

AUTOMATED ATTENDANCE SYSTEM USING FACE RECOGNITION

CS 6611 - CREATIVE AND INNOVATIVE PROJECT REPORT

Submitted by

JUMANA BEGUM. J (2019103022)

SAMREEN AYESHA. S (2019103055)

HARINI. E. R (2019103019)

In partial fulfillment of the requirements for the award of the

degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



COLLEGE OF ENGINEERING, GUINDY,

ANNA UNIVERSITY: CHENNAI 600 025

JUNE 2022

ACKNOWLEDGEMENT

Foremost, we would like to express our sincere gratitude to our project guide, **Mrs. Lalitha Devi K**, Teaching Fellow, Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her constant source of inspiration. We thank her for the continuous support and guidance which was instrumental in taking the project to successful completion.

We are grateful to **Dr.S.Valli**, Professor and Head , Department of Computer Science and Engineering, College of Engineering Guindy, Chennai for her support and for providing necessary facilities to carry out for our project.

We would also like to thank our friends and family for their encouragement and continued support. We would also like to thank the Almighty for giving us the moral strength to accomplish our task.

JUMANA BEGUM. J

E.R.HARINI

SAMREEN AYESHA. S

TABLE OF CONTENTS

INDEX NO	TITLE	PAGE NO
	ABSTRACT	4
1.	INTRODUCTION	4
	1.1 OBJECTIVE	5
	1.2 PROBLEM STATEMENT	5
	1.3 CHALLENGES IN THE SYSTEM	5
	1.4 SCOPE OF THE SYSTEM	6
2.	LITERATURE SURVEY	6
3.	PROPOSED APPROACH	8
4.	SYSTEM DESIGN	8
	4.1 BLOCK DIAGRAM	9
	4.2 SYSTEM REQUIREMENTS	10
5.	DETAILED ARCHITECTURE	10
	5.1 FACE DETECTION	10
	5.2 FACE RECOGNITION	12
	5.3 ATTENDANCE UPDATION	14
6.	IMPLEMENTATION	16
7.	RESULT ANALYSIS	41
8.	CONCLUSION AND FUTURE WORK	45
	REFERENCES	45

ABSTRACT

Taking attendance of students in schools and colleges is a mandatory task. But teachers taking attendance manually by calling out names is a tedious and time-consuming process. In addition to that it may not be error-free as there is a higher chance of proxy. That is why here we propose to automate the attendance system via face recognition. Face recognition-based attendance overcomes the cons of manual attendance and reduces manual labour. Our objective is to develop a face recognition-based attendance using machine learning algorithms such as Haar cascade for face detection and Local Binary Pattern Histogram for face recognition. Though there are other euclidean distance-based algorithms like Eigenfaces and Fisherfaces, Local Binary Pattern Histogram (LBPH) algorithm is better. We propose Haar cascade for face detection because of their robustness against monotonic grayscale transformations.

1. INTRODUCTION

Taking attendance of the students in schools and colleges is a mandatory task. But for the teachers, taking attendance manually by calling out the names of the students is a tedious and time-consuming process. In addition to that it is prone to error as there is a higher chance of proxy. That is why here we propose to automate the attendance system via face recognition. Face Recognition-based Attendance System overcomes the pitfall of manual attendance and reduces manual labour.

Face belongs to a tier of biometrics such as voice, fingerprint and iris used to recognise people uniquely. Face recognition-based applications is not something new. In real time, face recognition is used to unlock phones, to aid the blind in identifying a person and in forensic investigations.

Attendance systems done with RFID reader is fast and efficient in terms of time and space, but it has a higher chance of proxy. As anyone with the tag can mark the student as present. Fingerprint based attendance minimises the proxy but has safety issues and fingerprints are found to be faded for some people. Hence, there is a need to move to face-recognition based attendance system which overcomes above mentioned flaws of automated attendance system.

1.1. OBJECTIVE

Our objective here is to develop a Face-Recognition based Attendance using machine learning algorithms such as Haar Cascade for face detection and Local Binary Pattern Histogram for face recognition. We propose Haar cascade for face detection because of their low computational time. And we use LBPH algorithm to recognise faces because of their robustness against monotonic grayscale transformations, computational simplicity and discriminative power. This project aims to recognize students even when students are wearing glasses or have grown facial hair.

1.2. PROBLEM STATEMENT

As mentioned earlier the RFID based attendance have higher chance of proxy and fingerprint-based attendance is not found to be reliable. Thus, a reliable automated attendance system which can recognise a person under different lighting conditions is essential. The automated attendance system should recognise multiple faces at a time. It should be robust to frontal and side face recognitions. And should recognise the person even after slight facial changes. In this system, student's faces are detected by face detection module from the images captured by the camera and face recognition is done for those detected faces and the attendance sheet is updated accordingly.

1.3. CHALLENGES FACED IN THE SYSTEM

- **ILLUMINATION:** Face recognition systems are highly sensitive to varying light conditions. Although the person looks the same and gives the same expressions because of varying light conditions, results might vary.
- **POSE:** Another major challenge is to recognise the person at different angles of their faces. Images captured at real time by the camera may not always be frontal faces. Developing a system which is robust to such images is a difficult task. This requires us to generate a dataset which holds all possible angles of the faces and train the model with such an extensive dataset so as to detect without any anomalies. But accomplishing this becomes difficult when done on a large scale.

- **EMOTIONS:** Voluntary or involuntary facial emotions can affect the accuracy of the automated attendance system.
- **LOW RESOLUTION OF CAMERA:** When a camera captures a low-resolution image, extracting facial features from such images becomes tough. If features are not properly detected, face recognition becomes erroneous.
- **MOTION:** Continuous motion in front of the camera results imprecise detection of faces and generates blurred images. These blurred images are not accepted by the system for recognition and are eventually discarded by the system.
- **FACIAL CHANGES:** Over the course of time, the person in the dataset might undergo facial changes which becomes difficult for the automated attendance system to recognise.

1.4. SCOPE OF THE SYSTEM

Pandemic such as Covid-19 has driven people to adopt contactless technologies such as face automated attendance system thereby minimizing physical contact. Although the face-based attendance system is not 100 % accurate, if the drawbacks are removed automated attendance system can be used to recognise people in real time with minimal human intervention.

2. LITERATURE SURVEY

Visar Shehu et. al. [1] In this paper a new approach in automatic attendance management systems, extended with computer vision algorithms is proposed. Using real time face detection algorithms integrated on an existing Learning Management System (LMS), automatic detection and registration of students attending on a lecture is done. Here HAAR classifier is used for face detection. The system implements a server-based module, programmed in Python which takes benefit of eigenfaces to identify a face

Nirmalya Kar et. al. [2] This paper describes a method for Student's Attendance System which will integrate with the face recognition technology using Personal Component Analysis

(PCA) algorithm. The authors used the eigenface approach for face recognition. This system has prevented the fake attendance due to the implementation of clock time which is used for checking the students' presence inside the class for the period or not. A drawback is that the recognition task is high in frontal face only.

Jomon Joseph et. al. [3] Automated attendance is implemented using Principle Component Analysis (PCA) with MATLAB and eigenfaces for face recognition. In this proposed model, a feature set for each of the images provided in the database is generated using PCA. During real time, the images of human face may be extracted from a USB camera. This involves MATLAB's Image Acquisition Toolbox, using which a camera is configured, accessed & brought one frame at a time into MATLAB's workspace for further processing using MATLAB's Image Processing Toolbox. The pre-processing that includes the cropping of the region of interest (ROI) makes the accuracy higher.

Rekha, E. et. al. [4] The objective of this paper is to automate the attendance system by integrating the face recognition technology using Eigen Face database and PCA algorithm with Matlab GUI. There are many limitations in implementing face recognition technologies like Image Quality, Image Size, Face angle, varying intensity of light. In order to overcome these issues various techniques like Illumination Invariant, Histogram equalization, PCA are used. A drawback of this model is that the computational cost is high because of the algorithm implemented.

Priyanka Wagh et. al. [5] This paper aims to overcome disadvantages like intensity of light problem and head pose problem by using various techniques like illumination invariant, Viola and Jones algorithm and Principle component analysis. Better accuracy is obtained by implementing Ada-Boost algorithm for face detection

Smitha et. al. [6] proposed a system in which Face detection and recognition is performed using Haar-Cascade classifier and Local Binary Pattern Histogram algorithm respectively.

Nandhini R. et. al. [7] In this paper deep learning methods are used for improving the accuracy of face recognition. CNN algorithm can be implemented to detect the faces. A CNN (Convolution Neural Network) uses a system like a multilayer perceptron that has been designed to process the requirements faster. The CNN layer consist of an input layer, an output layer and a hidden layer that includes multiple convolution layers, pooling layers, fully connected layers, and normalization layers. The removal of limitations and increase in efficiency for image processing results in a system that is far more effective, simpler to trains limited for image processing.

3. PROPOSED APPROACH

We have implemented an automated attendance system which would recognise the person captured by the camera and update the attendance sheet accordingly. The dataset for the system has been created using the images captured by the camera after necessary pre-processing to train the LBPH model. For face detection Haar cascade frontal face and profile face classifiers are used. In order to detect the students under different lighting conditions, histograms of the image are normalised by CLAHE (Contrast Limited Adaptive Histogram Equalization) before face detection. For face recognition, LBPH recogniser is used. In order to better recognise the person, three different face recognisers trained with three kinds of dataset (with front faces, left sided faces and right sided faces) is used and the best of the label is chosen based on the lowest confidence score. To avoid unnecessary computations, blur images are removed and to better recognise the person some images are sharpened. The system automatically generates attendance sheet everyday with the list of students. The system after recognising the person appropriately, updates the attendance sheet. In case the person is not recognised, he/she is added to the intruder's folder. After updating the attendance sheet, the system sends appropriate mail to the parents/guardian and to the teachers using smtplib functions

4. SYSTEM DESIGN

The system design is illustrated by the diagram given below

There are three modules namely face detection, face recognition and attendance updation.

4.1. BLOCK DIAGRAM

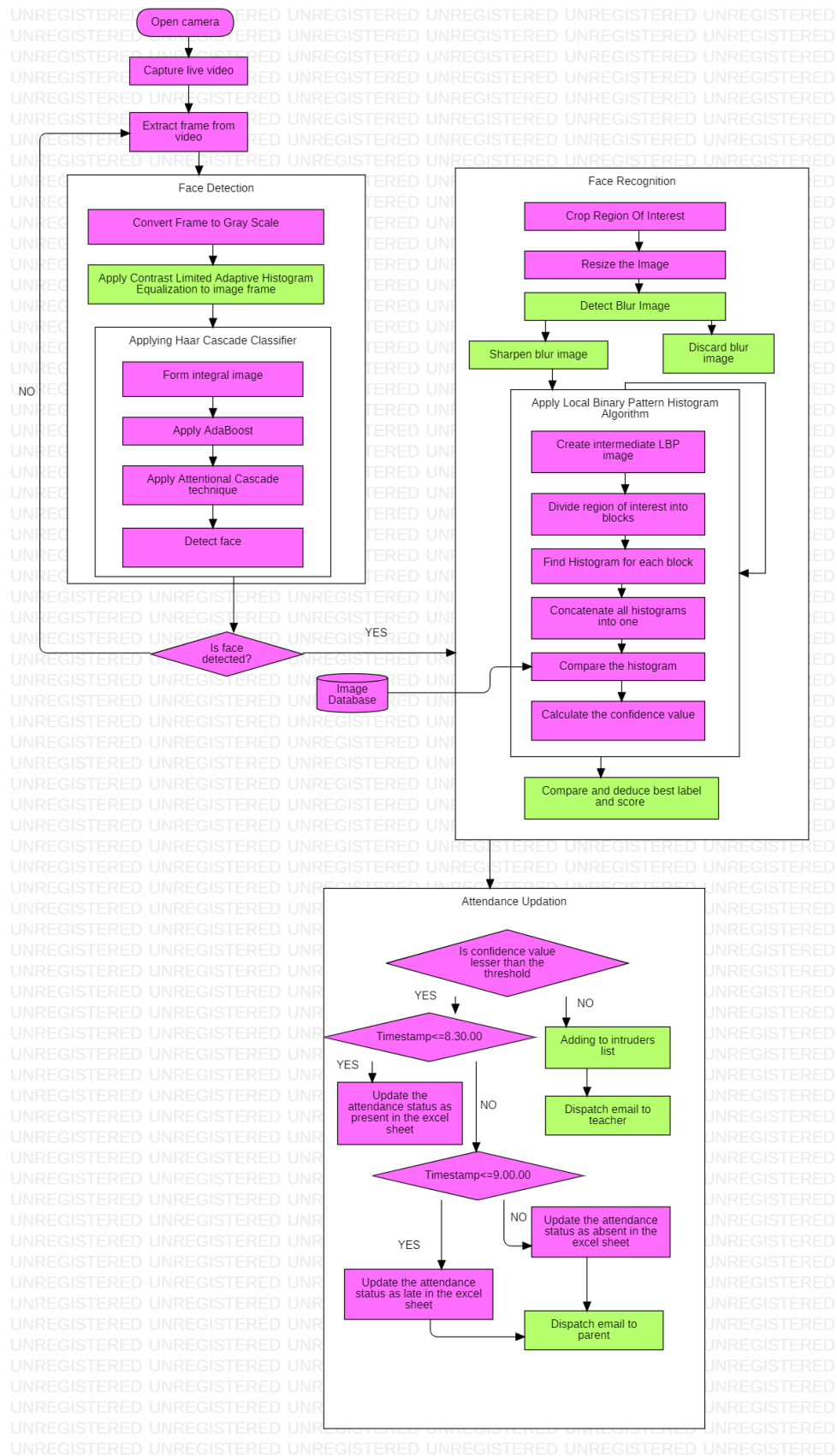


Figure 4.1 Block Diagram

4.2. SYSTEM REQUIREMENTS

- Python 3.9.7
- Webcam with resolution 640×480 or above
- Python packages: cv2, os, re, numpy, csv, datetime, pandas, math, PIL, smtplib, email (MIMEMultipart, MIMEBase, MIMEText, encoders), sci kit learn
- Jupyter notebook 6.4.5

5. DETAILED ARCHITECTURE:

Detailed Block Diagram for each module

- Face Detection
- Face recognition
- Attendance updation

5.1. FACE DETECTION

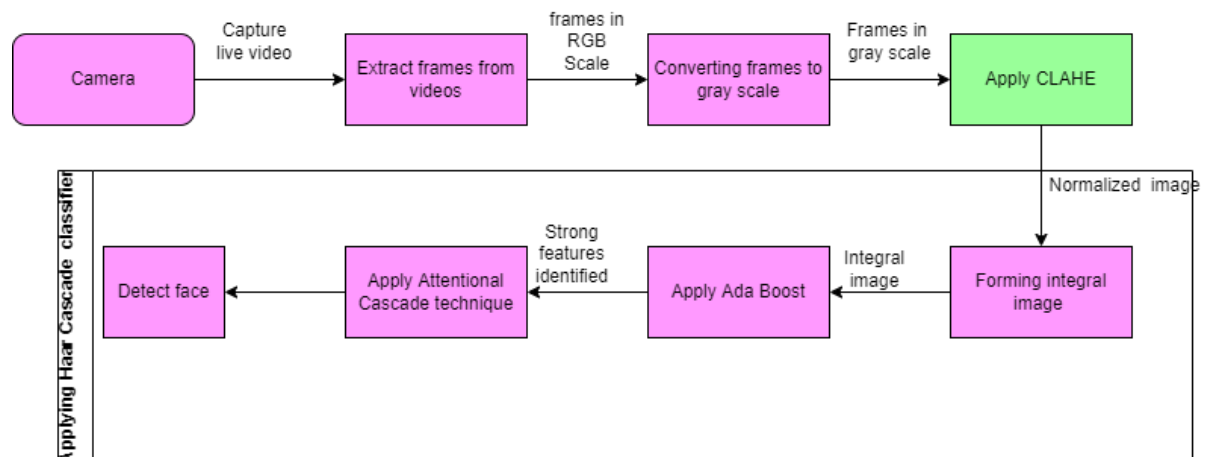


Figure 5.1 Block Diagram of Face Detection

This module takes frames from the live video stream captured by the camera as input. It converts the RGB image to GRAY scale image. In order to enhance the image contrast, Contrast Limited Adaptive Histogram Equalization technique is applied. HaarCascade is a

Viola-Jones Algorithm to detect faces. Once image is preprocessed, HaarCascade classifier is applied. It converts the image to an Integral image to improve the speed of computation of differences between sum of the pixels in the darker area of the Haar features and sum of the pixels in the lighter area of the Haar features. Then it applies Adaboost method to ignore irrelevant features and focus on prominent features to detect the face. This Algorithm is further improved by applying Attentional Cascade which ignores negative images if it doesn't find a feature, in order to avoid further processing of the image and decrease the computation time.

INPUT: Frames in RGB scale

OUTPUT: Frames with Bounding Box around detected faces

Pseudocode:

Algorithm 1: Face detection

```
1: Loading classifiers
2: while open_camera==true and capture_video==true:
3:   Read each frame
4:   while each frame is open:
5:     Convert to gray
6:     Apply clahe
7:     Apply classifiers
8:     Detect face
9:     if face detected:
10:      Draw bounding box in frame
11:    end if
```

5.2. FACE RECOGNITION

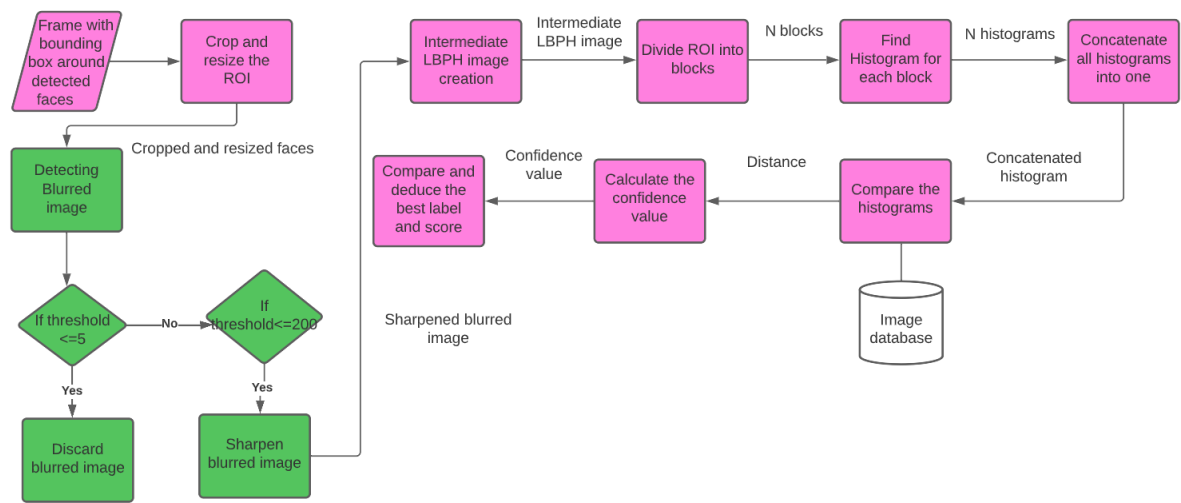


Figure 5.2 Block Diagram of Face Recognition

The region of interest is cropped from the image frames based on the coordinates of the bounding box. After cropping the faces (ROI) the images are uniformly resized to the same dimensions. The cropped faces are converted to intermediate LBP image. Then they are divided into blocks and a histogram is calculated for each block. Then all the obtained histograms are concatenated into one and is compared with the histograms of the student images dataset. The confidence can be measured by calculating the distance between two histograms using the Euclidean distance formula.

Lesser distance translates to lower confidence. A threshold value is set to estimate if the face has been correctly recognized. If the confidence is lower than the threshold value then it is assumed that the face has been correctly recognized.

INPUT: Frames with Bounding Box around detected faces

OUTPUT: Label and score of the student identified or recognized

Pseudocode:

Algorithm 1(contd): Face recognition

```
12:   for each face detected:
13:       Crop the face
14:       Resize the face
15:       if variance_of_laplace <= 5 : #Blur handling
16:           discard image
17:       elif variance_of_laplace <= 200 :
18:           sharpen image
19:       end elif
20:       label,score = face.predict() #Predict the face
```

TRAIN FUNCTION

Train():

```
1: Train the LBPH model using faces and label
```

PREDICT FUNCTION

```
1: predict():
2:   Predict the face using LBPH model
3:   return label and score
```

5.3. ATTENDANCE UPDATION

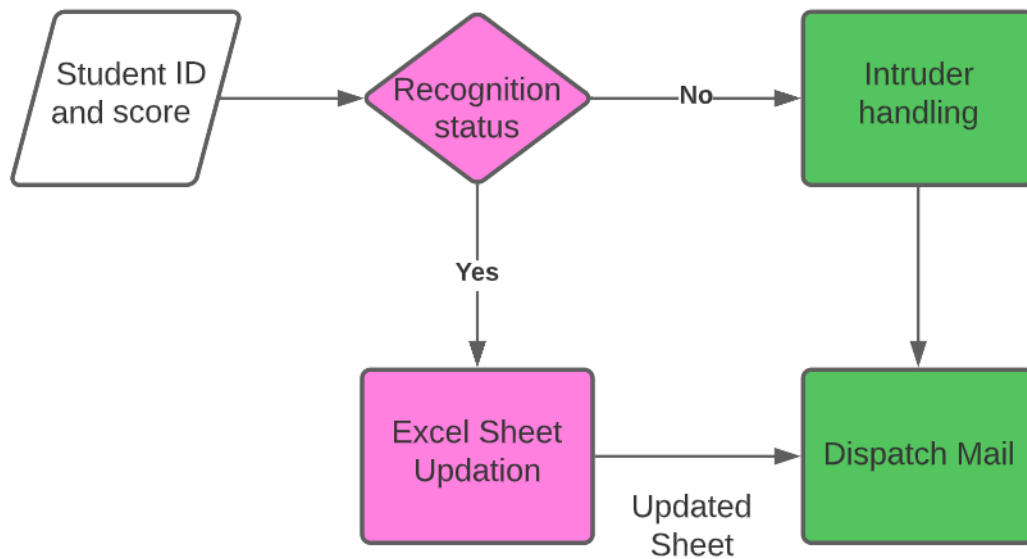


Figure 5.3 Block Diagram of Attendance Updation

Once the face is successfully recognized, the timestamp is noted. If the timestamp is less than 8.30 am then the attendance status for that student is updated as 'Present' in the excel sheet. If the timestamp is after 8.30 am but before 9.00 am then the attendance status for that student is updated as 'Late' in the excel sheet. Any timestamp after 9.00 am is updated as 'Absent'. If the face is not recognized then the system labels the person as an intruder and adds him/her to the intruder list. Mail is dispatched to the parent/guardian in case the student is late or absent. Mail is also sent to the teacher if an intruder is present in the classroom.

INPUT: Label and score of the student identified and time of entry of the student.

OUTPUT: Updating of status as present, absent, or late depending on time of entry, Intruder addition to intruder list, Dispatching of email to the parents/guardian regarding absent or late Students. Dispatching email to the class teacher regarding the intruder.

Pseudocode:

Algorithm 1(contd): Attendance updation

21: if score \geq 30:

```
22:         Write the intruder image in a folder
23:     else:
24:         Update attendance in csv file ()
25:     end of for
26: end of while
27: end of while
28: Send Mail To Parents
29: Send Intruder To Teacher
```

UPDATE ATTENDANCE FUNCTION:

```
1. open the attendance csv file
2. if score<=20:
3.     for row in csv_file:
4.         if label == row[0]: #searching for the name column
5.             if currentTime < today830am and Status=="Absent":
6.                 Assign Status = 'Present'
7.                 Assign Time of entry = currentTime
8.             elif currentTime < today900am and Status=="Absent":
9.                 Assign Status = 'Late'
10.                Assign Time of entry = currentTime
11.            end of elif
12.        end of if
13.    end of for
14. end of if
```

6. IMPLEMENTATION WITH EXECUTION SNAPSHOT

DATASET CREATION

For Left Side Profile Face, a dataset of 9 students with 10 images for each student has been created. Similarly for Right Side Profile Face and for Frontal face, datasets of 9 students with 10 images each has been created.

The images are converted to grayscale, CLAHE and blurred images handling was applied for better results.

LEFT SIDE PROFILE FACE DATASET



Figure 6.1 Left Sided Profile Faces

RIGHT SIDE PROFILE FACE DATASET



Figure 6.2 Right Sided Profile Faces

FRONTAL FACE DATASET



Figure 6.3 Frontal Faces

FACE DETECTION

Libraries and Global Variables required:

```
import os

import re

import cv2

import numpy as np

import csv

from datetime import date,datetime

import math

import pandas as pd

from PIL import Image

import smtplib


studentlabels={'3019' : 'Harini' , '3022' : 'Jumana Begum', '3026' : 'Karthika' , '3030' : 'Kirupa'
, '3039' : 'Narmatha' , '3047' : 'Preethi', '3049' : 'Ramya', '3055' : 'Samreen Ayesha', '3063' : 'Sineka'
}

now = datetime.now()

today830am = now.replace(hour=8, minute=30, second=0, microsecond=0)

today900am = now.replace(hour=9, minute=0, second=0, microsecond=0)
```

Driver Program:

```
#Loading classifiers

faceCascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")

sideCascade = cv2.CascadeClassifier("haarcascade_profileface.xml")
```

```

cap = cv2.VideoCapture(0)

if (cap.isOpened() == False):

    print("Error opening video stream or file")

i = 0

while(cap.isOpened()):

    ret, frame = cap.read()

    if ret == True:

        #gray scaling

        gray_frame=cv2.cvtColor(frame,cv2.COLOR_RGB2GRAY)

        #clahe

        clahe = cv2.createCLAHE(clipLimit =2.0, tileGridSize=(8,8))

        gray_frame = clahe.apply(gray_frame)


    faces1 = faceCascade.detectMultiScale(

        gray_frame,

        scaleFactor = 1.05,

        minNeighbors = 20,

        minSize = (100,100),

    )

    faces2 = sideCascade.detectMultiScale(

        gray_frame,

        scaleFactor = 1.05,

        minNeighbors = 20,

        minSize = (100,100),

```

)

for(x,y,w,h) in faces1:

```
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,255),4)
```

for(x,y,w,h) in faces2:

```
cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,255),4)
```

OUTPUT: Bounding box around the face detected

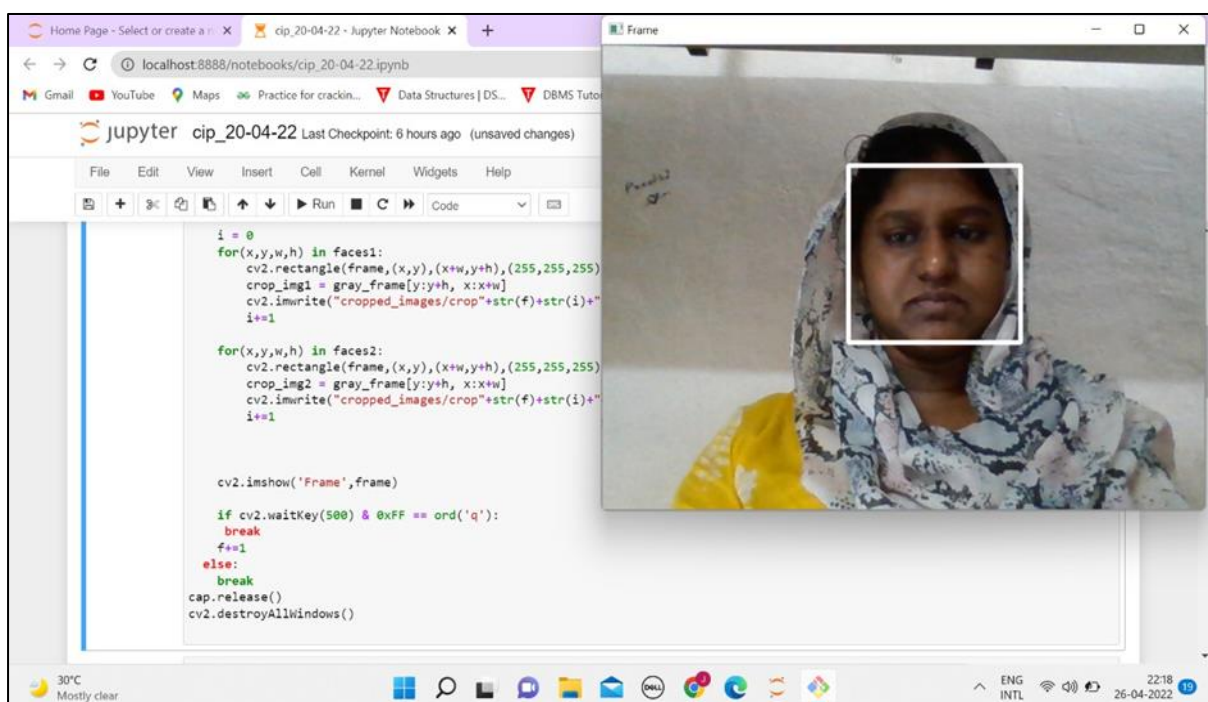


Figure 6.4 Output of Face Detection

FACE RECOGNITION

Driver Program (contd):

```
width = 500;

height = 500;

for(x,y,w,h) in faces1:

    #crop

    crop_img1 = gray_frame[y:y+h, x:x+w]

    #resize

    crop_img1 = cv2.resize(crop_img1, (width, height))

    #blur image handling

    if(variance_of_laplacian(crop_img1) <= 5):

        print("discarded : "+str(variance_of_laplacian(crop_img1)))

        continue

    elif(variance_of_laplacian(crop_img1) <= 200):

        kernel = np.array([[0, -1, 0],

                            [-1, 5, -1],

                            [0, -1, 0]])

        image = cv2.filter2D(src=crop_img1, ddepth=-1, kernel=kernel)

    #predict

    label_text1,score1 = predict_front(image)

    label_text2,score2 = predict_left_side(image)

    label_text3,score3 = predict_right_side(image)

    if score1<score2 and score1<score3:
```

```

        label_text = label_text1

        score = score1

    elif score2 < score1 and score1 < score3:

        label_text = label_text2

        score = score2

    else:

        label_text = label_text3

        score = score3

    print(label_text, score)

for(x,y,w,h) in faces2:

    cv2.rectangle(frame,(x,y),(x+w,y+h),(255,255,255),4)

    #crop

    crop_img2 = gray_frame[y:y+h, x:x+w]

    #resize

    crop_img2 = cv2.resize(crop_img2, (width, height))

    #sharpen

    if(variance_of_laplacian(crop_img2) <= 5):

        print("discarded : "+str(variance_of_laplacian(crop_img2)))

        continue

    elif(variance_of_laplacian(crop_img2) <= 200):

        kernel = np.array([[0, -1, 0],

                            [-1, 5, -1],

                            [0, -1, 0]])

```



```

        image = cv2.filter2D(src=crop_img2, ddepth=-1, kernel=kernel)

#predict

label_text1,score1 = predict_front(image)

label_text2,score2 = predict_left_side(image)

label_text3,score3 = predict_right_side(image)

```

TRAIN PROGRAM FOR FRONTAL FACE DATASET:

```

#training the LBPH model for frontal face

faces = []

labels = []

path = "C:/Users/GCS/CIP/trainFront_new"

dir_list = os.listdir(path)

for elt in dir_list:

    impath = path+'/'+elt

    image = cv2.imread(impath)

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

    mystring = impath.split("/")[5]

    mystring = mystring.split("(")[0]

    labels.append(int(mystring))

    faces.append(gray)

print("Total faces: ", len(faces))

```

```
print("Total labels: ", len(labels))
```

```
face_recognizer_front = cv2.face.LBPHFaceRecognizer_create(radius=5, neighbors=4,  
grid_x=8, grid_y=8)
```

```
face_recognizer_front.train(faces, np.array(labels))
```

TRAIN PROGRAM FOR LEFT PROFILE FACE DATASET:

```
#training the LBPH model for left side
```

```
faces = []
```

```
labels = []
```

```
path = "C:/Users/GCS/CIP/train_left_side_new"
```

```
dir_list = os.listdir(path)
```

```
for elt in dir_list:
```

```
    impath = path+'/'+elt
```

```
    image = cv2.imread(impath)
```

```
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    mystring = impath.split("/")[-1]
```

```
    mystring = mystring.split("(")[0]
```

```
    labels.append(int(mystring))
```

```
    faces.append(gray)
```

```
print("Total faces: ", len(faces))
```

```
print("Total labels: ", len(labels))
```

```
face_recognizer_left_side = cv2.face.LBPHFaceRecognizer_create(radius=5, neighbors=4,
grid_x=8, grid_y=8)

face_recognizer_left_side.train(faces, np.array(labels))
```

TRAIN PROGRAM FOR RIGHT PROFILE FACE DATASET:

```
faces = []
```

```
labels = []
```

```
path = "C:/Users/GCS/CIP/train_right_side_new"
```

```
dir_list = os.listdir(path)
```

```
for elt in dir_list:
```

```
    impath = path+'/'+elt
```

```
    image = cv2.imread(impath)
```

```
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    mystring = impath.split("/")[5]
```

```
    mystring = mystring.split("(")[0]
```

```
    labels.append(int(mystring))
```

```
    faces.append(gray)
```

```
print("Total faces: ", len(faces))
```

```
print("Total labels: ", len(labels))
```

```
face_recognizer_right_side = cv2.face.LBPHFaceRecognizer_create(radius=5, neighbors=4,
grid_x=8, grid_y=8)
```

```
face_recognizer_right_side.train(faces, np.array(labels))
```

PREDICT FUNCTIONS:

```
def predict_front(test_img):
```

```
    label= face_recognizer_front.predict(test_img)
```

```
    label_text = studentlabels[str(label[0])]
```

```
    score = label[1]
```

```
    score = math.ceil(score)
```

```
    return label_text,score
```

```
def predict_left_side(test_img):
```

```
    label= face_recognizer_left_side.predict(test_img)
```

```
    label_text = studentlabels[str(label[0])]
```

```
    score = label[1]
```

```
    score = math.ceil(score)
```

```
    return label_text,score
```

```
def predict_right_side(test_img):
```

```
    label= face_recognizer_right_side.predict(test_img)
```

```
    label_text = studentlabels[str(label[0])]
```

```
    score = label[1]
```

```
    score = math.ceil(score)
```

```
    return label_text,score
```

VARIANCE OF LAPLACIAN FUNCTION:

```
def variance_of_laplacian(image):  
  
    return cv2.Laplacian(image, cv2.CV_64F).var()
```

OUTPUTS

Cropped faces after face detection

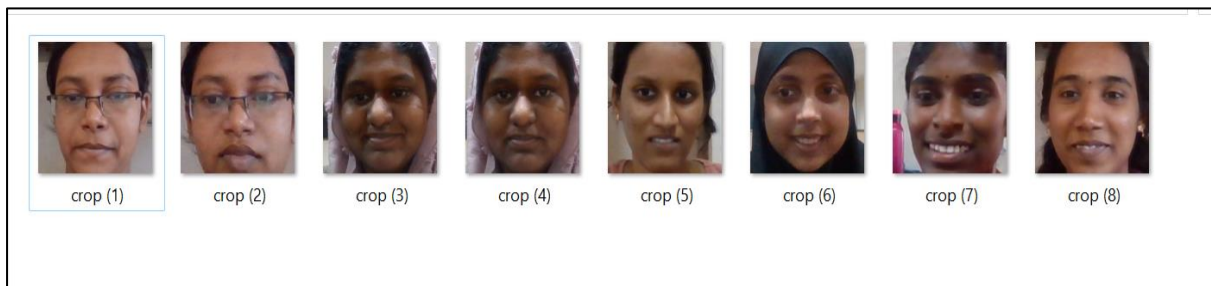


Figure 6.5 Cropped Faces after Face Detection

The images discarded in real time (blur detection and handling)

```
discarded : 2.0489238236  
Kirupa 14  
Kirupa 23  
Kirupa 23  
Kirupa 23  
Kirupa 23  
Kirupa 23  
Kirupa 22  
Kirupa 21  
discarded : 4.3001639942239995  
discarded : 3.5588999964  
Jumana Begum 30  
Jumana Begum 26  
Jumana Begum 27  
Jumana Begum 27  
Jumana Begum 26  
Jumana Begum 27  
Jumana Begum 25
```

Figure 6.6 Real time Face Recognition

Training the LBPH model

```
#training the LBPH model for front
faces = []
labels = []

path = "C:/Users/GCS/CIP/CIP/trainFront"
dir_list = os.listdir(path)

for elt in dir_list:
    impath = path+'/'+elt
    image = cv2.imread(impath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    mystring = impath.split("/")[6]
    mystring = mystring.split("(")[0]
    labels.append(int(mystring))
    faces.append(gray)

print("Total faces: ", len(faces))
print("Total labels: ", len(labels))

face_recognizer_front = cv2.face.LBPHFaceRecognizer_create(radius=5, neighbors=4, grid_x=8, grid_y=8)
face_recognizer_front.train(faces, np.array(labels))
```

```
Total faces: 90
Total labels: 90
```

Figure 6.7 Training Front Face Recogniser

```
#training the LBPH model for left side
faces = []
labels = []

path = "C:/Users/GCS/CIP/CIP/train_left_side"
dir_list = os.listdir(path)

for elt in dir_list:
    impath = path+'/'+elt
    image = cv2.imread(impath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    mystring = impath.split("/")[6]
    mystring = mystring.split("(")[0]
    labels.append(int(mystring))
    faces.append(gray)

print("Total faces: ", len(faces))
print("Total labels: ", len(labels))

face_recognizer_left_side = cv2.face.LBPHFaceRecognizer_create(radius=5, neighbors=4, grid_x=8, grid_y=8)
face_recognizer_left_side.train(faces, np.array(labels))
```

```
Total faces: 90
Total labels: 90
```

Figure 6.8 Training Left Sided Face Recogniser

The Label and score for a face (Jumana Begum 50)

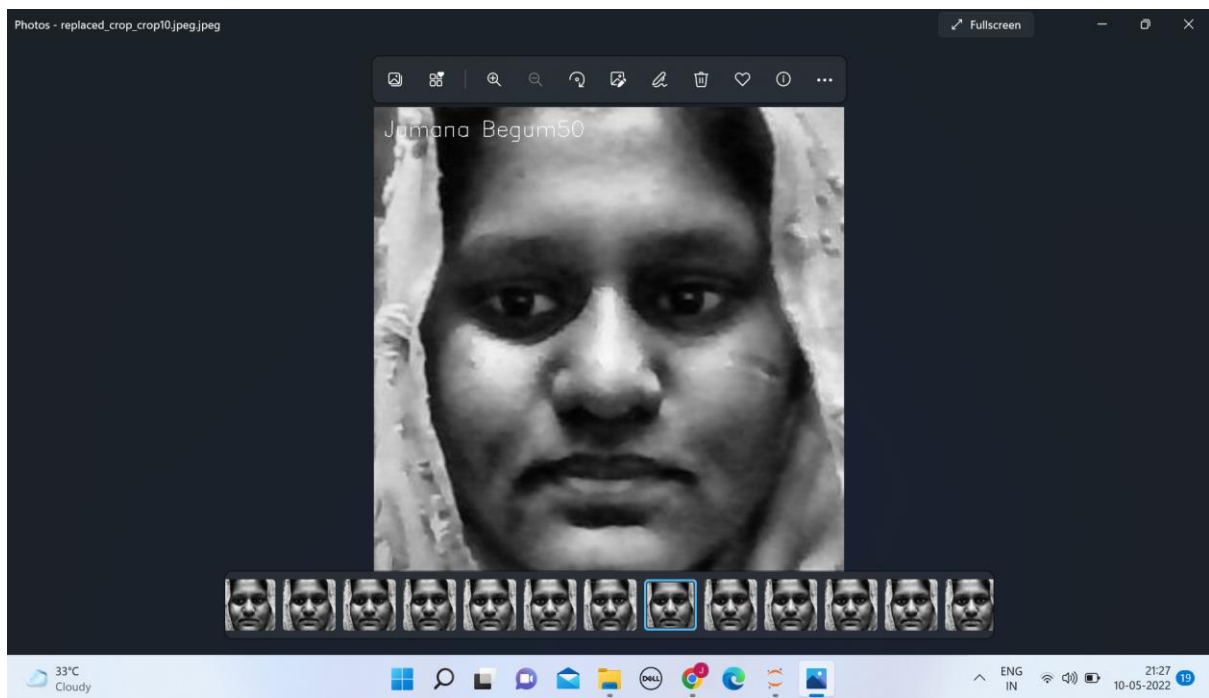


Figure 6.11 Image with Label and Score

ATTENDANCE UPDATION

Driver Program (contd):

```
for(x,y,w,h) in faces1:
    if score >= 30 :
        cv2.imwrite("CIP/INTRUDERS/intruder"+str(i)+".jpeg",image)
        i += 1
    attendanceupdate(label_text,score)
for(x,y,w,h) in faces2:
    if score > =30 :
        cv2.imwrite("CIP/INTRUDERS/intruder"+str(i)+".jpeg",image)
        i += 1
    attendanceupdate(label_text,score)

cv2.imshow('Frame',frame)
if cv2.waitKey(1000) & 0xFF == ord('q'):
    break
else:
    break
cap.release()
cv2.destroyAllWindows()
sendmail()
sendmailintruder()
```

FUNCTION TO CREATE A NEW ATTENDANCE SHEET FOR EACH DAY

```
def createfile():
```

```
    i = 0
```

```
    students={'2019103019' : ['Harini','harinisasi26@gmail.com'] ,
```

```
              '2019103022' : ['Jumana Begum','begumjumana2712@gmail.com'],
```

```
              '2019103026' : ['Karthika','karthikat008@gmail.com'],
```

```
              '2019103030' : ['Kirupa','kirupashrir2001@gmail.com'],
```

```
              '2019103039' : ['Narmatha','narmathama23@gmail.com'],
```

```
              '2019103047' : ['Preethi','preethiba@gmail.com'],
```

```
              '2019103049' : ['Ramya','ramya26@gmail.com'],
```

```
              '2019103055' : ['Samreen Ayesha','samreenayesha137@gmail.com'],
```

```
              '2019103063' : ['Sineka','sinekapkt@gmail.com']}]
```

```
    header = ['Name','Register No','Email ID','Time of entry','Status']
```

```
    today = date.today()
```

```
    current_date = today.strftime("%b-%d-%Y")
```

```
    f = open(current_date+'.csv', 'w',newline=")
```

```
    writer = csv.writer(f)
```

```
    writer.writerow(header)
```

```
    Empt= "
```

```
    abse = 'Absent'
```

```
    for key in students.keys():
```

```
        f.write("%s,%s,%s,%s,%s\n" % (students[key][0], key,students[key][1],Empt,abse))
```

```
        i += 1
```

```
    f.close()
```

FUNCTION TO UPDATE ATTENDANCE SHEET

```
def attendanceupdate(label_text,score):

    today = date.today()

    current_date = today.strftime("%b-%d-%Y")

    df = pd.read_csv(current_date+'.csv')

    csv_file = csv.reader(open(current_date+'.csv', "r"), delimiter=",")

    if score <= 20:

        i = -2

        for row in csv_file:

            i += 1

            if label_text==row[0]:

                now = datetime.now()

                nowstr = now.strftime("%H:%M:%S")

                if now < today830am and row[4]=="Absent":

                    df.loc[i, "Status"] = 'Present'

                    df.loc[i,"Time of entry"] = nowstr

                elif now < today900am and row[4]=="Absent":

                    df.loc[i, "Status"] = 'Late'

                    df.loc[i,"Time of entry"] = nowstr

                df.to_csv(current_date+'.csv', index=False)
```

FUNCTION TO SEND EMAIL TO THE PARENT INCASE OF ABSENT OR LATE STUDENT

```
def sendmail():

    today = date.today()
```

```

current_date = today.strftime("%b-%d-%Y")

csv_file = csv.reader(open(current_date+'.csv', "r"), delimiter=",")

mailabs = []

maillate = []

for row in csv_file:

    if row[4]=="Absent":

        mailabs.append(row[2])

    elif row[4]=="Late":

        maillate.append(row[2])


s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

s.login("projectmail2712@gmail.com", "*****")

for dest in mailabs:

    message = "Your ward was absent today."

    s.sendmail("projectmail2712@gmail.com",dest,message)

for dest in maillate:

    message = "Your ward was late today."

    s.sendmail("projectmail2712@gmail.com",dest,message)

s.quit()

```

FUNCTION TO SEND EMAIL TO THE TEACHER WHEN INTRUDERS ARE IDENTIFIED

```

def sendmailintruder():

    import smtplib

```

```
from email.mime.multipart import MIMEMultipart
```

```
from email.mime.text import MIMEText
```

```
from email.mime.base import MIMEBase
```

```
from email import encoders
```

```
fromaddr = "projectmail2712@gmail.com"
```

```
toaddr = "begumjumana2712@gmail.com"
```

```
msg = MIMEMultipart()
```

```
msg['From'] = "projectmail2712@gmail.com"
```

```
msg['To'] = "begumjumana2712@gmail.com"
```

```
msg['Subject'] = "INTRUDER LIST"
```

```
body = "IMAGES OF THE INTRUDER"
```

```
msg.attach(MIMEText(body, 'plain'))
```

```
#looping through the intruders images
```

```
path = "C:/Users/GCS/CIP/INTRUDERS"
```

```
dir_list = os.listdir(path)
```

```
for elt in dir_list:
```

```
    filename = elt
```

```
    impath = path+'/'+elt
```

```
    attachment = open(impath, "rb")
```

```
    p = MIMEBase('application', 'octet-stream')
```

```

p.set_payload((attachment).read())

encoders.encode_base64(p)

p.add_header('Content-Disposition', "attachment; filename= %s" % filename)

msg.attach(p)


#sending email to teacher regarding the intruders

s = smtplib.SMTP('smtp.gmail.com', 587)

s.starttls()

s.login("projectmail2712@gmail.com", "*****")

text = msg.as_string()

s.sendmail(fromaddr, toaddr, text)

s.quit()

```

OUTPUT:

A new csv file that is created

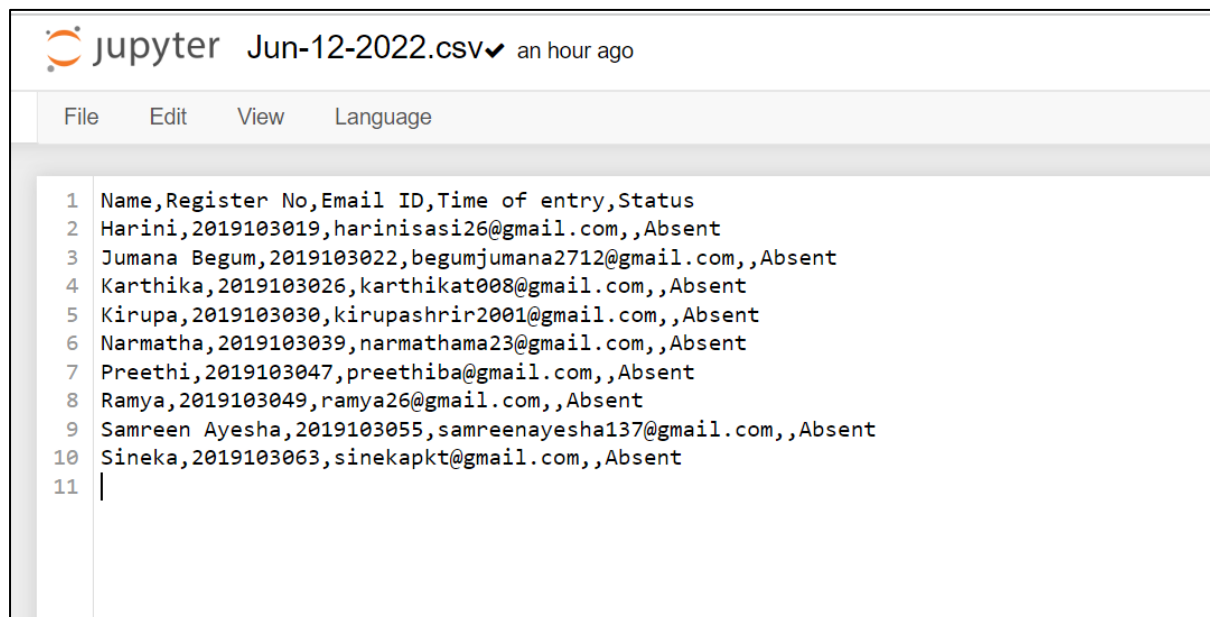
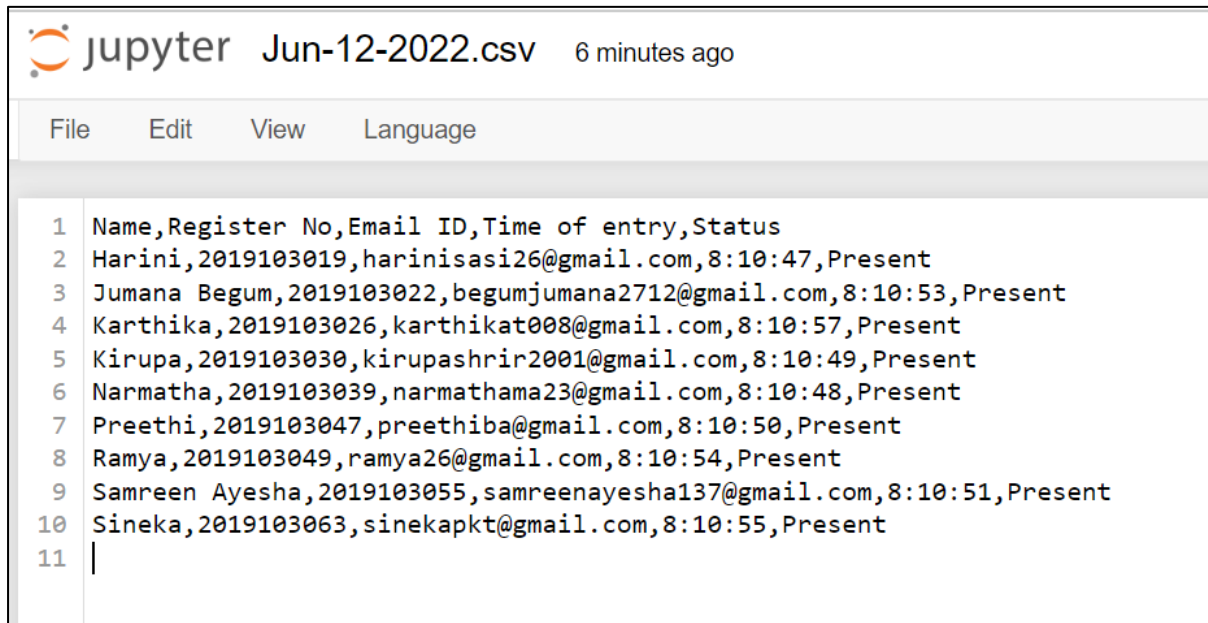


Figure 6.12 CSV File creation

The updated attendance sheet with the Status and the Time of entry entered for the recognized faces



```

jupyter Jun-12-2022.csv 6 minutes ago
File Edit View Language

1 Name,Register No,Email ID,Time of entry,Status
2 Harini,2019103019,harinisasi26@gmail.com,8:10:47,Present
3 Jumana Begum,2019103022,begumjumana2712@gmail.com,8:10:53,Present
4 Karthika,2019103026,karthikat008@gmail.com,8:10:57,Present
5 Kirupa,2019103030,kirupashrir2001@gmail.com,8:10:49,Present
6 Narmatha,2019103039,narmathama23@gmail.com,8:10:48,Present
7 Preethi,2019103047,preethiba@gmail.com,8:10:50,Present
8 Ramya,2019103049,ramya26@gmail.com,8:10:54,Present
9 Samreen Ayesha,2019103055,samreenayasha137@gmail.com,8:10:51,Present
10 Sineka,2019103063,sinekapkt@gmail.com,8:10:55,Present
11 |
  
```

Figure 6.13 CSV File updation

The intruders identified are stored in a Intruders folder

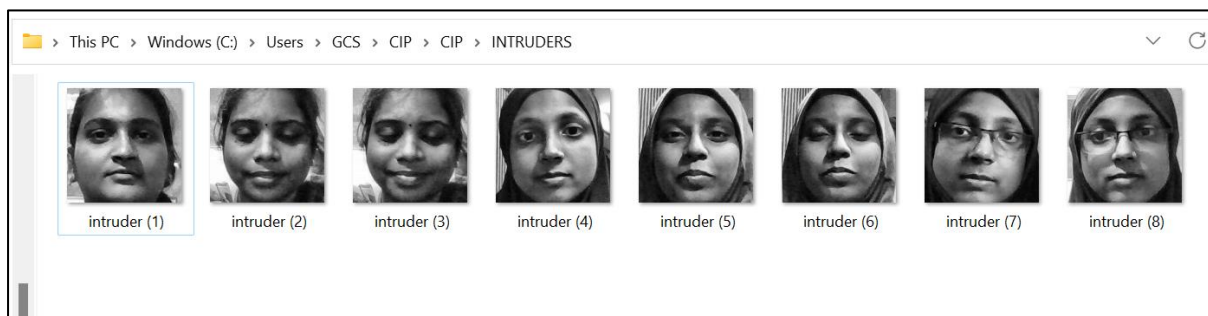


Figure 6.14 Intruders Identified

Mail sent to Parent/Guardian



Figure 6.15 Mail sent to the Parent/Gaurdian for late and absent students

Mail sent to Teacher

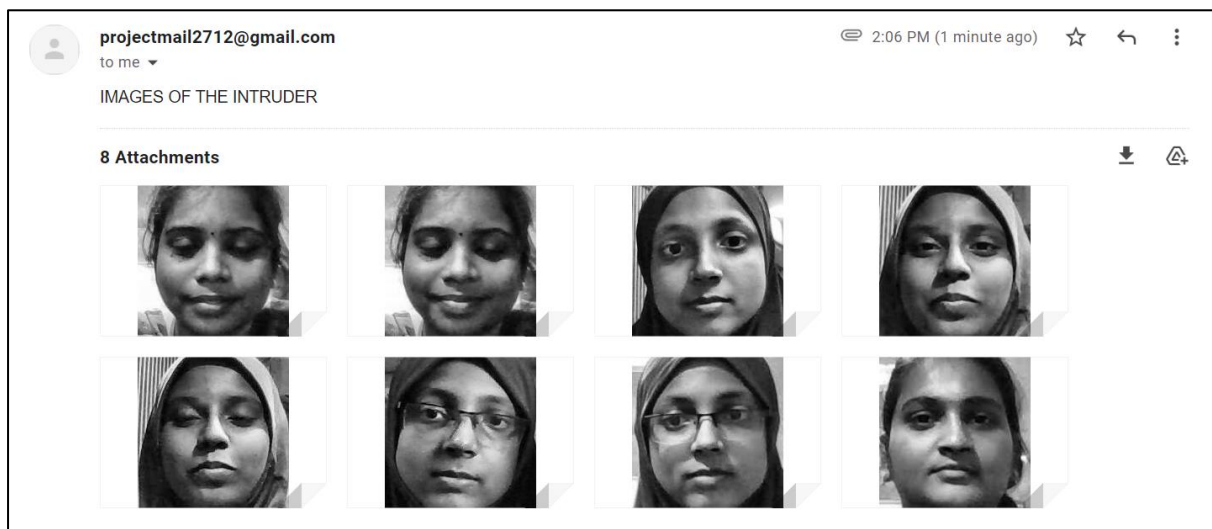


Figure 6.16 Mail sent to Teacher regarding the intruders

7. RESULT ANALYSIS

PERFORMANCE METRICS USED:

➤ ACCURACY

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

➤ PRECISION

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

➤ RECALL

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

➤ F-SCORE

$$F_1\text{-score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN}$$

PERFORMANCE METRICS:

Table 7.1 Performance Metrics

MODEL TYPE	ACCURACY	PRECISION	RECALL	F ₁ -SCORE
Without CLAHE and without blurred images handling	28.13%	31.48%	20.71%	17.74%
With CLAHE and without	75%	93.33%	79.52%	81.94%

blurred images handling				
Without CLAHE and with blurred images handling	81.48%	84.44%	82%	79.49%
With CLAHE and with blurred images handling (implemented model)	87.5%	95%	90.71%	90.61%

OUTPUT:

Without CLAHE and without blurred images handling

```

Accuracy Score: 0.28125
Precision Score: 0.31481481481481477
Recall Score: 0.20714285714285713
F1 Score: 0.17747311827956988
Confusion Matrix:
[[0 0 0 0 0 0 0 0 0 2]
 [0 3 0 0 0 0 0 0 0 4]
 [0 0 0 0 0 0 0 0 0 2]
 [0 0 0 0 0 0 0 0 0 2]
 [0 0 0 0 0 0 0 0 0 2]
 [0 0 0 0 0 1 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 3]
 [0 0 0 0 0 0 0 1 0 6]
 [0 0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0 4]]

```

Figure 7.1 Metrics for Model 1

With CLAHE and without blurred images handling

```
Accuracy Score: 0.75
Precision Score: 0.9333333333333333
Recall Score: 0.7952380952380953
F1 Score: 0.8193939393939393
Confusion Matrix:
[[1 0 0 0 0 0 0 0 0 0 1]
 [0 5 0 0 0 0 0 0 0 0 2]
 [0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0]
 [0 0 0 0 2 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 1]
 [0 0 0 0 0 0 2 0 0 0 1]
 [0 0 0 0 0 0 0 4 0 0 3]
 [0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 4]]
```

Figure 7.2 Metrics for Model 2

Without CLAHE and with blurred images handling

```
Accuracy Score: 0.8148148148148148
Precision Score: 0.8444444444444444
Recall Score: 0.82
F1 Score: 0.7948717948717949
Confusion Matrix:
[[0 0 0 0 0 0 0 0 0 0 1]
 [0 6 0 0 0 0 0 0 0 0 0]
 [0 0 2 0 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 0 3 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0 4]
 [0 0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0 4]]
```

Figure 7.3 Metrics for Model 3

With CLAHE and with blurred images handling (implemented model)

```

Accuracy Score: 0.875
Precision Score: 0.95
Recall Score: 0.9071428571428571
F1 Score: 0.9060606060606060
Confusion Matrix:
[[1 0 0 0 0 0 0 0 0 0 1]
 [0 7 0 0 0 0 0 0 0 0]
 [0 0 2 0 0 0 0 0 0 0]
 [0 0 0 2 0 0 0 0 0 0]
 [0 0 0 0 2 0 0 0 0 0]
 [0 0 0 0 0 2 0 0 0 0]
 [0 0 0 0 0 0 3 0 0 0]
 [0 0 0 0 0 0 0 4 0 3]
 [0 0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0 4]]

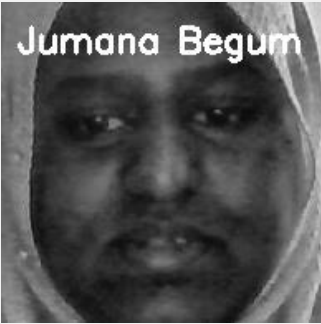

```

Figure 7.4 Metrics for Model 4(Implemented Model)

TEST CASES AND VALIDATION

Table 7.2 Test Cases

Test Case	Validation
Front Profile Face	
Side Profile Face	

Face with Scarf	
Face with spectacles	

8. CONCLUSION AND FUTURE WORK

LBPH is one of the prominent techniques for face recognition. Our system successfully recognizes a student with their front face, side face and even after unintentional changes like wearing glasses. Here the problem is that the dataset is small. In future, an effort could be made to build a better dataset, that might practically give a more accurate result. With current corona pandemic situation with people wearing masks, in future a model could be built to recognize people with their masks on. A system alert (voice and visual) can be included if an intruder is detected in the class.

REFERENCES

[1] Visar Shehu, Agni Dika, “Using Real Time Computer Vision Algorithms in Automatic Attendance Management Systems”, Proceedings of the ITI 2010

[2] Nirmalya Kar, Mrinal Kanti Debbarma, Ashim Saha, Dwijen Rudra Pal, "Study of Implementing Automated Attendance System Using Face Recognition Technique", International Journal of Computer and Communication Engineering 2012

[3] Jomon Joseph, K. P. Zacharia, "Automatic Attendance Management System Using Face Recognition", International Journal of Science and Research (IJSR), Volume 2 Issue 11, November 2013, 327 - 330

[4] Rekha, E. & Poojary, Ramaprasad, "An efficient automated attendance management system based on Eigen Face recognition" 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence (Confluence)

[5] Priyanka Wagh, Jagruti Chaudhari, Roshani Thakare, Shweta Patil "Attendance System based on Face Recognition using Eigen face and PCA Algorithms", 2015 International Conference on Green Computing and Internet of Things (ICGCIoT)

[6] Smitha, Pavithra S Hegde, Afshin, "Face Recognition based Attendance Management System", June 2020 International Journal of Engineering and Technical Research

[7] Nandhini R, Duraimurugan N, S.P.Chokkalingam, "Face Recognition Based Attendance System", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-3S, February 2019