using a double layer of WISARD in neural networks.

Graph matching is other option for face recognition. The object as well as the face recognition can be formulated using graph matching performed by optimization of a matching function.

Hidden Markov Models is the way by which stochastic modeling of nonstationary vector time series based on HMM model applied to the human face recognition wherein the faces gets divided into parts such as the eyes, nose, ears, etc The face recognition and correct matching is 87% correct as it always gives out the best and right choice of face detection through stored dataset. Or else the relevant model reveals the identity of the face.

The geometrical feature matching is the technique which is based on the geometrical shapes of the face. The geometrical face configuration has sufficient dataset for face detection and recognition system. This is one of the commonly used method of the face recognition and detection. This system apparently gives satisfactory results.

Template matching is one of the techniques through which the test image is represented as a two- dimensional array of values which can be compared using Euclidean distance with single template representing the whole face. This method can also use more than one face template from different points of view to represent an individual face.

METHODOLOGY

The concept of OpenCV was put forth by Gary Bradski which had the ability to perform on multi-level framework. OpenCV has a number of significant abilities as well as utilities which appears from the outset. The OpenCV helps in recognizing the frontal face of the person and also creates XML documents for several areas such as the parts of the body.

Deep learning evolved lately in the process of the recognition systems. Hence deep learning along with the face recognition together work as the deep metric learning systems. In short deep learning in face detection and recognition will broadly work on two areas the first one being accepting the solidary input image or any other relevant picture and the second being giving the best outputs or the results of the image of the picture. We would be using dlib facial recognition framework that would be the easy way to organize the face evaluation. The two main significant libraries used in the system are dlib and face_recognition.

Python being a very powerful programming languages and one of the programming languages that are being used all over the world has proven to give best results in the face recognition and detection systems. Together face recognition and detection becomes very easy and fruitful with the help of the python programming language and OpenCV.

Need Of An Automated System:

Due to the rising need for the systems which can help in the areas such as surveillance as well as security this kind of individual authentication can no longer be done using simple handmade methods hence there is a rising need of the automated systems that can easily rectify the faults and process the human face recognition. When the work is done by machines it can perform tasks efficiently in very less duration of time and cuts off the major mistakes occurred by humans. A real time GUI based face recognition system built can ease this work of face detection and can be achieved in various ways.

Graphical User Interface:

The graphical user interface (GUI) is the platform that will allow the inputs from the user ends a kind of interaction with the system. GUI's are used in mobiles, media players, games and many others. We can design visual composition and the temporal behaviour of the GUI in any of the software application as well as programming in the areas of the human computer interaction. The GUI for this project will be widely based on the training and the testing phase which in turn will allow the capture and train of the image.

The minimum requirements for the software would be python along with OpenCV and the required dataset. The minimum requirements for the hardware would be intel i3 or any processor above it and 4 core CPU. Operating systems of windows 10 will be sufficient and random access memory 8GB required. From the user end a computer or laptop active internet connection and a scanner optional.

Arrangement For System Design:

In order to create this system first we will have to make the datasets. When the image quality becomes favourable different procedures will take place in the face recognition system the tasks are performed using the python queries “python encode_faces.py”. The input will be taken from the dataset which will be received in the “encodings.py”. There will be precision formatting in the system wherein face embedding for each face will occur. Secondly a file “recognize_faces_images.py” will contain all the required methods and the techniques for the process of identification of the face of the person from the given image of the dataset.

The given file will be executed by the python command “python recognize_faces_image.py-encodings”. We can resize or turn the image for approximity with the goal for getting the desired output. The present classifier along with OpenCV libraries will enhance the outcome or results in the face recognition system.

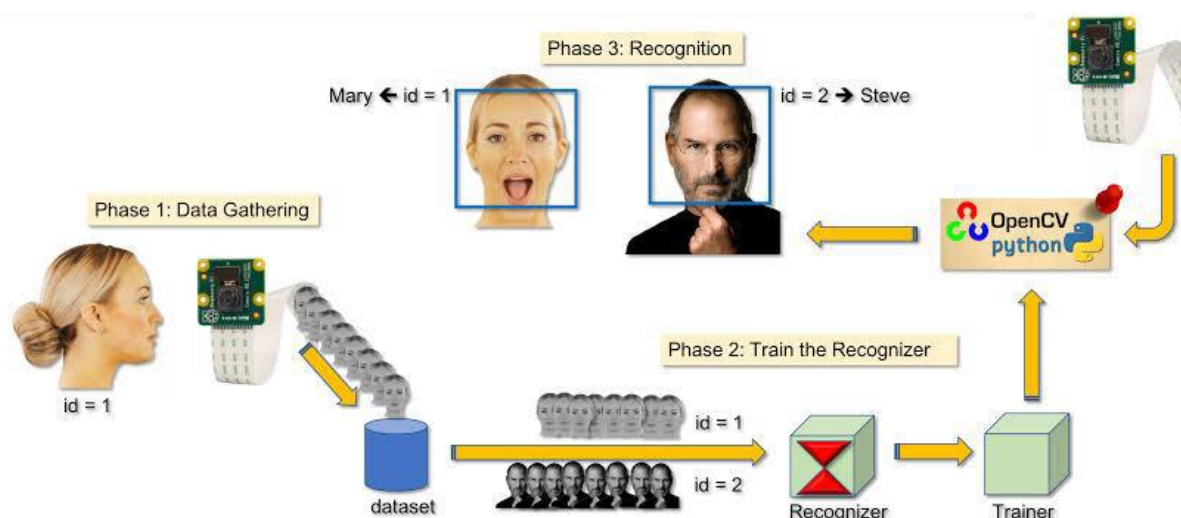


Figure 1: face recognition system design using python and OpenCV.

FACE RECOGNITION FROM WEBCAM

Capture Video from Camera:

Often, we have to capture live stream with camera. OpenCV provides a very simple interface to this. Let's capture a video from the camera (we are using the in-built webcam of my laptop), convert it into grayscale video and display it. Just a simple task to get started.

Firstly, we use `face_recognition` library.

`face_recognition` library in Python can perform a large number of tasks:

- Find all the faces in a given image
- Find and manipulate facial features in an image
- Identify faces in images
- Real-time face recognition

Here, we will talk about the 3rd use case – identify faces in images or videos.

To capture a video, we need to create a **VideoCapture** object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected (as in our case). So, I simply pass 0 (or -1). We can select the second camera by passing 1 and so on. After that, we can capture frame-by-frame. But at the end, don't forget to release the capture. After this, we have to load the sample picture and learn how to recognize it.

Then we have to grab the single frame from the video, resize it and convert the image from BGR color to RGB color. Find all the faces and face encoding in the current frame of video. Then we have to compare the face. So we can say that is unknown face or known face. And we can use the known face with the smallest distance to the new face. And then we use **cv2.rectangle** and **cv2.putText** to draw a box around the face and a label with a name below the face. Then it will show the result.

Function (Modules) used:

cv2. VideoCapture(): It is used to get a **video capture** object for the camera. Which is helpful to capture videos through webcam.

cv2.resize(): resizing an image means changing the dimensions of it, be it width alone, height alone or both.

cv2.rectangle(): it is used to draw a rectangle on any image.

cv2.putText(): It is used to draw a text string on any image.

cv2.imshow(): It is used to display an image in a window. the window automatically fits to image size.

cv2.destroyAllWindows(): simply destroys all the windows we created.

cv2.waitKey(): It waits for specified milliseconds for any keyword even.

face_recognition.face_encoding():given an image, return the 128 dimension face encoding for each face in the image.

face_recognition.face_location():returns an 2d array of bounding boxes of human faces in a image using the cnn face detector.

face_recognition.face_distance():given a list of face encodings, compare them to a known face encoding and get a Euclidean distance for each comparison face. The distance tell us how similar the faces are.

haarcascade_frontalface_default.xml : Detects faces.

haarcascade_eye.xml : Detects the left and right eyes on the face

haarcascade_smile.xml: Detects the simle

COMPARE TWO FACES:

We start by importing the packages we'll need face_recognition for recognizing the face, NumPy for numerical processing, and cv2 for our OpenCV bindings.

Then we locate the image at some fix address loaded it and then encoded it. And then we have to define the function for compamring the faces. And then we use cv2.rectangle and cv2.putText to draw a box around the face and a label with a name below the face. Then it will show the result.

PROGRAM CODE, WORKING AND RESULTS:

Program Code 1: Compare two faces

- We have done face recognition using dlib and opencv. Here, after giving 2 images -> 2 images are compare -> are identical or not -> display the results.

```
import cv2
import numpy as np
import face_recognition
#Step1: loading the images and coverting it into RGB because we r getting the image
as BGR but the library understand it as RGB
imgElon = face_recognition.load_image_file('D:\EC\SEM 7\TP - TERM
PROJECT\Code\elon.jpg')
imgElon = cv2.cvtColor(imgElon,cv2.COLOR_BGR2RGB)
imgTest = face_recognition.load_image_file('D:\EC\SEM 7\TP - TERM
PROJECT\Code\elontest.jpg')
imgTest = cv2.cvtColor(imgTest,cv2.COLOR_BGR2RGB)
#Step2: finding the face in our image and then finding their encoding as well
faceLoc = face_recognition.face_locations(imgElon)[0] #detect the face , as we are
sending single image so we will get only first element [0]
encodeElon = face_recognition.face_encodings(imgElon)[0]
cv2.rectangle(imgElon,
(faceLoc[3],faceLoc[0]),(faceLoc[1],faceLoc[2]),(255,0,255),2) #(edge of the image ,
color, thickness)
#print(faceLoc)
faceLocTest = face_recognition.face_locations(imgTest)[0]
encodeTest = face_recognition.face_encodings(imgTest)[0]
cv2.rectangle(imgTest,
(faceLocTest[3],faceLocTest[0]),(faceLocTest[1],faceLocTest[2]),(255,0,255),2)
#Step3 : comparing these faces and finding the distance between them
results = face_recognition.compare_faces([encodeElon],encodeTest) #([give list],
compare with img)
faceDis = face_recognition.face_distance([encodeElon], encodeTest)
print(results,faceDis)
cv2.putText(imgTest,f'{results}
{round(faceDis[0],2)}',(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,0,255),2)
cv2.imshow('elon musk',imgElon)
cv2.imshow('elon musk test ',imgTest)
cv2.waitKey(0)
```

Working of code 1:

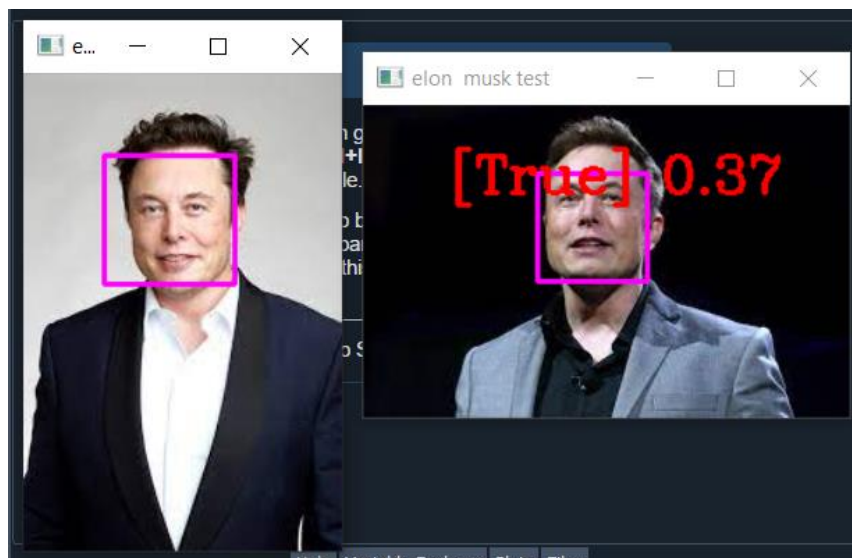
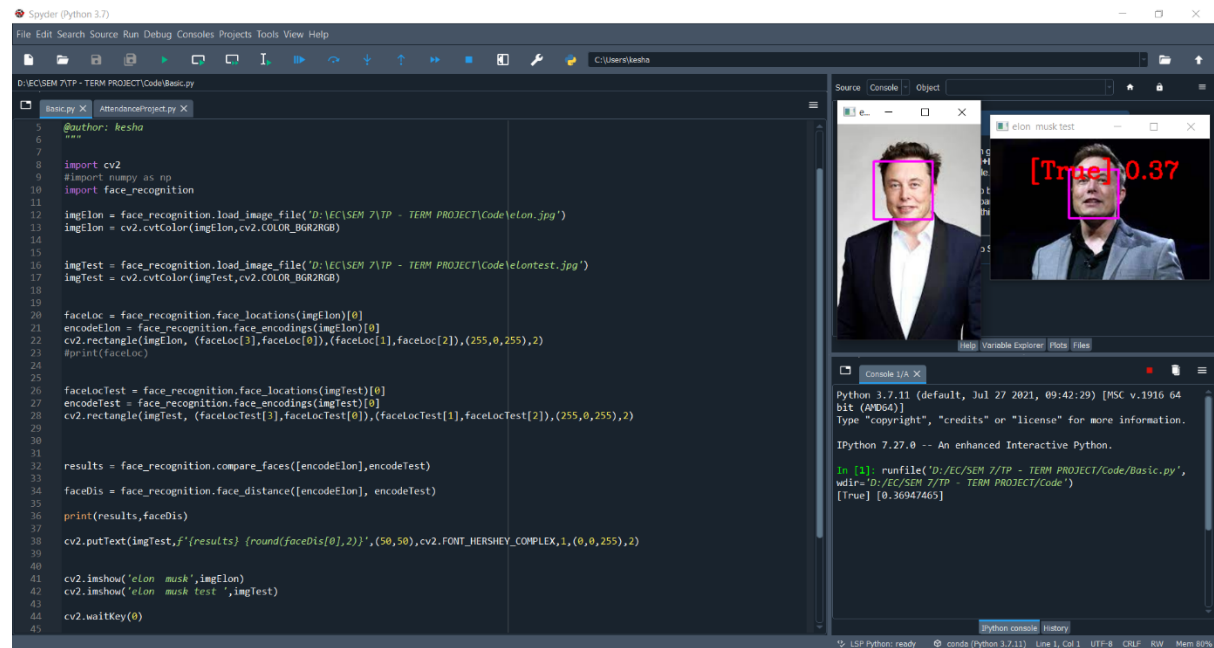
2 images are taken then

->Step1: loading the images and coverting it into RGB because we r getting the image as BGR but the library understand it as RGB (face_recognition.load_image = loading the image , cv2.COLOR_BGR2RGB = Convert BGR to RGB image)

-> Step2: finding the face in our image and then finding their encoding as well
(face_recognition.face_locations = finding the image , face_recognition.face_encodings = encoding the image)

->Step3 : comparing these faces and finding the distance between them
(face_recognition.compare_faces = comparing the images , face_recognition.face_distance = finding distance between them , face distance is called inter-pupillary distance / Typical commercial face recognition engines require face images with at least 60 pixels between the eyes for successful recognition)

RESULT:



Program Code 2: Face recognition using Webcam

- We have done face recognition using dlib and opencv. Here, after collecting data then webcam detect the correct face and identify the person.

```

import cv2
import numpy as np
import face_recognition
import os
#from datetime import datetime
#create a list that will get the images from our folder automatically and then it will
generate the encoding for it automatically and then it will try to find it in our webcam
path = 'D:\EC\SEM 7\TP - TERM PROJECT\Images Attendance'
images = [] #create a list of all the images then we will import
classNames = [] #write names of all these images
myList = os.listdir(path) #grab the list of images in this folder
print(myList)
# cl is class
for cl in myList:
    curImg = cv2.imread(f'{path}/{cl}')
    images.append(curImg) #going to append
    classNames.append(os.path.splitext(cl)[0])
print(classNames)
#create a simple function that will compute all the encodings
def findEncodings(images):
    encodeList = [] #create empty list
    for img in images:
        img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
encodeListKnown = findEncodings(images) #find encodings -> function (number of
encoding)
#print(len(encodeListKnown))
print('Encoding Complete')
#find the matches between our encodings
cap = cv2.VideoCapture(0)
while True:
    success, img = cap.read()
    imgS = cv2.resize(img,(0,0),None,0.25,0.25)
    imgS = cv2.cvtColor(imgS,cv2.COLOR_BGR2RGB)
    facesCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame = face_recognition.face_encodings(imgS,facesCurFrame)
    for encodeFace,faceLoc in zip(encodeCurFrame,facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown,encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown,encodeFace)
        #print(faceDis)
        matchIndex = np.argmin(faceDis)
        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            #print(name)
            y1,x2,y2,x1 = faceLoc
            y1,x2,y2,x1 = y1*4,x2*4,y2*4,x1*4
            cv2.rectangle(img,(x1,y1),(x2,y2),(0,255,0),2)

```

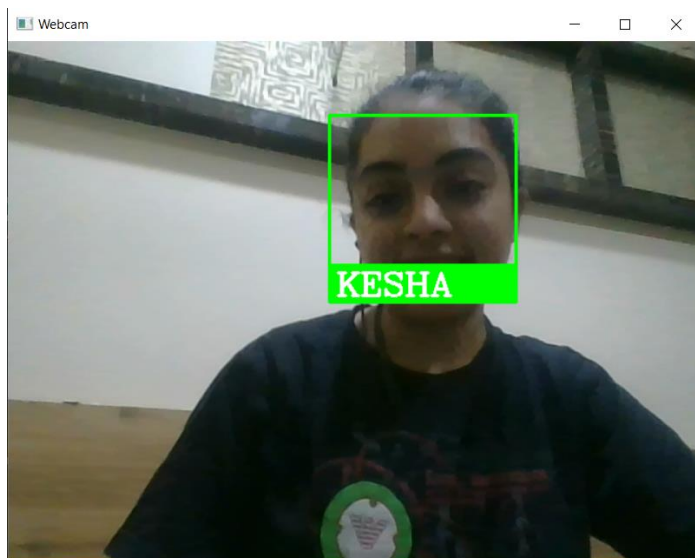
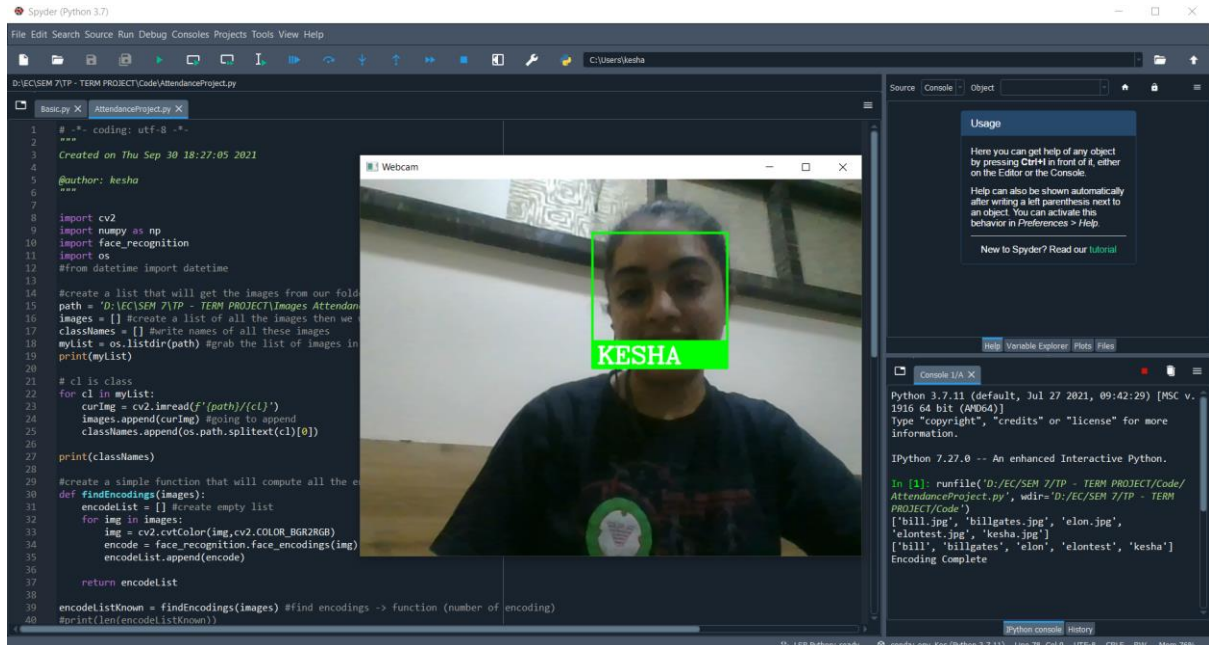


```

cv2.rectangle(img,(x1,y2-35),(x2,y2),(0,255,0),cv2.FILLED)
cv2.putText(img,name,(x1+6,y2-
6),cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),2)
cv2.imshow('Webcam',img)
cv2.waitKey(1)

```

RESULT:



Working of code 2:

Webcam identify the person's name, Here Code begins with storing a data of the person in a database (class) -> encoding is done ; Find encodings -> rectangle frame for the face detected

Program Code 3:

- We have done face recognition using Opencv. Here, First the samples are taken of a person using webcam and store in a database then afterwards samples are compare with the face detected on the webcam.

```
import cv2
import numpy as numpy
import lbph as lr
import sys, os,time
import sys
import msvcrt
while True:
    user=input('1.train 2.recognize : ')
    if user == '1':
        # create_database.py
        #password=raw_input('Enter Admin Password : ')
        password=input('Enter Admin Password (the password is password): ')
        if password == 'Password':
            count = 0
            size = 4
            fn_haar = 'haarcascade_frontalface_default.xml'
            fn_dir = 'database'
            fn_name = input('Enter Name Of The Person : ') #name of the person
            path = os.path.join(fn_dir, fn_name)
            if not os.path.isdir(path):
                os.mkdir(path)
            (im_width, im_height) = (68, 68)
            haar_cascade = cv2.CascadeClassifier(fn_haar)
            webcam = cv2.VideoCapture(0)
            print("-----Taking pictures-----")
            print("-----Give some expressions-----")
            # The program loops until it has 20 images of the face.
            while count < 64:
                (rval, im) = webcam.read()
                im = cv2.flip(im, 1, 0)
                gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
                mini = cv2.resize(gray, ((int)(gray.shape[1] / size), (int)(gray.shape[0] /
size)))

                faces = haar_cascade.detectMultiScale(mini)
                faces = sorted(faces, key=lambda x: x[3])
                if faces:
                    face_i = faces[0]
                    (x, y, w, h) = [v * size for v in face_i]
                    face = gray[y:y + h, x:x + w]
                    face_resize = cv2.resize(face, (im_width, im_height))
                    pin=sorted([int(n[:n.find('.')]) for n in os.listdir(path)
```

```

        if n[0]!='.' ]+[0]][-1] + 1
        cv2.imwrite('%s/%s.png' % (path, pin), face_resize)
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
        cv2.putText(im, fn_name, (x - 10, y - 10), cv2.FONT_HERSHEY_PLAIN,
            1,(0, 255, 0))
        time.sleep(0.38)
        count += 1
        cv2.imshow('OpenCV', im)
        key = cv2.waitKey(10)
        if key == 27:
            break
    print(str(count) + " images taken and saved to " + fn_name + " folder in
database ")

    cv2.destroyAllWindows()
    webcam.release()

import sys, os,time
size = 4
fn_haar = 'haarcascade_frontalface_default.xml'
fn_dir = 'database'

print('Training...')
# Create a list of images and a list of corresponding names
(images, lables, names, id) = ([], [], {}, 0)
for (subdirs, dirs, files) in os.walk(fn_dir):
    for subdir in dirs:
        names[id] = subdir
        subjectpath = os.path.join(fn_dir, subdir)
        for filename in os.listdir(subjectpath):
            path = subjectpath + '/' + filename
            lable = id
            images.append(cv2.imread(path, 0))
            lables.append(int(lable))
            id += 1
(im_width, im_height) = (68, 68)

# Create a Numpy array from the two lists above
(images, lables) = [numpy.array(lis) for lis in [images, lables]]
trained_face_recognizer=lr.train_lbph(images)
print('done')
numpy.save('trainedRec.npy',trained_face_recognizer)
else:
    print('Wrong Password! Authentication Failed! Try again!')
elif user=='2':
    trained_face_recognizer=numpy.load('trainedRec.npy')
    fn_dir = 'database'
    # Load prebuilt model for Frontal Face
    cascadePath = "haarcascade_frontalface_default.xml"
    (im_width, im_height) = (68, 68)
    # Part 2: Use fisherRecognizer on camera stream

```

```

(images, lables, names, id) = ([], [], {}, 0)
for (subdirs, dirs, files) in os.walk(fn_dir):
    for subdir in dirs:
        names[id] = subdir
        subjectpath = os.path.join(fn_dir, subdir)
        for filename in os.listdir(subjectpath):
            path = subjectpath + '/' + filename
            lable = id
            images.append(cv2.imread(path, 0))
            lables.append(int(lable))
        id += 1
face_cascade = cv2.CascadeClassifier(cascadePath)
webcam = cv2.VideoCapture(0)
while True:
    (_, im) = webcam.read()
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(im,(x,y),(x+w,y+h),(255,0,0),2)
        face = gray[y:y + h, x:x + w]
        face_resize = cv2.resize(face, (im_width, im_height))
        # Try to recognize the face
        #inputimg = numpy.zeros((1, 4096))
        #inputimg[0, :] = images[48, :]
        #lr.predict_logistic_regression(inputimg, lables, trained_face_recognizer)
        prediction=lr.predict_lbph(face_resize,trained_face_recognizer,lables)
        #prediction = recognizer.predict(face_resize)
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
        if (prediction[1])<100:
            #cv2.putText(im,'%s - %.0f' % (names[prediction[0]],prediction[1]),(x-
10, y-10), cv2.FONT_HERSHEY_PLAIN,1,(0, 255, 0))
            cv2.putText(im,'recognized - %.0f' % (prediction[1]),(x-10, y-10),
cv2.FONT_HERSHEY_PLAIN,1,(0, 255, 0))
        else:
            cv2.putText(im,'not recognized',(x-10, y-10),
cv2.FONT_HERSHEY_PLAIN,1,(0, 255, 0))

        cv2.imshow('OpenCV', im)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cv2.destroyAllWindows()
    webcam.release()
elif user=='3':
    break;
else:
    print('Enter a valid input')

```

RESULT:

```
1 import cv2
2 import numpy as numpy
3 import lbph as lb
4 import sys, os, time
5 import sys
6 import msvcrt
7
8
9 while True:
10     user=input('1.train 2.recognize : ')
11
12     if user == '1':
13         # create_database.py
14         #password=raw_input('Enter Admin Password : ')
15         password=input('Enter Admin Password (the password is password): ')
16
17         if password == 'Password':
18             count = 0
19             size = 4
20             fn_haar = 'haarcascade_frontalface_default.xml'
21             fn_dir = 'database'
22             fn_name = input('Enter Name Of The Person : ') #name of the person
23             path = os.path.join(fn_dir, fn_name)
24             if not os.path.isdir(path):
25                 os.mkdir(path)
26                 (im_width, im_height) = (68, 68)
27                 haar_cascade = cv2.CascadeClassifier(fn_haar)
28                 webcam = cv2.VideoCapture(0)
29
30
31             print("-----Taking pictures-----")
32             print("-----Give some expressions-----")
33             # The program loops until it has 20 images of the face.
34
```

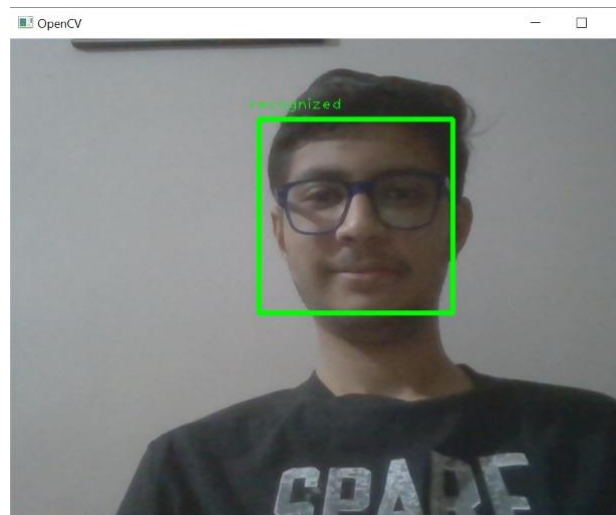
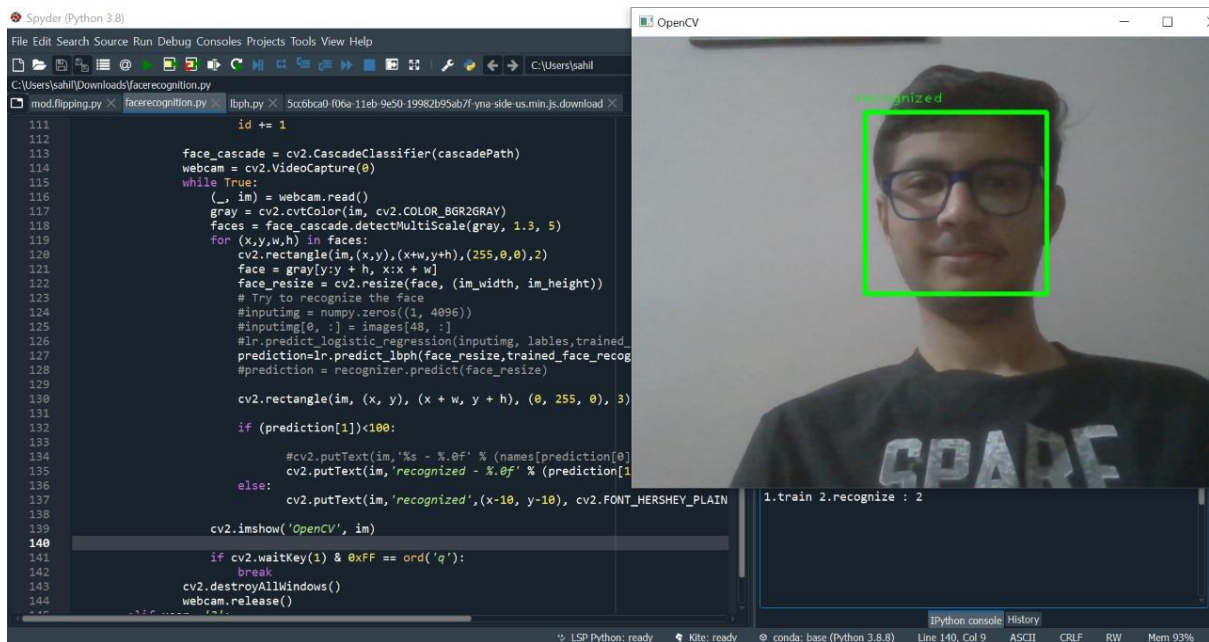
Console 1/A

```
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license()" for more information.
IPython 7.22.0 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/sahil/Downloads/facerecognition.py',
wdir='C:/Users/sahil/Downloads')
1.train 2.recognize : 1
Enter Admin Password (the password is password): Password
Enter Name Of The Person : Sahil
-----Taking pictures-----
-----Give some expressions-----
```

```
1 import cv2
2 import numpy as numpy
3 import lbph as lb
4 import sys, os, time
5 import sys
6 import msvcrt
7
8
9 while True:
10     user=input('1.train 2.recognize : ')
11
12     if user == '1':
13         # create_database.py
14         #password=raw_input('Enter Admin Password : ')
15         password=input('Enter Admin Password (the password is password): ')
16
17         if password == 'Password':
18             count = 0
19             size = 4
20             fn_haar = 'haarcascade_frontalface_default.xml'
21             fn_dir = 'database'
22             fn_name = input('Enter Name Of The Person : ') #name of the person
23             path = os.path.join(fn_dir, fn_name)
24             if not os.path.isdir(path):
25                 os.mkdir(path)
26                 (im_width, im_height) = (68, 68)
27                 haar_cascade = cv2.CascadeClassifier(fn_haar)
28                 webcam = cv2.VideoCapture(0)
29
30
31             print("-----Taking pictures-----")
32             print("-----Give some expressions-----")
33             # The program loops until it has 20 images of the face.
34
```

Console 1/A

```
1.train 2.recognize : 1
Enter Admin Password (the password is password): Password
Enter Name Of The Person : Sahil
-----Taking pictures-----
-----Give some expressions-----
64 images taken and saved to Sahil folder in database
Training...
[ WARN:1] global C:/Users/runneradmin/AppData/Local/Temp/pip-req-build-q3d_8t8e/opencv/modules/videoio/src/cap_msmf.cpp (438) "anonymous-namespace":SourceReaderCB::~SourceReaderCB terminating async callback
done
1.train 2.recognize :
```



Program Code 4:

- We have done face recognition using Opencv and haarcascade models (haarcascade_frontalface_default.xml , haarcascade_eye.xml ,haarcascade_smile.xml). Here , Webcam identify the total number of faces in a frame and so identify the simple , face and eye of the detected face.

import cv2

cascPath = "haarcascade_frontalface_default.xml"

eyePath = "haarcascade_eye.xml"

smilePath = "haarcascade_smile.xml"

```

faceCascade = cv2.CascadeClassifier(cascPath)
eyeCascade = cv2.CascadeClassifier(eyePath)
smileCascade = cv2.CascadeClassifier(smilePath)
font = cv2.FONT_HERSHEY_SIMPLEX
video_capture = cv2.VideoCapture(0)
while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces =
faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(200,
200),flags=cv2.CASCADE_SCALE_IMAGE)

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 3)

        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]

        cv2.putText(frame,'Face',(x, y), font, 2,(255,0,0),5)

        smile = smileCascade.detectMultiScale(roi_gray,scaleFactor=
1.16,minNeighbors=35,minSize=(25, 25),flags=cv2.CASCADE_SCALE_IMAGE)

        for (sx, sy, sw, sh) in smile:
            cv2.rectangle(roi_color, (sh, sy), (sx+sw, sy+sh), (255, 0, 0), 2)

            cv2.putText(frame,'Smile',(x + sx,y + sy), 1, 1, (0, 255, 0), 1)

        eyes = eyeCascade.detectMultiScale(roi_gray)

        for (ex,ey,ew,eh) in eyes:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

            cv2.putText(frame,'Eye',(x + ex,y + ey), 1, 1, (0, 255, 0), 1)

        cv2.putText(frame,'Number of Faces : ' + str(len(faces)),(40, 40), font, 1,(255,0,0),2)

    # Display the resulting frame

```



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

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

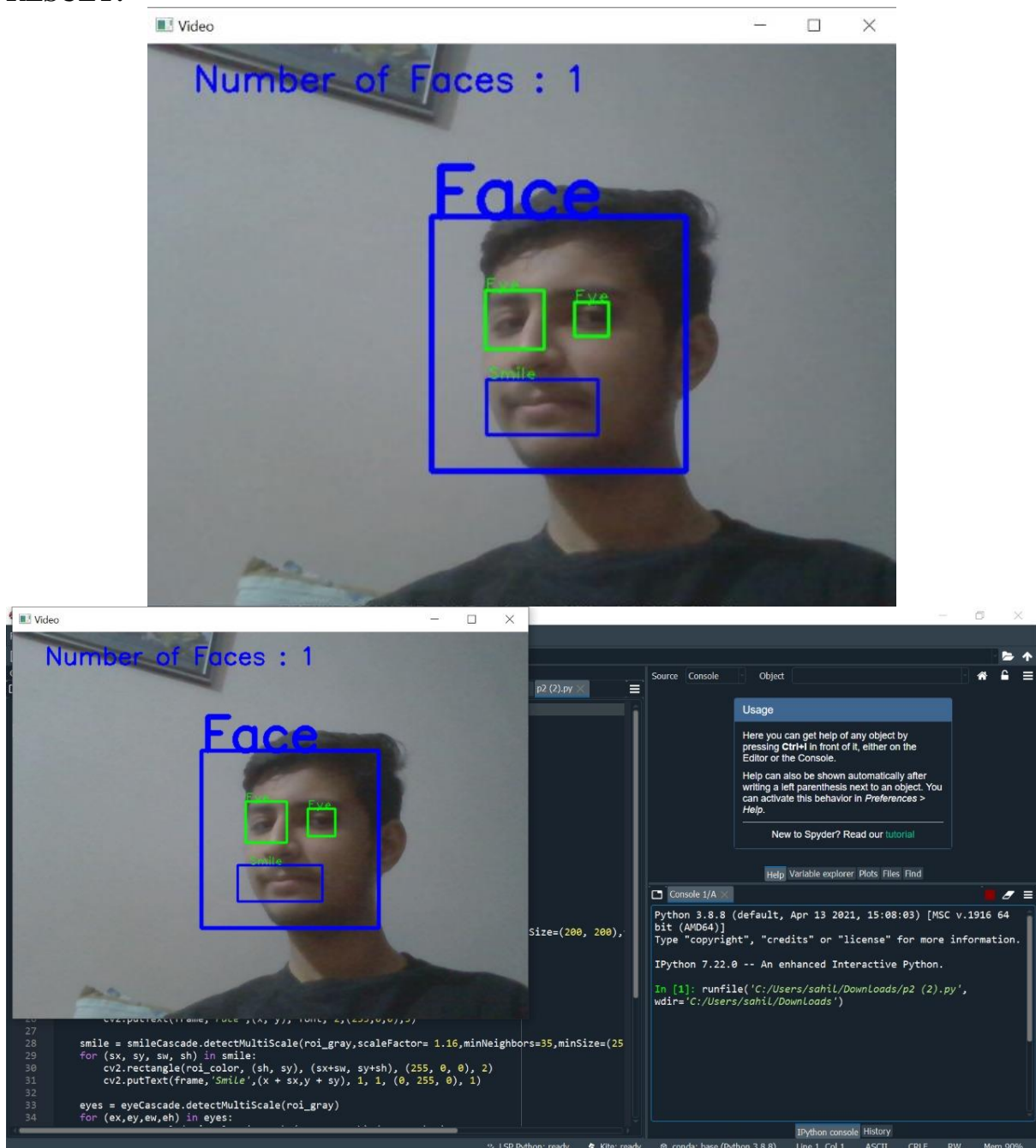
```
    break
```

```
# When everything is done, release the capture
```

```
video_capture.release()
```

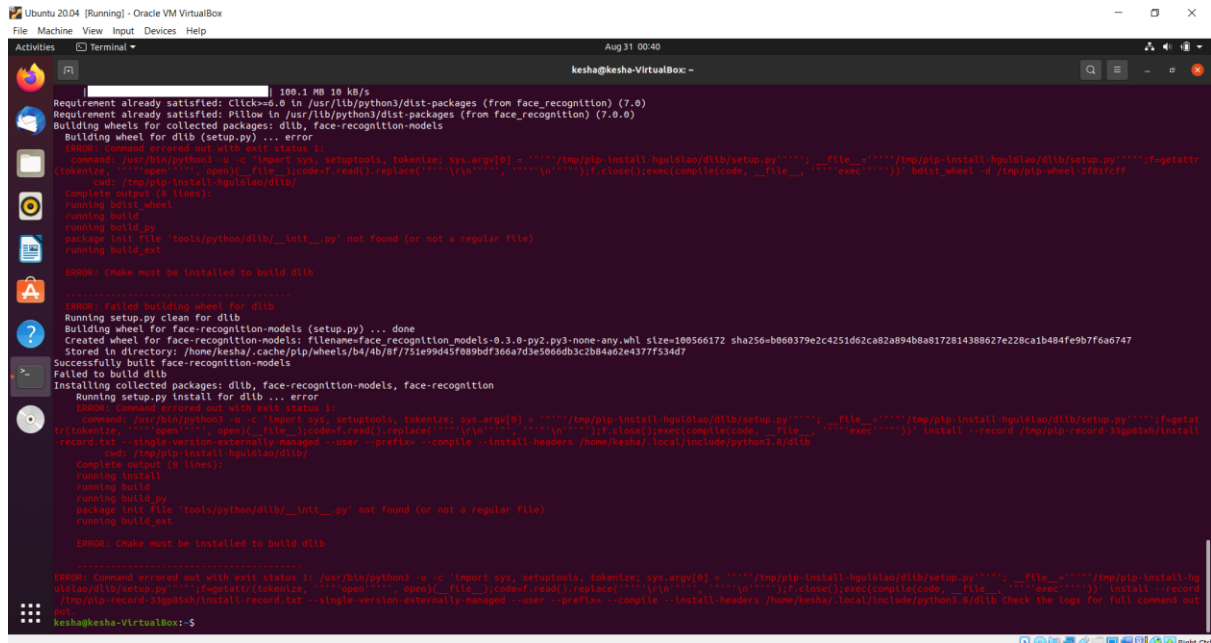
```
cv2.destroyAllWindows()
```

RESULT:



TROUBLE SHOOTING

Where not able to install dlib library in windows as well as Linux, then doing research on it at last we were able to install dlib library.



```
Ubuntu 20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Activities Terminal Aug 31 00:40 keshag@keshag-VirtualBox: ~

Requirement already satisfied: click==8.0 in /usr/lib/python3/dist-packages (from face_recognition) (7.0)
Requirement already satisfied: pillow in /usr/lib/python3/dist-packages (from face_recognition) (7.0.0)
Building wheels for collected packages: dlib, face_recognition-models
Building wheel for dlib (setup.py) ... error
Command: /usr/bin/python3 -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'/tmp/pip-install-hguldaw/dlib/setup.py'"'; __file__ = '"'/tmp/pip-install-hguldaw/dlib/setup.py'"'; f=getattr(tokenize, '"__open__"', open)(__file__); code=f.read().replace('"'\n'"', '"'\n'"'); f.close(); exec(compile(code, __file__, '"__exec__"'))' bdist_wheel -d /tmp/pip-wheel-zf8fcff
cwd: /tmp/pip-install-hguldaw/dlib/
Complete output (8 lines):
running bdist_wheel
running build
running build_py
package init file 'tools/python/dlib/__init__.py' not found (or not a regular file)
running build_ext

ERROR: CMake must be installed to build dlib

Command errored out with exit status 1:
  command: /usr/bin/python3 -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'/tmp/pip-install-hguldaw/dlib/setup.py'"'; __file__ = '"'/tmp/pip-install-hguldaw/dlib/setup.py'"'; f=getattr(tokenize, '"__open__"', open)(__file__); code=f.read().replace('"'\n'"', '"'\n'"'); f.close(); exec(compile(code, __file__, '"__exec__"'))' install --record /tmp/pip-record-39g85sh/install-record.txt --single-version-externally-managed --user --prefix --compile --install-headers /home/keshag/.local/include/python3.8/dlib
  cwd: /tmp/pip-install-hguldaw/dlib/
  Complete output (8 lines):
  running install
  running build
  running build_py
  package init file 'tools/python/dlib/__init__.py' not found (or not a regular file)
  running build_ext

  ERROR: CMake must be installed to build dlib

Command errored out with exit status 1:
  command: /usr/bin/python3 -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'/tmp/pip-install-hguldaw/dlib/setup.py'"'; __file__ = '"'/tmp/pip-install-hguldaw/dlib/setup.py'"'; f=getattr(tokenize, '"__open__"', open)(__file__); code=f.read().replace('"'\n'"', '"'\n'"'); f.close(); exec(compile(code, __file__, '"__exec__"'))' install --record /tmp/pip-record-39g85sh/install-record.txt --single-version-externally-managed --user --prefix --compile --install-headers /home/keshag/.local/include/python3.8/dlib
  cwd: /tmp/pip-install-hguldaw/dlib/
  Complete output (8 lines):
  running install
  running build
  running build_py
  package init file 'tools/python/dlib/__init__.py' not found (or not a regular file)
  running build_ext

  ERROR: CMake must be installed to build dlib

keshag@keshag-VirtualBox:~$
```


FUTURE SCOPE

- Governments across the world are increasingly investing their resources in facial recognition technology. Specially the US and China are the leaders in the facial recognition market.
- The technology is expected to grow and will create massive revenues in the coming years. Surveillance and security are the major industries that will be intensely influenced by technology.
- Schools and universities and even healthcare are also planning to implement the facial recognition technology on their premises for better management.
Complicated technology used in facial technology is also making its way to the robotics industry.

ADVANTAGES AND DISADVANTAGES

The advantages of the face recognition system include faster processing, automation of the identity, breach of privacy, massive data storage, best results, enhanced security, real time face recognition of students in schools and colleges, employees at corporate offices, smartphone unlock and many more in day-to-day life.

Few disadvantages in this system include the costing, or the funding, very good cameras of high definition are required, poor image quality may limit the effectiveness of this system, size of the image will matter because it becomes difficult to recognize the face in small images, Face angles can limit the face recognition reliability, massive storage is required for this system to work effectively.

CONCLUSION

Face recognition systems are currently associated with many top technological companies and industries making the work of face recognition easier. Face recognition technology has come a long way in the last twenty years. In this project, we learned about some interesting library of OpenCV. We have done to detect the images through the webcam and also from the video. We can recognise three faces at a time. We have tried to recognise more faces at a time. And we go away from the webcam then it cannot detect the face properly. And also, we have learned to compare the two faces. This system is very important and useful system now a days. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

The use of python programming and OpenCV makes it an easier and handy tool or system which can be made by anyone according to their requirement. The proposed system discussed in this project will be helpful for many as it is user friendly and cost_efficient system. Hence by the use of python and OpenCV the face recognition system can be designed for various purposes.