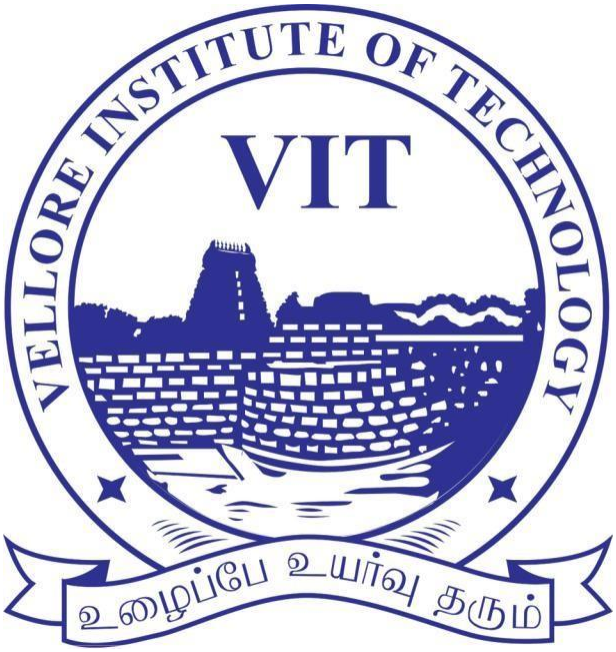


Registration Number	Name
19BCT0015	**Ch.Tharun Tej**

FACE RECOGNITION ALGORITHM

Abstract:

Face recognition is one the most trending technologies which is used everywhere in the present world. It involves the scanned part of our body parts which are unique to everyone and are stored in a database this makes hackers impossible to steal passwords or any personal information.

It has a lot many advantages in this present world where it helps a lot of scientists, political parties to protect their private party matters from stealing by the other parties or the hackers. Face recognition involves a lot of coding and it even requires mathematical knowledge to retrieve one's information.

Mathematics topics like eigen vectors and integration is used to retrieve data, since these eigen values and integration take the lengths of the body parts. computer interaction. Now that face recognition makes hackers impossible to steal someone's passwords and hack their belongings that have to be safe guarded. Face recognition is useful for people like- journalists, political parties to ensure their data is safe and to protect their data from stealing by other political parties or by the stealers. Our topic is face recognition which one of the most used technology in our daily lives. It involves scanning of face and get the points on the face which is unique for everyone and it get the points using mathematics topics like eigen vectors and integration. It also involves python language and matlab language. Matlab is used to get the mathematical values and python is general coding language which is used in biometrics and it is also used in artificial intelligence and machine learning. Python has inbuilt libraries which we make use of them and get the face values. All these faces are stored in databases and whenever we keep a live face it recognizes the face of the person which is already present in the database. Even face recognition is used in the data security and high security labs where only limited people have access to control over the labs. So face recognition is even used there to ensure high security to the companies.

Almost all the software companies, educational institutions have these face recognition technology to ensure the fraud or make their more comfortable. And now in our project we look this complex process will happen in simpler terms.

Introduction:

Now a days face recognition is growing rapidly. So many people are researching on it. We need to have some extra knowledge to know about it. We should know about the face highlights and geometric invariants. It depends on the face revolution and distortion. We even have some advantages too and disadvantages but one of the most peculiar disadvantage is there is less solid invariants.

Face detection involves the 2 step procedure: 1 containing the faces. It is hard to find the faces so the first step involved in face recognition technology is to get or capture the face. There are different problems while capturing the face such as light, background, quality and many more. So we need to have an ideal face detector which detect any face at any point of time. After getting the input using face detector we can get the output in 2 ways one way is keeping all the images in the folder as an input and when we scan the face then it will say if the given image is present or not, if the given image is present then it will give the output yes or else it will give the output no. Next method when we give the faces as the input then it will check and measure all the algebraic terms such as width, height, and color and then it will recognize the given input face to provide an easier human-machine interaction routine when user authentication is needed through face detection and recognition. With the aid of a regular web camera, a machine is able to detect and recognize a person's face; a custom login screen with the ability to filter user access based on the users' facial features will be developed.The objectives of this thesis are to provide a set of detection algorithms that can be later packaged in an easily-portable framework amongst the different processor architectures we see in machines (computers) today. These algorithms must provide at least a 95% successful recognition rate, out of which less than 3% of the detected faces are false positives.

Motivation

Face biometrics are useful fora person's authentication, which is a simple and non-intrusive method. Face recognition have substantial potential in two areas: a) it can help the users to caught criminals and terrorists. b) It can be used in controlling access to areas where security risks are especially high c) It can also be used for automatic access control. High security is required for accumulation of data and information. Government can use it for Law enforcement, Voter Verification, Security, Immigration, Legislature, prisons, Benefit Payments and to find out Missing Children and robbers. It can also be used in the Commercial applications like Banking, Gaming Industry, Residential Security, Internet, E-commerce, and Health Care. Therefore it expected to have lot of demand for real time face recognition systems for access control and not only for investigation of unlawful activities, it will be also useful to stop the of unlawful or terrorist activities. However, if it is not a real system, we can detect the activity after the incident is occurred but we cannot stop/prevent it before it happens. If the Face recognition systems to be used to stop any unlawful or terrorism activity or use it for access control, an user friendly real time face recognition system is required.

Literature review:

This Research work is concentrates on face recognition problem as a part of biometric attendance system.Face recognition has been an active research area over last 30 years. This Research spans several disciplines such as image processing, pattern recognition, computer vision, and neural networks. Face recognition has applications mainly in the fields of biometrics, access control, law enforcement, and security and surveillance systems

The problem of face recognition can be stated as follows:
Given still images or video of a scene, identifying one or more persons in the scene by using a stored database of faces. The problem is mainly a classification problem. Training the face recognition system with images from the known individuals and classifying the newly coming test images into one of the classes is the main aspect of the face recognition systems. This Problem seems to be easily solved by humans where limited memory can be the main problem; whereas the problems for a machine face recognition system are: Facial Expression Change Illumination Change Aging Pose change Scaling factor Presence and absence of spectacles, beard, mustache etc. Occlusion due to scarf, mask or obstacles in front. The problem of automatic face recognition (AFR) is a composite task that involves detection of faces from a cluttered background, facial feature extraction, and face identification. A complete face recognition system has to solve all sub problems, where each one is a separate research problem. This research work concentrates on the problem of facial feature extraction and face identification. Most of the current face recognition algorithms can be categorized into two classes, image template based and geometry feature-based. The template based methods compute the correlation between a face and one or more model templates to estimate the face identity. Where as feature based methods extract local facial features and use their geometric and appearance properties. The face is our primary focus of attention in social intercourse, playing a major role in conveying identity and emotion. We can recognize thousands of faces learned throughout our lifetime and identify familiar faces at a glance after years of separation. Computational models of face recognition, in particular, are interesting because they can contribute not only to theoretical insights but also to practical applications. Computers that recognize faces could be applied to a wide variety of problems, including criminal identification, security systems, image and film processing, and human computer interaction. Unfortunately, developing a computational model of face recognition is quite difficult, because faces are complex, multidimensional, and meaningful visual stimuli. Eigen face is a face recognition approach that can locate and track a subject's head, and then recognize the person by comparing characteristics of the face to those of known individuals

Problem Statement
The Aim of the Project is to develop a real time GUI based face recognition system that would use a camera to view the outside world and then detect and recognize the individual faces in the scene by using Open Face. Open Face is the face recognition tool developed using Open CV and we will using that as the input to mark attendance and store them in CS

Methodology and Pseudo Code:

Face Detection and Extract
First of all, function call is done which captures photo by turning on camera.
Extract Face function call obtains frontal face from videos by using haar cascade technique and loads the traiiner.xml as the classifier for the process. The output of “1” and “0” are given by classifier. "1" if the region shows the object (i.e., face), and "0" if it does not. If someone wants to search an object out of image, one needs to perform checking of every location with the help of classifier. This requires moving of search window. The classifier is very flexible and can easily be used for different sizes by "resizing" when possibility of getting objects at other sizes exist. By scanning at different

scales a previously unknown size object in the image can be found out. a gray scale image of 50x50 pixels is obtained after face detection process.

Learn and Train Face Images

A function is the function that performs the task of applying algorithms on the training set. The implementation of above function requires following four steps:

1Load the training data.

2Do Haar Cascade on it to find a subspace.

3Project the training faces.

4Store all the training information.

a. Eigenvalues
b. Eigenvectors
c. The average training face image
d. Projected face image
e. Person ID numbers

The built-in Open CV function is called to perform the HaraCascade, cvCalcEigen object (), which calculates subspace. Then an output variable is created by rest doPCA() that stores outcomes on returning of cvCalcEigenObjects () .

If we have to perform Haar Cascade, then we must make sure that the dateset is “centered”. The meaning for facial images that dateset should be “centered” means that the average image – a type of image where every pixel contains the average value of that pixel of all facial images that are part of training set. Then, the process of data set centering is done by deducting the values of pixels of average face form every training image. This process is performed under cvCalcEigenObjects ().

But we require us to get a grip on the average image, because later it will be necessary to project the data for that motive, and therefore assignment of memory for average image, which is a floating point image is compulsory. This step leads us to getting a subspace using algorithm of Haar-cascade and this subspace will contain training images converted into points. This process is named as “projecting” of training image. And the function of Open Cv that is used for this task is cvEigenDecomposite (). Finally, the built-in persistence functions of the Open CV store learned face representation data using an XML file. Recognize and Identification

Pseudoscope:

Step 1:Face recognition:

Start capturing images via a camera module placed in the bus: Get Started:

- Precapture the captured image and extract the face image.

- the first step is to perform calculation of eigen values of the captured face image and compare it with the eigen values of the existing faces in the training data Sets of college database.

- If the eigen value does not match the existing one, the storage of the information of new face image in the face database (xml file) should be done.

- If the eigen value matches with the existing one then the validation step will occur and it will give the notification in app.

End;

Step 2: Face recognition:

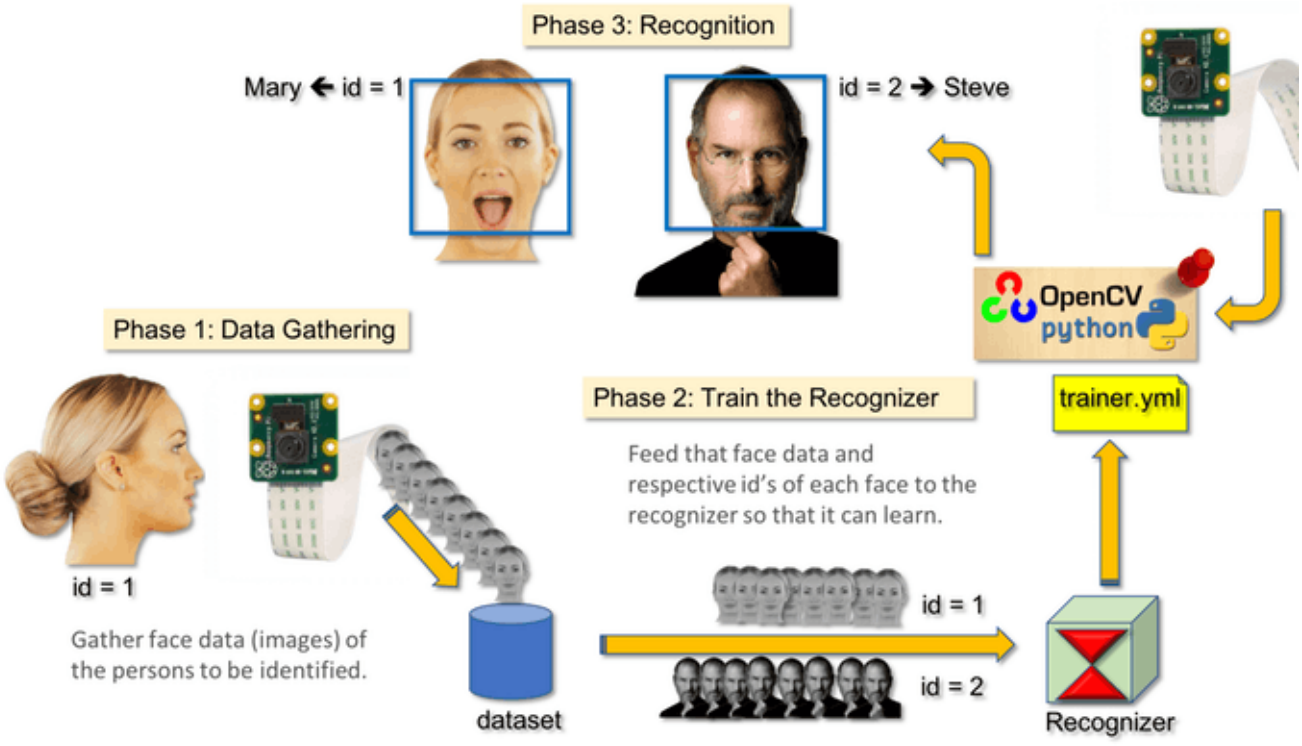
These steps will be performed for face recognition using the defeat cascade algorithm:

Start:

- Get the facial information of the matching face image from the database.

- Then updating of log table with the corresponding facial image and system time that completes attendance for an individual student should be done.

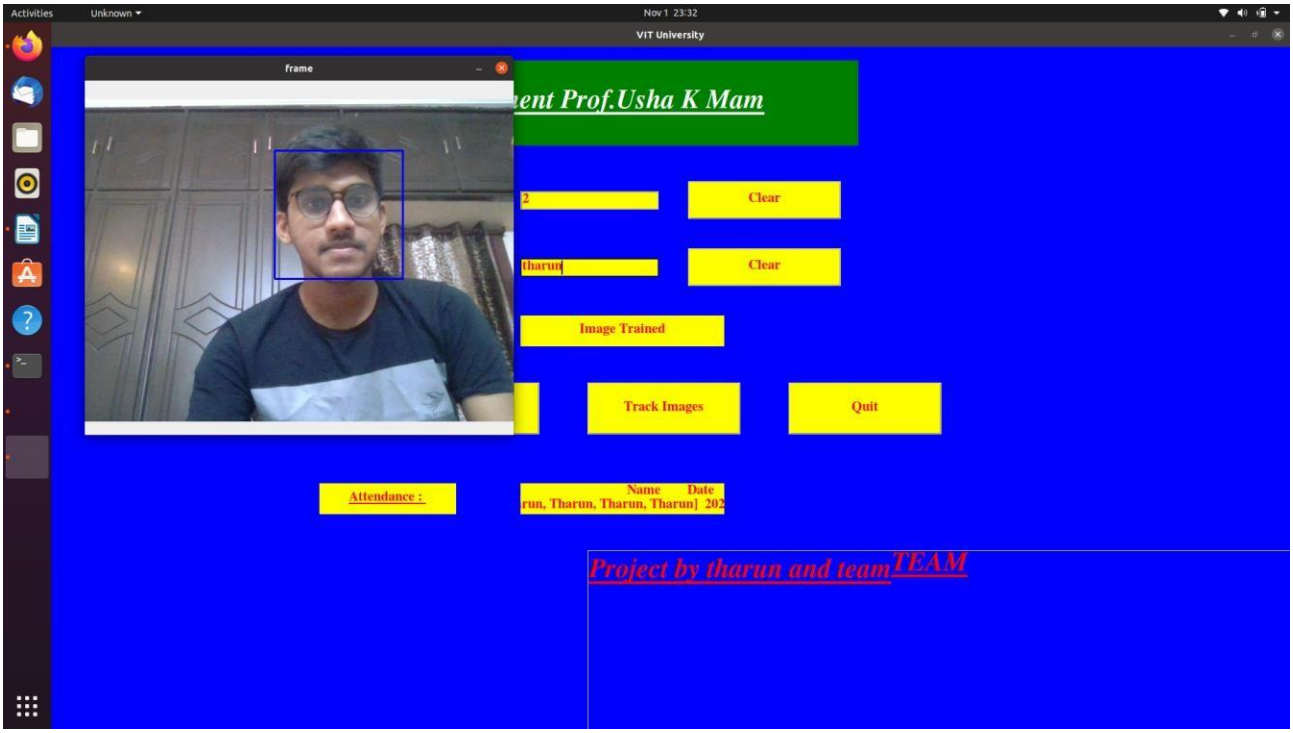
End;

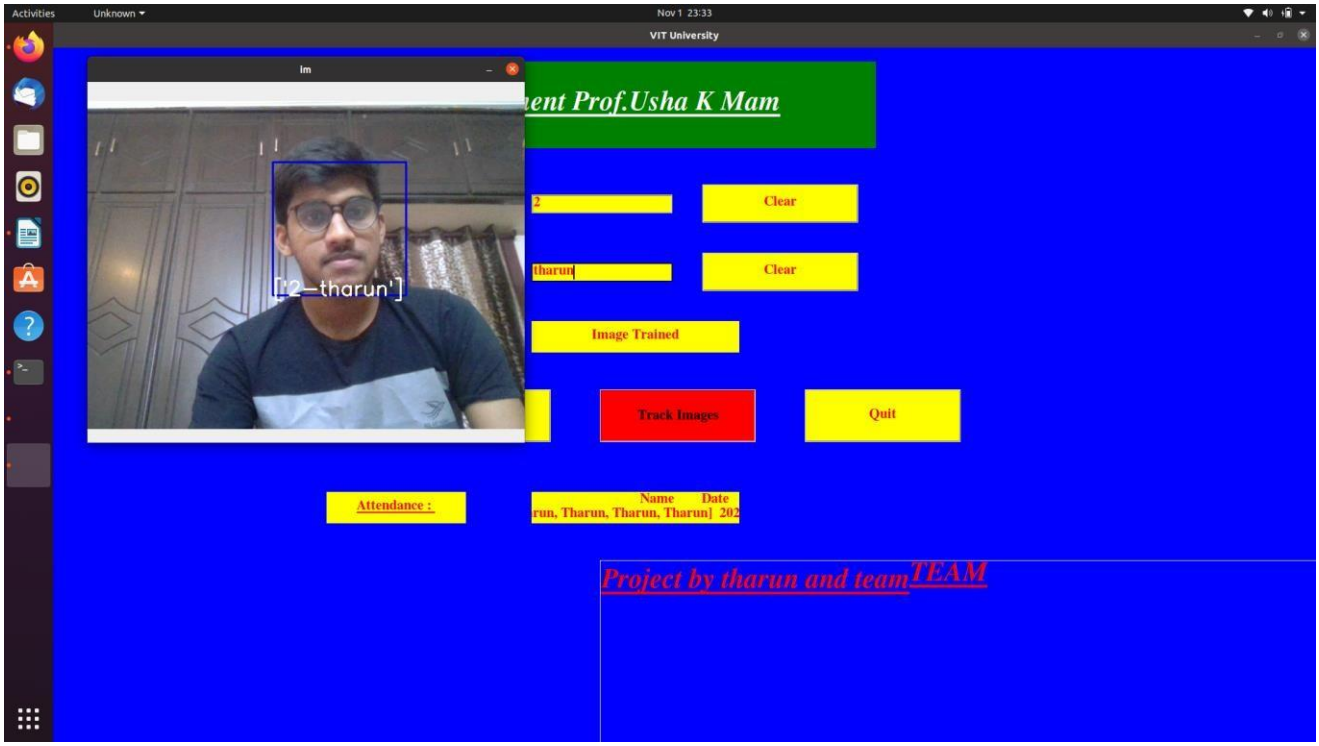


Tools used:

- Python3
- idna
- cx_freeze
- opencv-python
- wheel
- numpy
- pandas
- datetime
- pillow
- image
- opencv-contrib-python

Output Screenshots:





Pros and Cons:

Pros:

- It offers easy storage of templates in database.
- It reduces the statistic complexities to recognize face image.
- It involves no physical contact with the system.

Cons:

- Facial traits change over time.
- Uniqueness is not guaranteed, for example, in case of identical twins.
- If a candidate face shows different expressions such as light smile, then it can affect the result.
- It requires adequate lighting to get correct input.

Applications of Facial Recognition System:

- General Identity Verification.
- Verification for access control.
- Human-Computer Interaction

References:

- https://shervinemami.info/faceRecognition.html,
- https://arxiv.org/pdf/1503.03832.pdf
- https://www.cognotics.com/opencv/servo2007_series/part_1/index.html,
- https://opencv.willowgarage.com
- http://www.face-rec.org/algorithms/
- http://en.wikipedia.org /wiki /Threedimensional_face
- https://en.wikipedia.org/wiki/Active_appe arance_model
- en.cnki.com.cn/Article_en/CJFDTotal-HLGX200905034.htm

Code:

```
import tkinter as tk
from tkinter import Message ,Text
import cv2,os
import shutil
import csv
import numpy as np
from PIL import Image, ImageTk
import pandas as pd
import datetime
import time
import tkinter.ttk as ttk
import tkinter.font as font


window = tk.Tk()
#helv36 = tk.Font(family='Helvetica', size=36, weight='bold')
window.title("VIT University")


dialog_title = 'QUIT'
dialog_text = 'Are you sure?'
#answer = messagebox.askquestion(dialog_title, dialog_text)


#window.geometry('1280x720')
window.configure(background='blue')


#window.attributes('-fullscreen', True)
```

```
window.grid_rowconfigure(0, weight=1)
window.grid_columnconfigure(0, weight=1)


#path = "profile.jpg"


#Creates a Tkinter-compatible photo image, which can be used everywhere Tkinter expects an image object.
#img = ImageTk.PhotoImage(Image.open(path))


#The Label widget is a standard Tkinter widget used to display a text or image on the screen.
#panel = tk.Label(window, image = img)



#panel.pack(side = "left", fill = "y", expand = "no")


#cv_img = cv2.imread("img541.jpg")
#x, y, no_channels = cv_img.shape
#canvas = tk.Canvas(window, width = x, height =y)
#canvas.pack(side="left")
#photo = PIL.ImageTk.PhotoImage(image = PIL.Image.fromarray(cv_img))
# Add a PhotoImage to the Canvas
#canvas.create_image(0, 0, image=photo, anchor=tk.NW)


msg = Message(window, text='Tharun Tej')


# Font is a tuple of (font_family, size_in_points, style_modifier_string)




message = tk.Label(window, text="Bi0metrics J Component Prof.Usha K
Mam" ,bg="Green"  ,fg="white"  ,width=50  ,height=3,font=('times', 30, 'italic bold underline'))


message.place(x=200, y=20)


lbl = tk.Label(window, text="Enter ID",width=20  ,height=2  ,fg="red"  ,bg="yellow" ,font=('times', 15, ' bold ') )
lbl.place(x=400, y=200)


txt = tk.Entry(window,width=20  ,bg="yellow" ,fg="red",font=('times', 15, ' bold '))
txt.place(x=700, y=215)
```

```
lbl2 = tk.Label(window, text="Enter
Name",width=20  ,fg="red"  ,bg="yellow"    ,height=2 ,font=('times', 15, ' bold '))
lbl2.place(x=400, y=300)


txt2 = tk.Entry(window,width=20  ,bg="yellow"  ,fg="red",font=('times', 15, ' bold ')  )
txt2.place(x=700, y=315)


lbl3 = tk.Label(window, text="Notification :
",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold underline '))
lbl3.place(x=400, y=400)


message = tk.Label(window, text="" ,bg="yellow"  ,fg="red"  ,width=30  ,height=2,
activebackground = "yellow" ,font=('times', 15, ' bold '))
message.place(x=700, y=400)


lbl3 = tk.Label(window, text="Attendance :
",width=20  ,fg="red"  ,bg="yellow"  ,height=2 ,font=('times', 15, ' bold  underline'))
lbl3.place(x=400, y=650)



message2 = tk.Label(window, text="" ,fg="red"  ,bg="yellow",activeforeground =
"green",width=30  ,height=2  ,font=('times', 15, ' bold '))
message2.place(x=700, y=650)


def clear():
    txt.delete(0, 'end')
    res = ""
    message.configure(text= res)


def clear2():
    txt2.delete(0, 'end')
    res = ""
    message.configure(text= res)


def is_number(s):
    try:
        float(s)
        return True
    except ValueError:
        pass


    try:
        import unicodedata
        unicodedata.numeric(s)
        return True
```

```
    except (TypeError, ValueError):
        pass


    return  False


def TakeImages():
    Id=(txt.get())
    name=(txt2.get())
    if(is_number(Id) and name.isalpha()):
        cam = cv2.VideoCapture(0)
        harcascadePath = "haarcascade_frontalface_default.xml"
        detector=cv2.CascadeClassifier(harcascadePath)
        sampleNum=0
        while(True):
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.3, 5)
            for (x,y,w,h) in faces:
                cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
                #incrementing sample number
                sampleNum=sampleNum+1
                #saving the captured face in the dataset folder TrainingImage
                cv2.imwrite("TrainingImage/ "+name +"."+Id +'.'+ str(sampleNum) + ".jpg",
gray[y:y+h,x:x+w])
                #display the frame
                cv2.imshow('frame',img)
            #wait for 100 miliseconds
            if cv2.waitKey(100) & 0xFF == ord('q'):
                break
            # break if the sample number is morethan 100
            elif sampleNum>60:
                break
        cam.release()
        cv2.destroyAllWindows()
        res = "Images Saved for ID : " + Id +" Name : "+ name
        row = [Id , name]
        with open('StudentDetails/StudentDetails.csv','a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()
        message.configure(text= res)
    else:
        if(is_number(Id)):
            res = "Enter Alphabetical Name"
            message.configure(text= res)
        if(name.isalpha()):
            res = "Enter Numeric Id"
            message.configure(text= res)
```

```
def TrainImages():
    recognizer = cv2.face_LBPHFaceRecognizer.create()#recognizer =
cv2.face.LBPHFaceRecognizer_create()#$cv2.createLBPHFaceRecognizer()
    harcascadePath = "haarcascade_frontalface_default.xml"
    detector =cv2.CascadeClassifier(harcascadePath)
    faces,Id = getImagesAndLabels("TrainingImage")
    recognizer.train(faces, np.array(Id))
    recognizer.save("Trainner.yml")
    res = "Image Trained"#+",".join(str(f) for f in Id)
    message.configure(text= res)


def getImagesAndLabels(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #print(imagePaths)


    #create empth face list
    faces=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        pilImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        imageNp=np.array(pilImage,'uint8')
        #getting the Id from the image
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces,Ids


def TrackImages():
    recognizer = cv2.face.LBPHFaceRecognizer_create()#cv2.createLBPHFaceRecognizer()
    recognizer.read("Trainner.yml")
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath);
    df=pd.read_csv("StudentDetails/StudentDetails.csv")
    cam = cv2.VideoCapture(0)
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names =  ['Id','Name','Date','Time']
    attendance = pd.DataFrame(columns = col_names)
    while True:
        ret, im =cam.read()
        gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
        faces=faceCascade.detectMultiScale(gray, 1.2,5)
        for(x,y,w,h) in faces:
            cv2.rectangle(im,(x,y),(x+w,y+h),(225,0,0),2)
```

```
            Id, conf = recognizer.predict(gray[y:y+h,x:x+w])
            if(conf < 50):
                ts = time.time()
                date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
                timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
                aa=df.loc[df['Id'] == Id]['Name'].values
                tt=str(Id)+"-"+aa
                attendance.loc[len(attendance)] = [Id,aa,date,timeStamp]


            else:
                Id='Unknown'
                tt=str(Id)
            if(conf > 75):
                noOfFile=len(os.listdir("ImagesUnknown"))+1
                cv2.imwrite("ImagesUnknown/Image"+str(noOfFile) + ".jpg", im[y:y+h,x:x+w])
            cv2.putText(im,str(tt),(x,y+h), font, 1,(255,255,255),2)
        attendance=attendance.drop_duplicates(subset=['Id'],keep='first')
        cv2.imshow('im',im)
        if (cv2.waitKey(1)==ord('q')):
            break
    ts = time.time()
    date = datetime.datetime.fromtimestamp(ts).strftime('%Y-%m-%d')
    timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')
    Hour,Minute,Second=timeStamp.split(":")
    fileName="Attendance/Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
    attendance.to_csv(fileName,index=False)
    cam.release()
    cv2.destroyAllWindows()
    #print(attendance)
    res=attendance
    message2.configure(text= res)



clearButton = tk.Button(window, text="Clear",
command=clear  ,fg="red"  ,bg="yellow"  ,width=20  ,height=2 ,activebackground =
"Red" ,font=('times', 15, ' bold '))
clearButton.place(x=950, y=200)
clearButton2 = tk.Button(window, text="Clear",
command=clear2  ,fg="red"  ,bg="yellow" ,width=20  ,height=2, activebackground =
"Red" ,font=('times', 15, ' bold '))
clearButton2.place(x=950, y=300)
takeImg = tk.Button(window, text="Take Images",
command=TakeImages  ,fg="red"  ,bg="yellow"  ,width=20  ,height=3, activebackground =
"Red" ,font=('times', 15, ' bold '))
takeImg.place(x=200, y=500)
trainImg = tk.Button(window, text="Train Images",
command=TrainImages  ,fg="red"  ,bg="yellow"  ,width=20  ,height=3, activebackground =
"Red" ,font=('times', 15, ' bold '))
trainImg.place(x=500, y=500)
```

```
trackImg = tk.Button(window, text="Track Images",
command=TrackImages  ,fg="red"  ,bg="yellow"  ,width=20  ,height=3, activebackground =
"Red" ,font=('times', 15, ' bold '))
trackImg.place(x=800, y=500)
quitWindow = tk.Button(window, text="Quit",
command=window.destroy  ,fg="red"  ,bg="yellow"  ,width=20  ,height=3, activebackground =
"Red" ,font=('times', 15, ' bold '))
quitWindow.place(x=1100, y=500)
copyWrite = tk.Text(window, background=window.cget("background"), borderwidth=0,font=('times',
30, 'italic bold underline'))
copyWrite.tag_configure("superscript", offset=10)
copyWrite.insert("insert", "Project by tharun and team","", "TEAM", "superscript")
copyWrite.configure(state="disabled",fg="red" )
copyWrite.pack(side="left")
copyWrite.place(x=800, y=750)


window.mainloop()
```