Software Engineering Mini Project Report
(Smart Attendance System with Face Recognition using OpenCV)

Shumbul Arifa (181CO152), Keerti Chaudhary (181CO226)

Students of the Department of Computer Science and Engineering, National Institute of Technology Karnataka Surathkal, Karnataka, India

shumbul.181co152@nitk.edu.in , keerti2001.kc@gmail.com

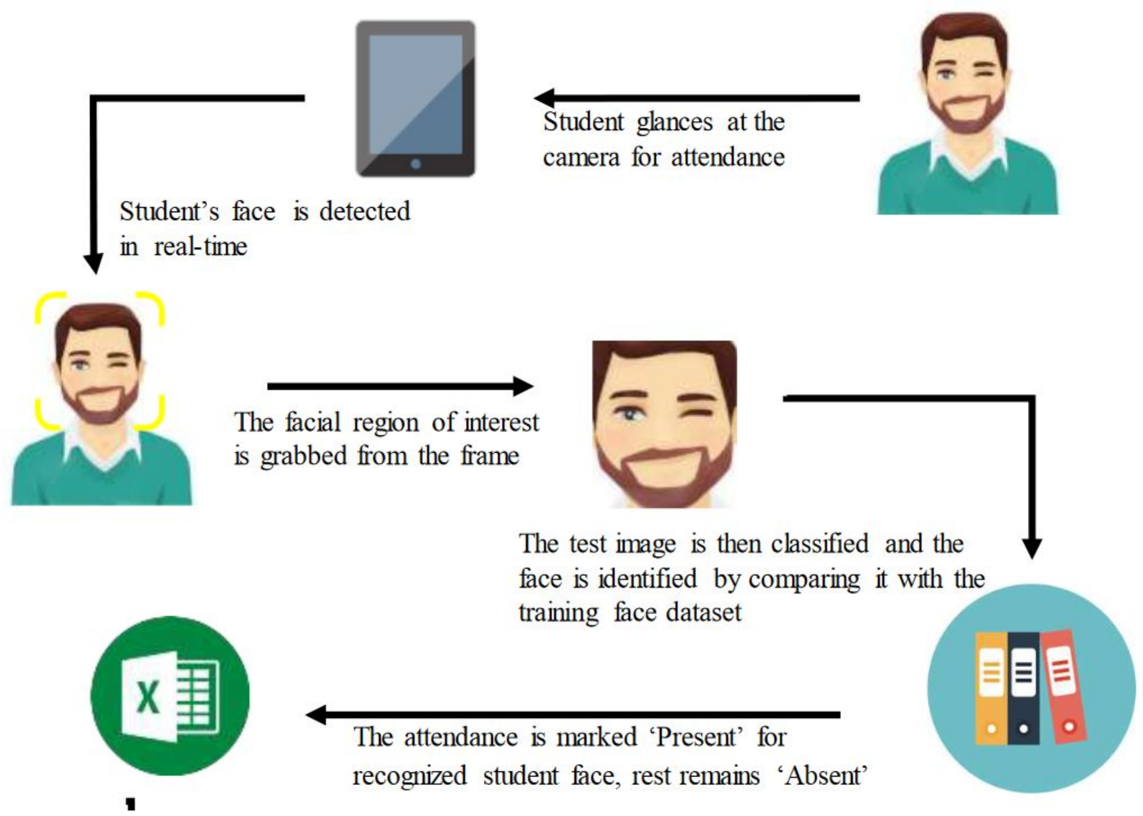
Introduction

This paper focuses on the implementation of the proposed system, ie. the smart attendance system using OpenCV. It also includes how it has been developed iteratively and has been followed through by structural modeling with an architectural design of the requirements captured during the requirements analysis. Also, a detailed description of the functionalities implemented with stages of how the prototype has evolved to completion is discussed. The traditional student attendance marking technique is often facing a lot of trouble.

The face recognition student attendance system emphasizes its simplicity by eliminating classical student attendance marking techniques such as calling student names or checking respective identification cards. Hence, there is a need to develop a real-time operating student attendance system which means the identification process must be done within defined time constraints to prevent omission. The extracted features from facial images that represent the identity of the students have to be consistent towards a change in background, illumination, pose, and expression. High accuracy and fast computation time will be the evaluation points of the performance.

Overview of System

Face recognition includes isolating picture windows into two classes; one containing faces turning the background (clutter). It is troublesome because although shared characteristics exist between faces, they can change significantly as far as age, skin tone, and outward appearance. The issue is additionally complicated by varying lighting conditions, image characteristics, and calculations, as well as the possibility of partial occlusion and disguise. An ideal face recognizer would hence have the option to distinguish the presence of any face under any arrangement of lighting conditions, upon any background. The face detection errand can be separated into two stages. The initial step is a grouping task that accepts some discretionary picture as information and yields a parallel estimation of yes or no, showing whether there are any faces present in the picture. The subsequent advance is the face limitation task that plans to accept a picture as information and yield the area of any face or faces inside that picture as some jumping box with (x, y, width, height). After snapping the photo the framework will look at the fairness of the photos in its data set and give the most related outcome.


Student glances at the camera for attendance
Student's face is detected in real-time
The facial region of interest is grabbed from the frame
The test image is then classified and the face is identified by comparing it with the training face dataset
The attendance is marked 'Present' for recognized student face, rest remains 'Absent'

Interface Design Objectives

In this project, sensible software is developed that is capable of recognizing the identity of every individual and eventually record down the info into a database system. Apart from that, an interface will be developed to supply visual access to the data. The objective of this project is to develop a Smart Attendance System with Face Recognition. The achievements to fulfill the objectives are:

• To detect the face segment from the moving video frame.
• To extract useful features from the face detected.
• To classify the features to recognize the face detected.
• To mark the attendance of the identified student.

Software System and Image Processing

The purpose of this module is to receive the incoming information, retrieve data from the database, and compare two pictures. To be efficient the image processing will be implemented and tested in parallel with the other software parts. Once every part is working, the integration phase will be done and testing will be conducted.

1. Face detection:

- The algorithm will be tested to detect the face on preloaded images and an image taken with a webcam.
- Different people will take part in the tests and different positions & lighting conditions will make the test cases.
- The algorithm will handle one face per picture as discussed in our requirement specification.

2. Face recognition

- Face recognition is related to the previous module. Once the face is detected the system will try and identify the person.
- A pre-loaded picture will be stored in the computer and the algorithm should be able to identify the person.
- Different persons will take part in the test and this iteration is repeated to identify different persons.

3. The software solution –Database and information retrieval

- A database will be created and filled with some random information for testing purposes. The first testing step after that is to make sure that the connection between the program and the database is secure and reliable.
- Once the connection is made, a random number with the same format as the tag number will be used to access the database.
- The second step will be to store an image in the database and to retrieve it using the same number.

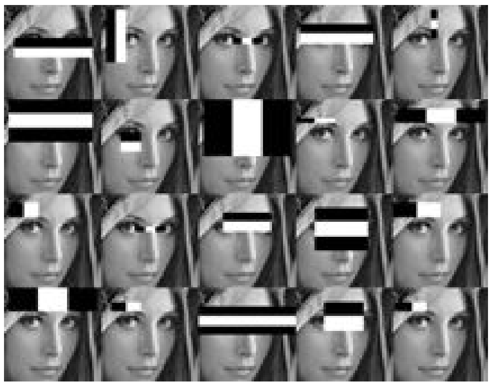
4. The software solution –Add image processing

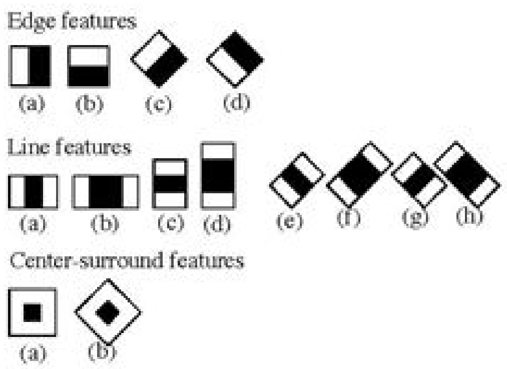
- After retrieving information, a person will try to be identified. A wrong person and the real person will take part in the test.

Implementation

Firstly the image is imported by giving the area of the picture. At that point, the image is changed from RGB to Grayscale because it is anything but difficult to distinguish faces in the grayscale. From that point forward, picture control is utilized, in which the resizing, editing, obscuring, and honing of the pictures is done if necessary. The following stage is picture division, which is utilized for form location or sections the different items in a solitary picture so the classifier can rapidly distinguish the articles and faces in the image.

The following step is to use an algorithm. The calculation is used for finding the area of the human countenances in a casing or picture. All human faces share some all-inclusive properties of the human face like the eyes area is more obscure than its neighbor pixels and the nose district is more splendid than the eye locale.





a. Working on face detection b. Edge, line, and central features

The haar-like calculation is likewise utilized for highlight determination or highlight extraction for an article in a picture, with the assistance of edge location, line identification, focus detection for recognizing eyes, nose, mouth, and so forth in the image. It is utilized to choose the fundamental highlights in a picture and concentrate these highlights for face detection.

The subsequent stage is to give the directions of x, y, w, h which makes a square shape enclose the image to show the area of the face or we can say that to show the locale of interest in the picture. After this, it can make a square shape enclose the zone of interest where it identifies the face. There are additionally numerous other identification strategies that are utilized together for recognition, for example, grin detection, eye location, squint detection, and so on.

We used some tools to build our system. Without the help of these tools, it would not be possible to make it done. Here we will discuss the most important one.

Softwares required for implementation:

1. OpenCV:

We used OpenCV 3 dependency for python 3. OpenCV is a library where there are lots of image processing functions available. This is a very useful library for image processing. Even one can get the expected outcome without writing a single code. The library is cross-platform and free for use under the open-source BSD license. Example of some supported functions are given below:

- Derivation: Gradient/laplacian computing, contours delimitation
- Hough transforms: lines, segments, circles, and geometrical shapes detection
- Histograms: computing, equalization, and object localization with a back-projection algorithm
- Segmentation: thresholding, distance transform, foreground/background detection, watershed segmentation
- Filtering: linear and nonlinear filters, morphological operations
- Cascade detectors: detection of a face, eye, car plates
- Interest points: detection and matching
- Video processing: optical flow, background subtraction, camshaft (object tracking)
- Photography: panoramas realization, high definition imaging (HDR), image inpainting

Instructions to install OpenCV:

```
1. sudo chmod 755 /file_name/pi/installopencv.bash

2. sudo /myfile/pi/installopencv.bash
```

2. Python IDE:

There are loads of IDEs for python. Some of them are PyCharm, VSCode, Thonny, Ninja, Spyder, and so on Ninja and Spyder both are amazing and free however we utilized Spyder as it was more component rich than a ninja. Spyder is somewhat heavier than ninja yet at the same time a lot lighter than PyCharm. You can run them in pi and get GUI on your PC through ssh-Y. We introduced Spyder through the order line underneath.

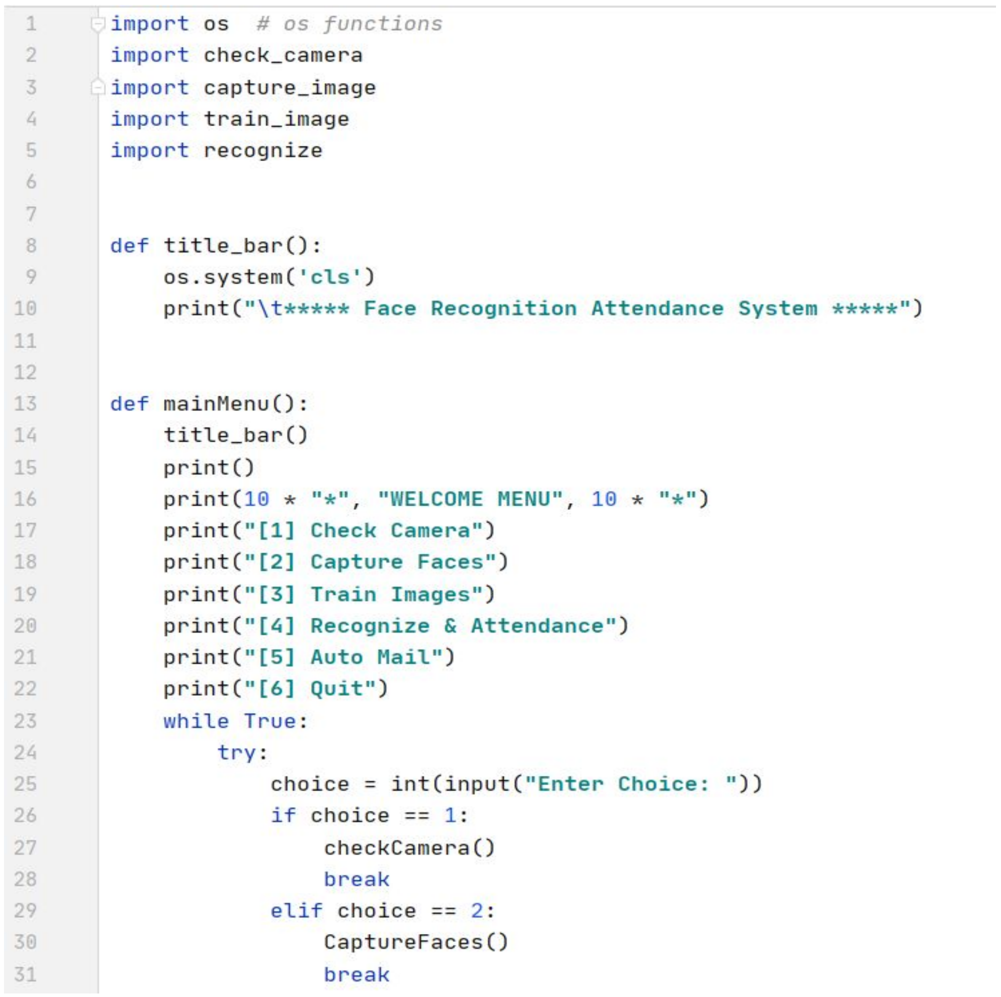
```
1. sudo apt-get install spyder
```

All our code is written in Python language. Below are the important python codes which are the core of our implementation of this software engineering project.

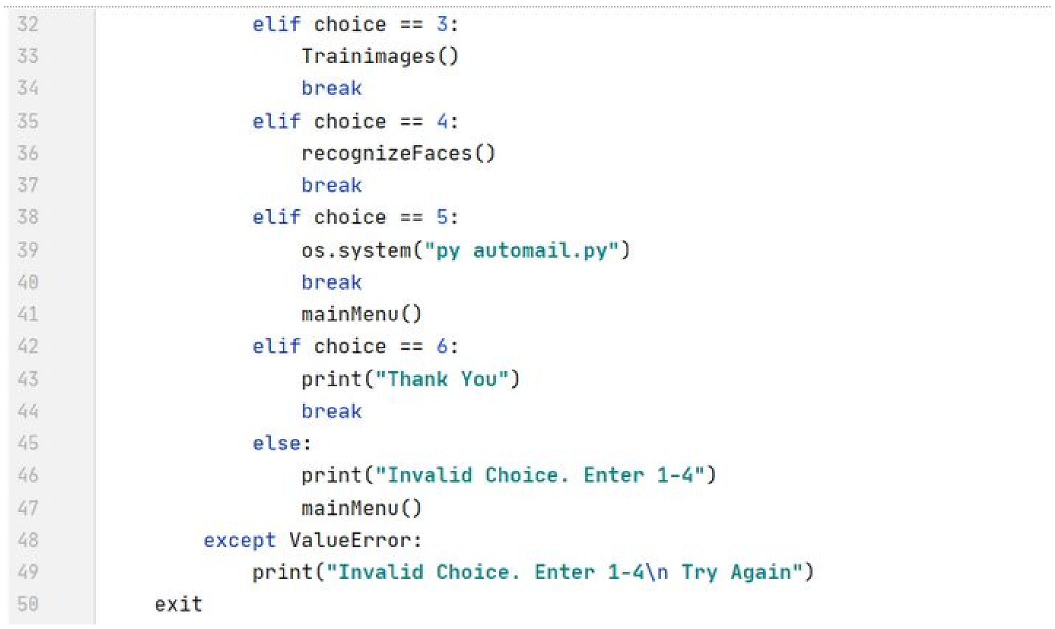
1. Dataset: Where all the faces are saved.
2. main.py: Main program file to run the program.
3. train_image.py: Train the captured images and work on datasets.
4. recognize.py: To recognize and mark attendance
5. automail.py: To send mail to a specific mail id.

1. main.py

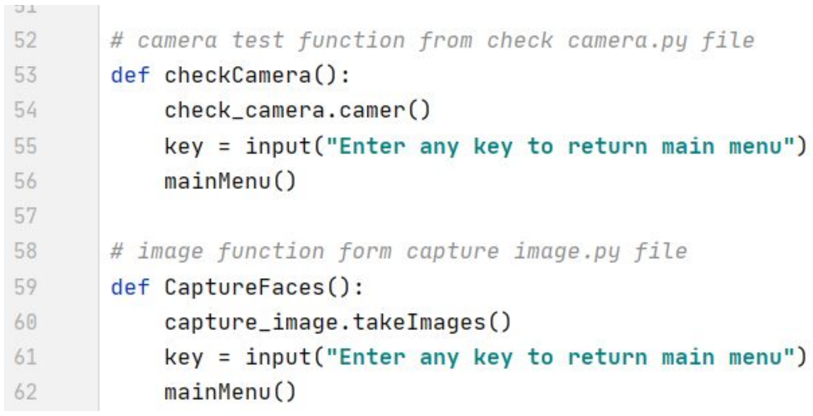
This is the main python code for our project that includes functions from other files and has all the options which are allowed in the user interface. Once this file is run, the user gets to choose whatever action has to be taken place, say whether it is to capture an image or to train an image, etc. The main function calls the other functions and specifies options to the user.



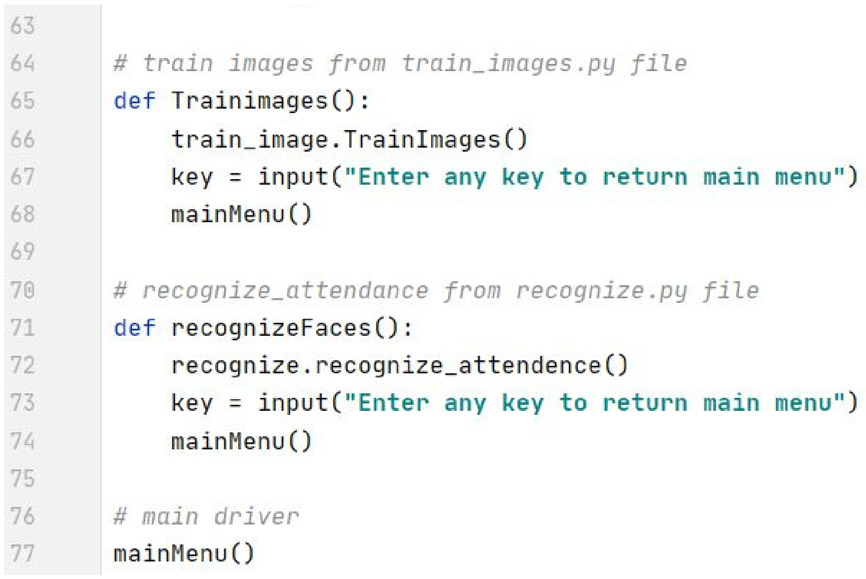
```python
import os  # os functions
import check_camera
import capture_image
import train_image
import recognize


def title_bar():
    os.system('cls')
    print("\t***** Face Recognition Attendance System *****")


def mainMenu():
    title_bar()
    print()
    print(10 * "*", "WELCOME MENU", 10 * "*")
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance")
    print("[5] Auto Mail")
    print("[6] Quit")
    while True:
        try:
            choice = int(input("Enter Choice: "))
            if choice == 1:
                checkCamera()
                break
            elif choice == 2:
                CaptureFaces()
                break
```



```python
            elif choice == 3:
                Trainimages()
                break
            elif choice == 4:
                recognizeFaces()
                break
            elif choice == 5:
                os.system("py automail.py")
                break
                mainMenu()
            elif choice == 6:
                print("Thank You")
                break
            else:
                print("Invalid Choice. Enter 1-4")
                mainMenu()
        except ValueError:
            print("Invalid Choice. Enter 1-4\n Try Again")
    exit
```



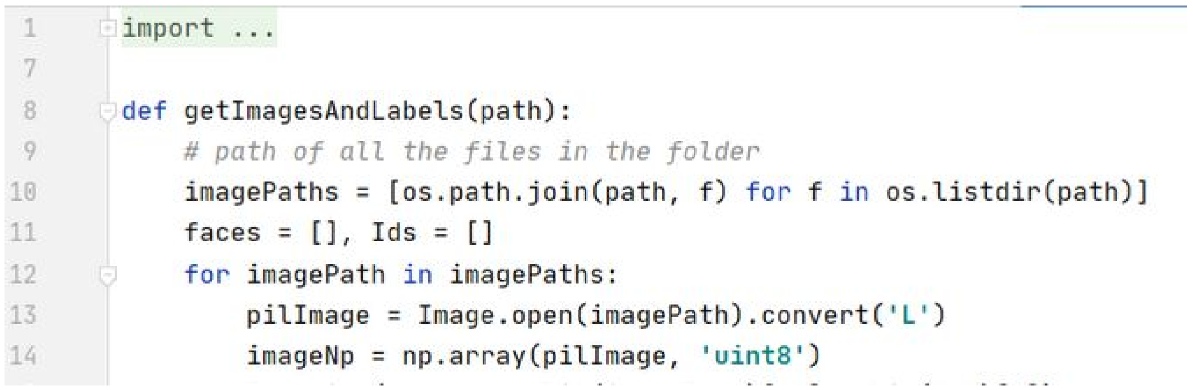
```python
# camera test function from check camera.py file
def checkCamera():
    check_camera.camer()
    key = input("Enter any key to return main menu")
    mainMenu()

# image function form capture image.py file
def CaptureFaces():
    capture_image.takeImages()
    key = input("Enter any key to return main menu")
    mainMenu()
```



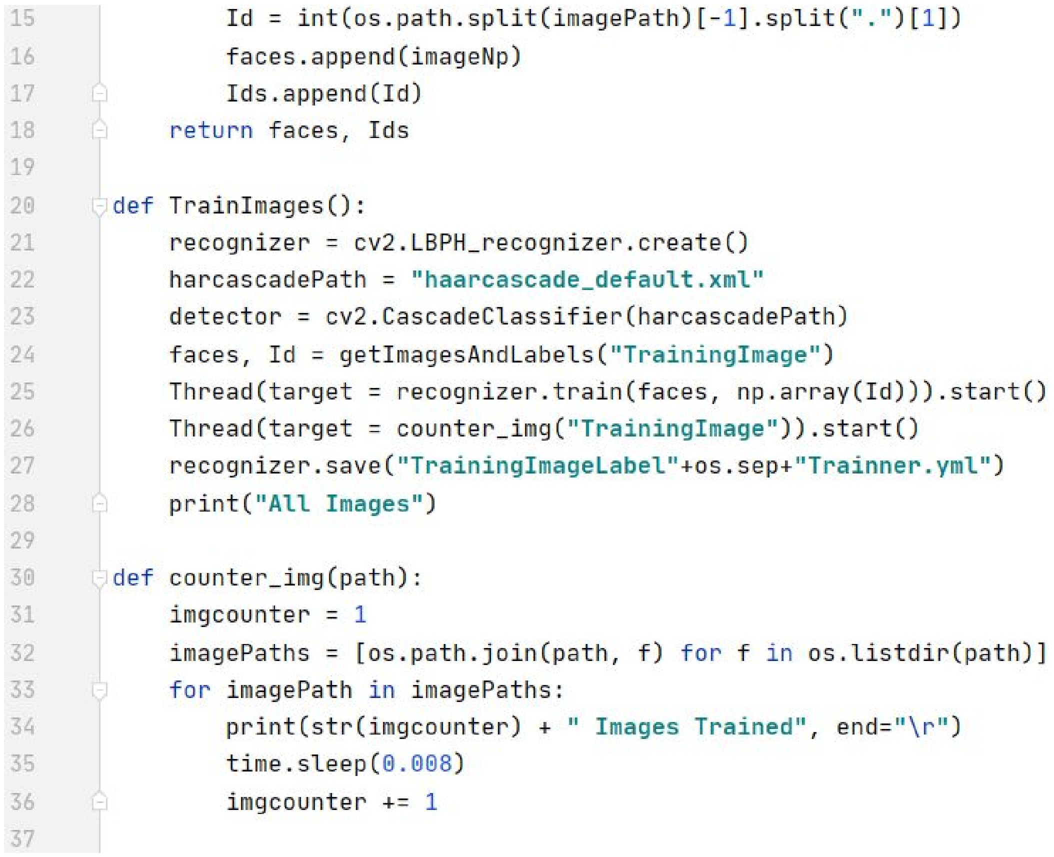
```python

# train images from train_images.py file
def Trainimages():
    train_image.TrainImages()
    key = input("Enter any key to return main menu")
    mainMenu()

# recognize_attendance from recognize.py file
def recognizeFaces():
    recognize.recognize_attendence()
    key = input("Enter any key to return main menu")
    mainMenu()

# main driver
mainMenu()
```

2.train_image.py

In this step, the image gets trained with database images and machine learning techniques. This is done by making use of a haar cascade classifier. Once the images are trained and the model is ready for the next step, the “image trained“ message appears on the screen.



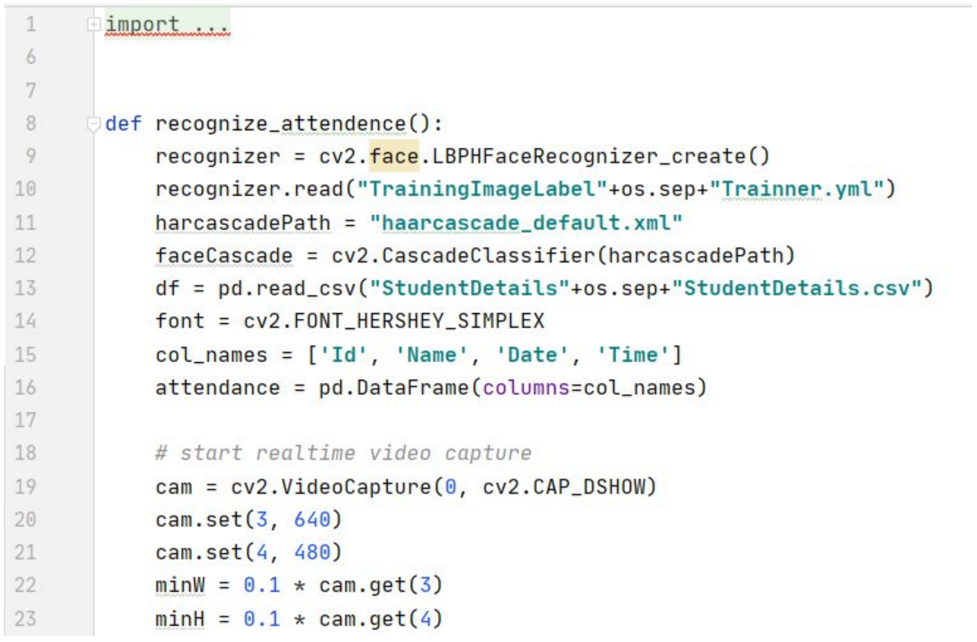
```python
import ...

def getImagesAndLabels(path):
    # path of all the files in the folder
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    faces = [], Ids = []
    for imagePath in imagePaths:
        pilImage = Image.open(imagePath).convert('L')
        imageNp = np.array(pilImage, 'uint8')
```



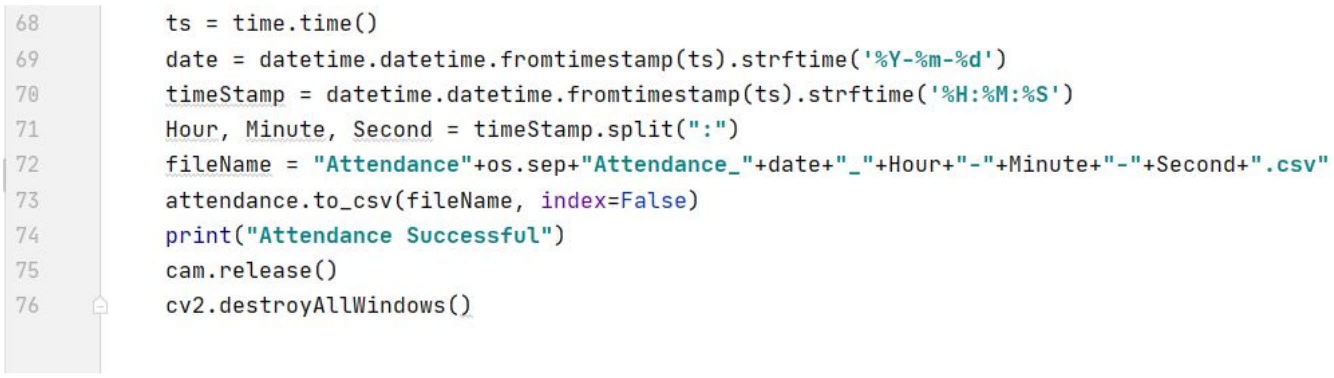
```python
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids

def TrainImages():
    recognizer = cv2.LBPH_recognizer.create()
    harcascadePath = "haarcascade_default.xml"
    detector = cv2.CascadeClassifier(harcascadePath)
    faces, Id = getImagesAndLabels("TrainingImage")
    Thread(target = recognizer.train(faces, np.array(Id))).start()
    Thread(target = counter_img("TrainingImage")).start()
    recognizer.save("TrainingImageLabel"+os.sep+"Trainner.yml")
    print("All Images")

def counter_img(path):
    imgcounter = 1
    imagePaths = [os.path.join(path, f) for f in os.listdir(path)]
    for imagePath in imagePaths:
        print(str(imgcounter) + " Images Trained", end="\r")
        time.sleep(0.008)
        imgcounter += 1

```

3. recognize.py

In this step, the faces are recognized with the help of the LBPH Face recognizer function. Along with this, the time and date also get recorded. If the function works successfully then the attendance is marked for the recognized face.



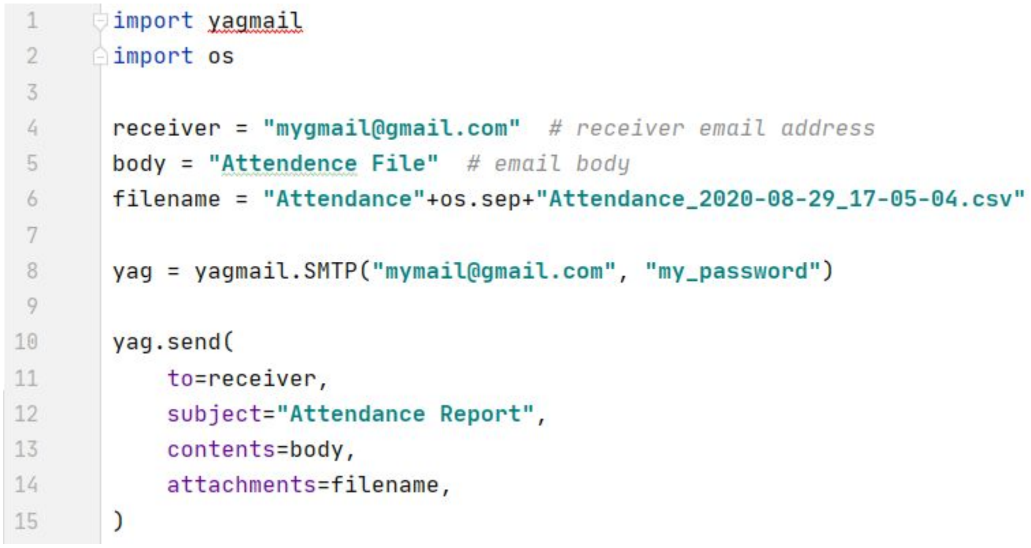
```python
import ...


def recognize_attendence():
    recognizer = cv2.face.LBPHFaceRecognizer_create()
    recognizer.read("TrainingImageLabel"+os.sep+"Trainner.yml")
    harcascadePath = "haarcascade_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)
    df = pd.read_csv("StudentDetails"+os.sep+"StudentDetails.csv")
    font = cv2.FONT_HERSHEY_SIMPLEX
    col_names = ['Id', 'Name', 'Date', 'Time']
    attendance = pd.DataFrame(columns=col_names)

    # start realtime video capture
    cam = cv2.VideoCapture(0, cv2.CAP_DSHOW)
    cam.set(3, 640)
    cam.set(4, 480)
    minW = 0.1 * cam.get(3)
    minH = 0.1 * cam.get(4)
```



```python
    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"+os.sep+"Attendance_"+date+"_"+Hour+"-"+Minute+"-"+Second+".csv"
    attendance.to_csv(fileName, index=False)
    print("Attendance Successful")
    cam.release()
    cv2.destroyAllWindows()
```

4. automail.py

In this project, we have added an extra feature called auto-mail. It automatically sends the attendance file to specific mail (specified in the system model). The code for the same is given below:



```python
import yagmail
import os

receiver = "mygmail@gmail.com"  # receiver email address
body = "Attendence File"  # email body
filename = "Attendance"+os.sep+"Attendance_2020-08-29_17-05-04.csv"

yag = yagmail.SMTP("mymail@gmail.com", "my_password")

yag.send(
    to=receiver,
    subject="Attendance Report",
    contents=body,
    attachments=filename,
)
```

Software Testing

Testing is the technique for assessing a framework or its parts to search out whether it fulfills the ideal requirements or not. In direct words, testing is executing a framework to recognize any holes, mistakes, or missing requirements. It is a fundamental advance in a product designing cycle to develop any product. Software Testing is fundamental since we as a whole commit errors. A portion of those errors are irrelevant, however, some of them are costly. We have to check everything and anything we produce since things can generally turn out wrong.

Black box testing

In black-box testing, the structure of the program isn't thought about. It considers the usefulness of the application as it was. It is likewise called practical testing. In this sort of testing, the analyzers focus on utilitarian testing, that is, on giving known information and checking if the realized yield is acquired.

This strategy is by and large followed while completing acknowledgment testing when the end-client isn't a product engineer however just a client.

White-box testing

White-box Testing is a kind of testing technique where the tester (a developer testing the application) knows about the framework internals. Its usage is straightforward. The goal is to guarantee that each line of the code is tried. The tester recognizes all consistent, plan, and typographical mistakes. The tester likewise needs to approve the interior structure of the thing viable alongside the yield.

CASE Tools

Computer-Aided Software Engineering (CASE) is the computer computerization of framework advancement exercises. These can be arranged into 3 classes: front-end, back-end, or cross life cycle CASE items, as indicated by when they are utilized during frameworks improvement. Front-end apparatuses now and again called capitalized devices, are utilized during primer examination, venture arranging, investigation, or plan. Back-end instruments, here and there called lower-CASE devices, are utilized during advancement, execution, assessment, and upkeep. Cross life cycle CASE apparatuses are utilized to help progressing exercises, for example, venturing the board and making documentation, which happens over numerous periods of framework improvement.

CASE apparatuses can likewise be free or coordinated. Framework documentation is regularly done utilizing CASE instruments that are autonomous of one another, to computerize explicit advancement exercises, for example, information stream outlines or ER diagrams. In any case, these individual apparatuses may not have similar organizations for sharing data. Any progressions to the framework imply that the tedious cycle of refreshing the documentation is frequently ignored. Future upkeep of the framework at that point turns out to be incredibly troublesome. Interestingly, incorporated CASE climate centers around the upkeep of configuration records, and CASE devices are utilized to deal with exercises all through the whole framework advancement. Incorporated CASE devices utilize a typical vault with the goal that data can be shared over all instruments and framework improvement exercises. Any progressions to the plan records typically bring about the most different parts of the framework mirroring those changes.

Requirement Traceability Matrix

A traceability matrix is a document that associates any two-gauge records that require a many-to-numerous relationship to check the fulfillment of the relationship. It is utilized to follow the prerequisites and to check the current venture necessities are met.
Requirement Traceability Matrix, RTM catches all necessities proposed by the customer client and advancement group client and their detectability in a solitary archive conveyed after the life-cycle.In different words, it is a report that guides and follows client necessities with experiments. The fundamental reason for the Requirement Traceability Matrix is to see that all experiments are covered so no usefulness should miss while testing.

Table1: Functional Requirements Tables

Functional Requirement - ID	Functional Requirements Description
FR-1	Face Detection
FR-2	Capturing Faces and Training Model
FR-3	Storing images in the database
FR-4	Face Recognition
FR-5	Recording Attendance
FR-6	Emailing Attendance

Table2: Test Cases

Test Case - ID	Test Case Description
TC-1	Web camera detecting faces live
TC-2	Capturing multiple images
TC-3	Storing images with student details
TC-4	Encoding faces
TC-5	Training model with training images
TC-6	Recognizing faces registered in the database
TC-7	Marking attendance along with time
TC-8	Emailing the attendance along with details of timestamp

Table 3: REQUIREMENTS TRACEABILITY MATRIX

Functional Requirements Document FRD			**Test Case Document**
FR - ID	**FR Use case**	**Priority**	**TC - ID**
FR -1	Face Detection	High	TC-1
FR -2	Capturing Faces and Training Model	High	TC-2 TC-4 TC-5

FR -3	Storing images in the database	High	TC-3
FR -4	Face Recognition	High	TC-6
FR -5	Recording Attendance	High	TC-7
FR -6	Emailing Attendance	Medium	TC-8

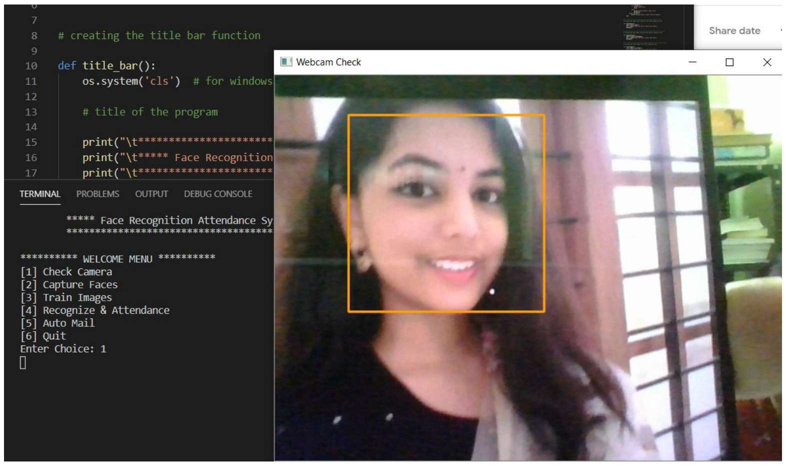
Test Plan

The main idea behind the test plan consists of testing the individual parts separately on the first stage for a unit test. Then merge all modules step by step for a final test. Therefore all modules functionality is validated through a unit test and the system functionality is verified through a final test.

Test Cases and Outputs

1: Web camera detecting faces live

The software locates the eyes and mouth, we can also rotate, scale, and shear the image which results in proper centering of the eyes and mouth to its best.



```
# creating the title bar function

def title_bar():
    os.system('cls')  # for windows

    # title of the program

    print("\t**********************
    print("\t***** Face Recognition
    print("\t**********************
```

```
     ***** Face Recognition Attendance Sy
     *************************************

********** WELCOME MENU **********
[1] Check Camera
[2] Capture Faces
[3] Train Images
[4] Recognize & Attendance
[5] Auto Mail
[6] Quit
Enter Choice: 1
```

Fig. Face detection

2: Capturing multiple images

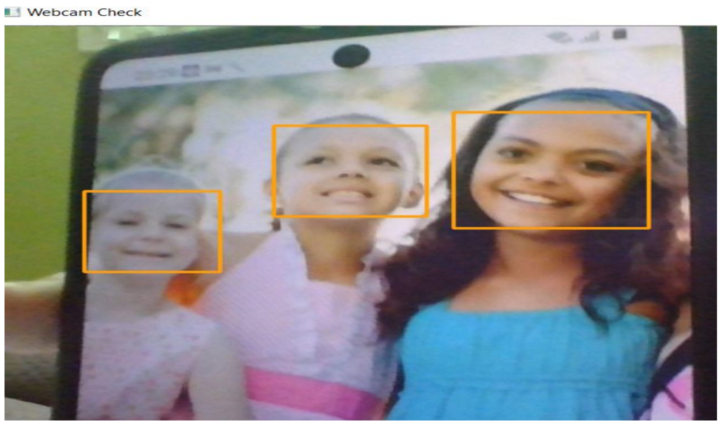
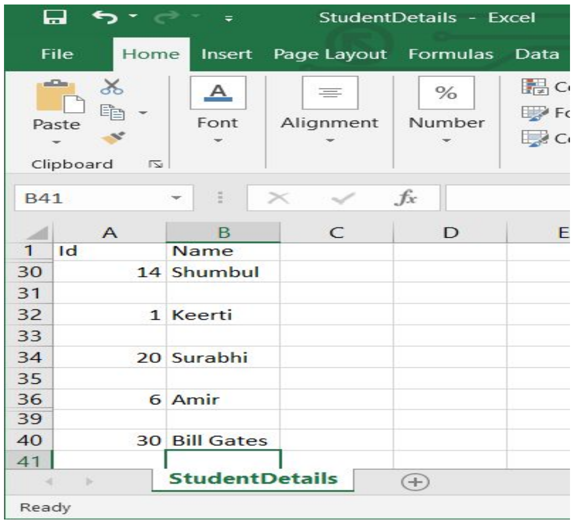


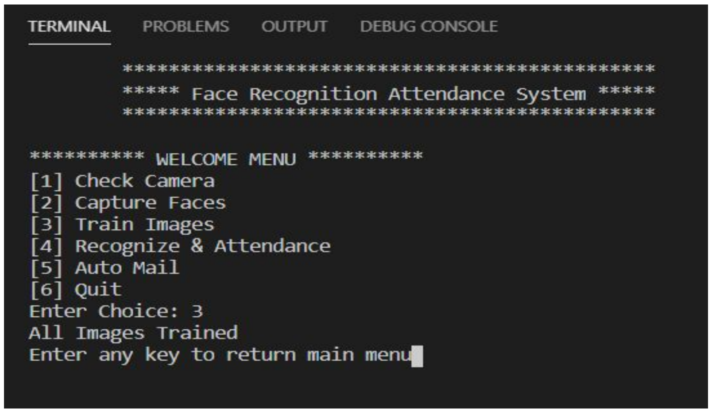
Fig. Detection of multiple faces

3: Storing images with student details



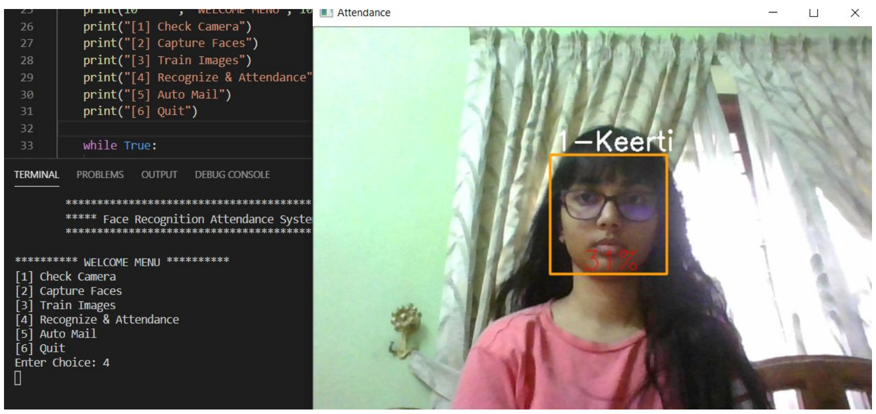
	A	B
1	Id	Name
30	14	Shumbul
31		
32	1	Keerti
33		
34	20	Surabhi
35		
36	6	Amir
39		
40	30	Bill Gates
41		

4: Training Model with captured images



```
        **************************************************
        ***** Face Recognition Attendance System *****
        **************************************************

********** WELCOME MENU **********
[1] Check Camera
[2] Capture Faces
[3] Train Images
[4] Recognize & Attendance
[5] Auto Mail
[6] Quit
Enter Choice: 3
All Images Trained
Enter any key to return main menu
```

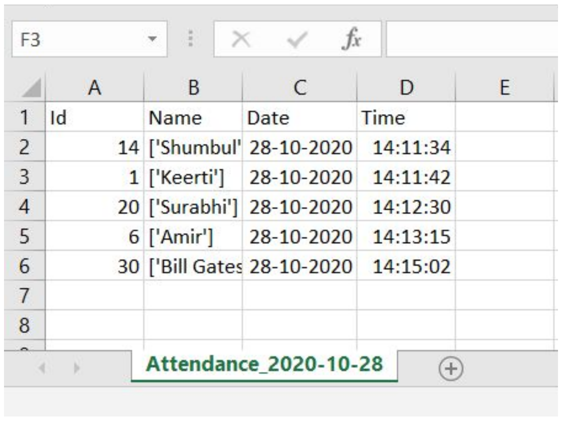
5: Recognizing faces registered in the database



```
    print("[1] Check Camera")
    print("[2] Capture Faces")
    print("[3] Train Images")
    print("[4] Recognize & Attendance"
    print("[5] Auto Mail")
    print("[6] Quit")

    while True:
```

```
        ************************************
        ***** Face Recognition Attendance Syste
        ************************************

********** WELCOME MENU **********
[1] Check Camera
[2] Capture Faces
[3] Train Images
[4] Recognize & Attendance
[5] Auto Mail
[6] Quit
Enter Choice: 4
```

6: Marking Attendance



F3

	A	B	C	D	E
1	Id	Name	Date	Time	
2	14	['Shumbul'	28-10-2020	14:11:34	
3	1	['Keerti']	28-10-2020	14:11:42	
4	20	['Surabhi']	28-10-2020	14:12:30	
5	6	['Amir']	28-10-2020	14:13:15	
6	30	['Bill Gates	28-10-2020	14:15:02	
7					
8					

Attendance_2020-10-28

Results and Conclusions

While testing the working model, we chose some pictures of ourselves, our friends, and some well-known celebrities. It turned out that the hit percent was pretty satisfying, i.e around 85%. Below is the observation table containing the information about the number of times faces got recognized correctly. The test was performed around 40-50 times and it includes multiple faces in the same image as well.

Table 4: Face recognition results and hit rates

Sr. No.	ID of person	No. of pictures	No. of times tested	No. of hits
1.	**14**	35	10	9
2.	**1**	40	10	9
3.	**20**	35	10	8
4.	**17**	37	10	7
5.	**6**	30	15	15
6.	**30**	35	15	14
7.	**33**	40	15	13

The face detection shows 88.23% of the right hit rate.

An important step is filtering out non-facial test images from an image. Several approaches have been taken, but it was not easy to find an absolute threshold value which can be applied to various pictures with different light conditions and composition. Approaches using luminance, average brightness, etc., have been tried, but they turned out not to be good enough to set an appropriate threshold for filtering out non-facial test images. Lastly, a statistical method was tried. After filtering out the leftmost column elements, which are12 test images. Out of 21 faces in the picture, the algorithm has detected 19 faces within the acceptable error of the location of faces. The two undetected faces are partially blocked by the other faces. Conclusively, this statistical approach, which is a seemingly rough estimation, works great in this picture and it turned out to produce good results for other pictures as well.

Constraints and Extensibility

The entire project has been developed from the requirements to a complete system alongside evaluation and testing. The system developed has achieved its aim and objectives. More careful analysis is required on a project intrinsically. The ways used may be combined with others to attain nice results. Completely different ways are enforced within the past in keeping with the literature review. The conclusion to set the parameters of this part of the system based on a very small class size was due to the failures obtained from the recognition part of the system. The size of the image is very important in face recognition as every pixel counts. The algorithm we've used would be analyzed with images of different sizes and these images would be showing students in a classroom setting with natural sitting positions showing faces of different sizes. The basic idea is, no matter how the face is turned, we should be able to center the eyes and mouth in roughly the same position in the image, thus recognizing the person and marking their attendance.

In the future this system would be improved as a result of these systems usually fails to recognize students from some distance, additionally, we tend to have some processing limitations, operating with a system of high processing speed could result in even higher performance of this software.

References

[1] K.Senthamil Selvi et al, "Face Recognition Based Attendance Marking System" in International Journal of Computer Science and Mobile Computing, Vol.3 Issue.2, February- 2014.

[2] CH. Vinod Kumar and Dr. K. Raja Kumar, "Face Recognition Based Student Attendance System with OpenCV" in International Journal of Advanced Technology and Innovative Research Volume. 08, issue.24, December-2016.

[3] Shervin EMAMI and Valentin Petrut SUCIU, "Facial Recognition using OpenCV", ResearchGate article March 2012.

[4] Divyarajsinh N. Parmar and Brijesh B. Mehta, "Face Recognition Methods & Applications", an article in International Journal of Computer Applications in Technology · January 2014.

[5] Sudhir Bussa, Shruti Bharuka, Ananya Mani, and Sakshi Kaushik, "Smart Attendance System using OPENCV based on Facial Recognition", Vol. 9 Issue 03, March-2020.