

A Project Report

On

Smart Face-Py Monitor

*Submitted in partial fulfilment of the
requirement for the award of the degree of*

**Bachelor of Technology in
Computer Science & Engineering**



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of

DR. Michael Raj TF

Professor

Submitted By

Amaan Akhtar Ansari – 18SCSE1010530

Aman Kumar Goyal – 18SCSE1010356

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA**

December, 2021

**SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA**

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled “ **Smart Face- Py Monitor** ” in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of JULY-2021 to DECEMBER-2021, under the supervision of **DR. Michael Raj TF, Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1010530 – AMAAN AKHTAR ANSARI

18SCSE1010356 – AMAN KUMAR GOYAL

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor

(Dr. Michael Raj TF)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of
**18SCSE1010530 – AMAAN AKHTAR ANSARI, 18SCSE1010356 – AMAN
KUMAR GOYAL** has been held on _____ and his/her work is
recommended for the award of **BACHELOR OF TECHNOLOGY IN
COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

Date:

Place:

Abstract

Automatic face recognition (AFR) technologies have seen dramatic improvements in performance over the past years, and such systems are now widely used for security and commercial applications. An automated system for human face recognition in a real time background for a college to mark the attendance of their employees. So Smart Attendance using Real Time Face Recognition is a real world solution which comes with day to day activities of handling employees. The task is very difficult as the real time background subtraction in an image is still a challenge. To detect real time human face are used and a simple fast Principal Component Analysis has used to recognize the faces detected with a high accuracy rate. The matched face is used to mark attendance of the employee. Our system maintains the attendance records of employees automatically.

Manual entering of attendance in logbooks becomes a difficult task and it also wastes the time. So, we designed an efficient module that comprises of face recognition to manage the attendance records of employees. Our module enrolls the staff's face. This enrolling is a onetime process and their face will be stored in the database. During enrolling of faces we require a system since it is a onetime process. You can have your own roll number as your employee id which will be unique for each employee. The presence of each employee will be updated in a database. The results showed improved performance over manual attendance management system. Attendance is marked after employee identification. This product gives much more solutions with accurate results in user interactive manner rather than existing attendance and leave management systems.

List of Figures

Figure No.	Page No.
Figure 1	11
Figure 2	15
Figure 3	16
Figure 4	17
Figure 5	18
Figure 6	26
Figure 7	27
Figure 8	33
Figure 9	34

Table of Contents

Title	Page No.
Abstract	I
List of Figures	II
Chapter 1 Introduction	
1.1 Introduction	7-9
1.2 Formulation of Problem	10
1.3 Tools and Technology Used	11
Chapter 2 Literature Review	12-14
Chapter 3 Architecture Diagram	15-18
Chapter 4 Module Description	19-23
Chapter 5 Framework and Algorithm Used	24-30
Chapter 6 Result	31-32
Chapter 7 Conclusion	34-35
Chapter 8 Source Code	36-47
Chapter 9 Reference	48

Chapter -1

1.1 INTRODUCTION

In computer vision, one essential problem we are trying to figure out is to automatically detect objects in an image without human intervention. Face detection can be thought of as such a problem where we detect human faces in an image. There may be slight differences in the faces of humans but overall, it is safe to say that there are certain features that are associated with all the human faces. There are various face detection algorithms but Viola-Jones Algorithm is one of the oldest methods that is also used today and we will use the same later in the article. You can go through the Viola-Jones Algorithm after completing this article as I'll link it at the end of this article.

Face detection is usually the first step towards many face-related technologies, such as face recognition or verification. However, face detection can have very useful applications. The most successful application of face detection would probably be photo taking. When you take a photo of your friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly.

Recent advances in automated face analysis, pattern recognition and machine learning have made it possible to develop automatic face recognition systems to address these applications. On the one hand, recognising face is natural process, because people usually do it effortlessly without much conscious. On the other hand, application of this process in area of computer vision remains a difficult problem. Being part of a biometric technology, automated face recognition has a plenty of desirable properties. They are based on the important advantage—non-invasiveness. The various biometric methods can be distinguished into physiological (fingerprint, DNA, face) and behavioural (keystroke, voice print)

categories. The physiological approaches are more stable and non-alterable, except by severe injury. Behavioural patterns are more sensitive to human overall condition, such as stress, illness or fatigue.

Face detection performance is a key issue, so techniques for dealing with non-frontal face detection are discussed. Subspace modelling and learning-based dimension reduction methods are fundamental to many current face recognition techniques. Discovering such subspaces so as to extract effective features and construct robust classifiers stands another challenge in this area. Face recognition has merits of both high accuracy and low intrusive, so it has drawn the attention of the researches in various fields from psychology, image processing to computer vision.

The first stage is face detection in the acquired image that is regardless of scale and location. It often uses an advanced filtering procedure to distinguish locations that represent faces and filters them with accurate classifiers. It is notable that all translations, scaling and rotational variations have to be dealt in the face detection phase. For example, regarding to , facial expressions and hairstyle changes or smiling and frowning face still stands important variations during pattern recognition stage.

Considering roughly presented elements above of the complex process of face recognition, a number of limitations and imperfections can be seen. They require clarification or replacing by new algorithms, methods or even technologies.

In this chapter, we have discussed face recognition processing, including major components such as face detection, tracking, alignment and feature extraction, and it points out the technical challenges of building a face recognition system. We focus on the importance of the most successful solutions available so far.

The final part of the chapter describes chosen face recognition methods and applications and their potential use in areas not related to face recognition.

The need for this study is justified by an invitation to participate in the further development of a very interesting technology, which is face recognition.

Despite the fact, there is continual performance improvement regarding several face recognition technology areas, and it is worth to note that current applications also impose new requirements for its further development.

Face recognition is a technology that is capable of identifying an object through an image or a video. It is a method of biometric identification that uses the body measures like face and head for the verification of an identity of a person through its facial biometrics. Basically, this technology collects the set of unique biometric patterns and data of each person associated with their facial expression to authenticate a person. This requires any device that has digital photographic technology to generate and obtain the images and record facial pattern.

Hence, Smart Face-py Monitor is also based upon that technology of capturing a person's face and then authenticating and finally mark down his/her attendance with the current date and time. It basically works by capturing an incoming image from a camera device and then verify that image with the image present in the database.

At first, usually a person's face is registered into the system with their allotted roll numbers or enrolment id. The system needs to train itself with multiple photos and then it become ready to take down the attendance. So, whenever the registered person comes in front of the camera, the system automatically note down the attendance and store it into an Excel sheet along with the date and time.

So, the process of marking the attendance become very easy and the concept of AI & ML has made these process much easier too as it made the operation highly secure and reliable. Integration of the algorithms like Viola Jones, Gaussian Distribution and computing techniques make the process to run in real-time.

1.2 FORMULATION OF PROBLEM

Recognizing faces in computer vision is a challenging problem. The illumination problem, the pose problem, scale variability, low quality image acquisition, partially occluded faces are some examples of the issues to deal with. Thus, face recognition algorithms must exhibit robustness to variations in the above parameters. The existing techniques do not perform well in cases of different illumination, background or rotation. Thus, there is a need to address the above mentioned disadvantages. The project aims to design and implement a system which is less sensitive to illumination, is rotation invariant, scale invariant and robust enough to be implemented in practical applications.

The manual attendance record system is not efficient and requires more time to arrange record and to calculate the average attendance of each student. Hence there is a requirement of a system that will solve the problem of student record arrangement and student average attendance calculation. One alternative to make student attendance system automatic is provided by face recognition.

In today's scenario of Covid-19, many places the attendance is taken via fingerprint which would be an effective reason for the spread of this virus. So, face recognition based attendance would also be much beneficial in this situation which makes the process of attendance contactless and spreading of the disease can be reduced. Hence, we can say that this is the safe way of taking down the attendance and also much efficient than other ways.

1.3 TOOLS AND TECHNOLOGY USED

- We first need to have Python environment installed in our systems.
- Platform – Python IDLE 3.8
Since Python 3 is very much advanced as compare to Python 2 and there are even much more libraries present in the current Python 3 version.
- All the packages that are need in the project should be installed like Numpy, Pandas, Tkinter, OpenCV and many more.
- Primary or secondary Webcam should be set up.
- Database – MS Excel is used as a database to store the attendance.



Figure 1

Chapter- 2

LITERATURE REVIEW/ COMPARATIVE STUDY

Maintaining the attendance is very important in all the institutes for checking the performance of employees. Every institute has its own method in this regard. Some are taking attendance manually using the old paper or file based approach and some have adopted methods of automatic attendance using some biometric techniques. But in these methods employees have to wait for long time in making a queue at time they enter the office. Many biometric systems are available but the key authentications are same in all the techniques. Every biometric system consists of enrolment process in which unique features of a person is stored in the database and then there are processes of identification and verification.

But today's scientific advancements can impact globally in very short time. Such progress is made in the field of artificial intelligence or AI. Keeping this in mind we are going to make an Artificial Intelligence based face recognition system that can have high impact on different organizations. Uniqueness or independence of an individual is his face. By considering this fact our system will be super faster and accurate in marking attendance of individual student. We are going to use face detection and recognition in this project. Face detection is used to locate the position of face region and face recognition is used for marking the attendance. The database will store faces of students. When the face of the student matches with one of the faces stored in the database then the attendance is recorded.

Comparative Study:-

One of the projects which is already going on in the market is based on Raspberry Pi module. In this all sorts of coding are done in limited way and the system and camera needed to be set up. In this case every camera needed to be in sync with Raspberry Pi, then only the software will run. If there are many numbers of cameras, then number of Raspberry Pi toolkit should be increased leading to more cost factor. But in our attendance system no need to buy any tool kit, only the webcam which are placed in the institution are sufficient to take down the attendance.

Advantages of face detection

As a key element in facial imaging applications, such as facial recognition and face analysis, face detection creates various advantages for users, including:

- Improved security. Face detection improves surveillance efforts and helps track down criminals and terrorists. Personal security is also enhanced since there is nothing for hackers to steal or change, such as passwords.
- Easy to integrate. Face detection and facial recognition technology is easy to integrate, and most solutions are compatible with the majority of security software.
- Automated identification. In the past, identification was manually performed by a person; this was inefficient and frequently inaccurate. Face detection allows the identification process to be automated, thus saving time and increasing accuracy.

Disadvantages of face detection

While face detection provides several large benefits to users, it also holds various disadvantages, including:

- Massive data storage burden. The ML technology used in face detection requires powerful data storage that may not be available to all users.
- Detection is vulnerable. While face detection provides more accurate results than manual identification processes, it can also be more easily thrown off by changes in appearance or camera angles.
- A potential breach of privacy. Face detection's ability to help the government track down criminals creates huge benefits; however, the same surveillance can allow the government to observe private citizens. Strict regulations must be set to ensure the technology is used fairly and in compliance with human privacy rights.

Chapter- 3

ARCHITECTURE DIAGRAM

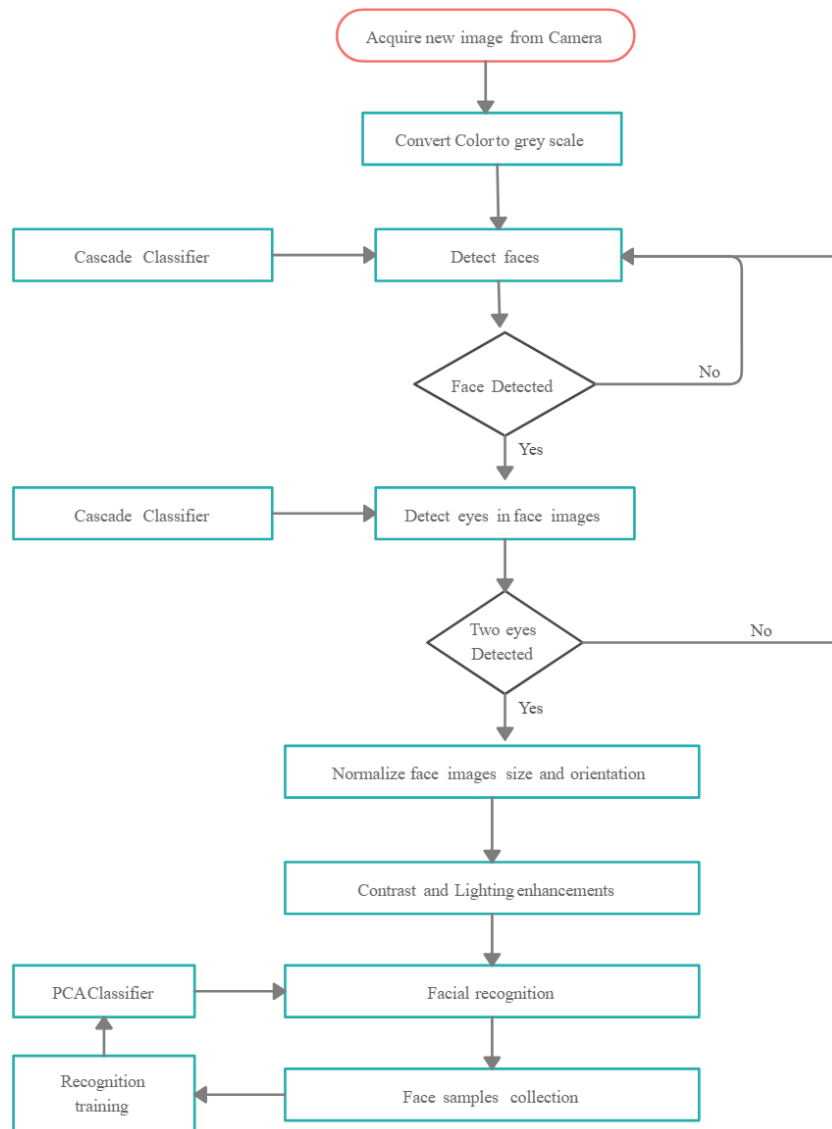


Figure 2

Above figure 2 is the Data Flow Diagram of face recognition attendance system. This is how the process flow from one module to another. At first image is captured by the software and the dataset is created by mapping various face coordinates and allot the students name and Id. Hence, afterwards if same face

comes in front of camera, the system automatically recognizes the face and record the attendance.

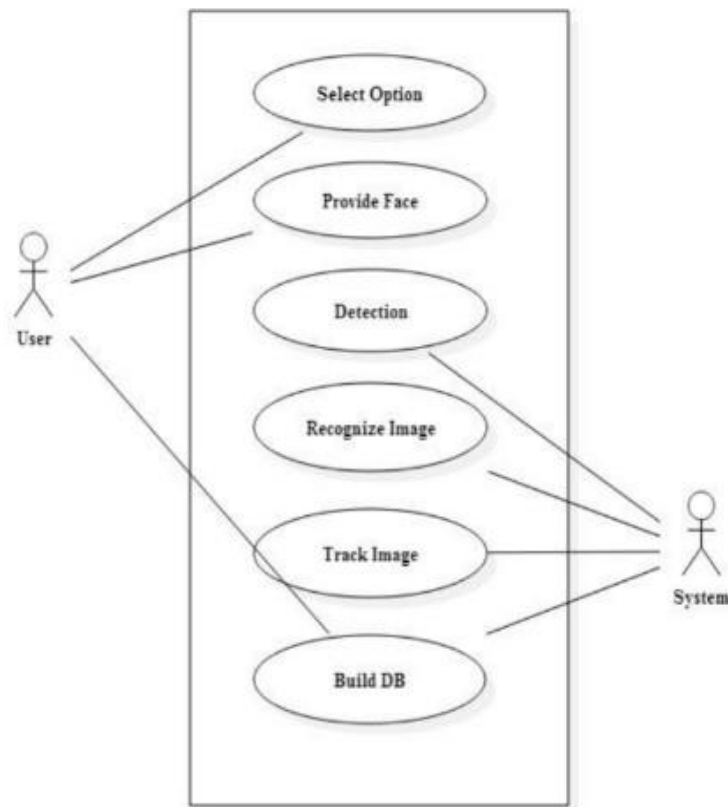


Figure 3

A use case diagram (figure 3) is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.

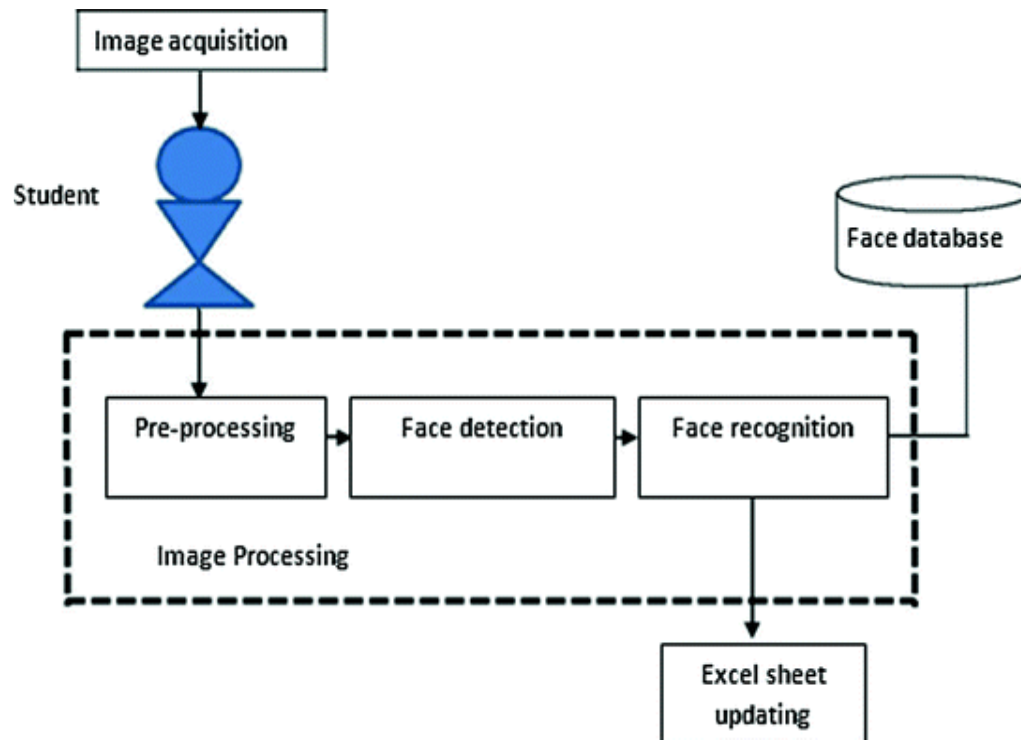


Figure 4

Above figure 4 show how the faces gets captured, recognized and stored into the database and the output is shown to the user.

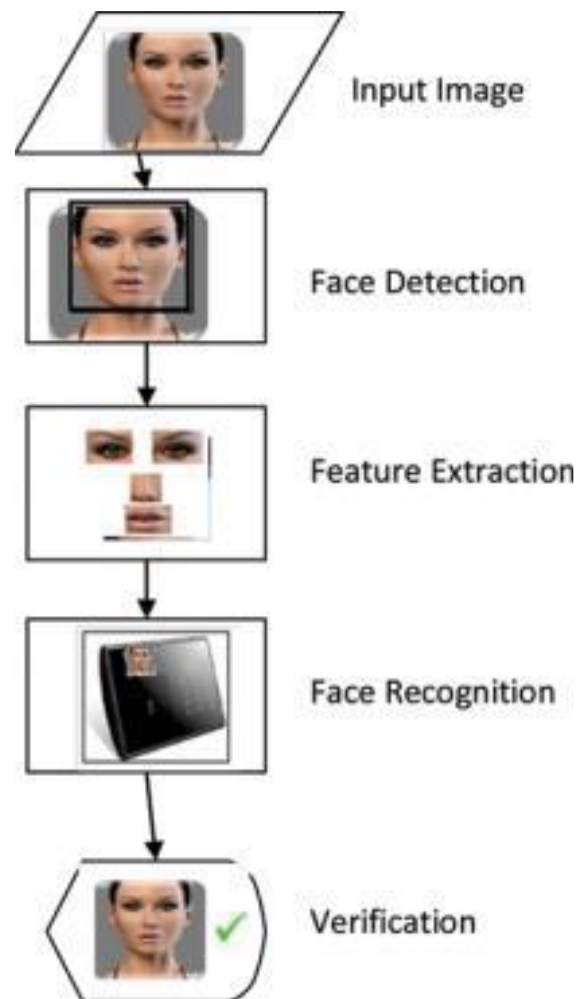


Figure 5

Above figure 5 shows how the input is processed i.e., at first face is cropped and feature extraction is done to differentiate different faces. After that it gets stored in the database and at the time of scanning the verification is done by matching the person's face with the face stored in the database.

Chapter-4

Module Description

Computer Vision

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

Applications of OpenCV: There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations

- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

Image-Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”.

Digital-Image :

An image may be defined as a two-dimensional function $f(x, y)$, where x and y are spatial(plane) coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function $f(x, y)$ at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

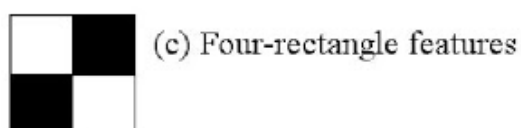
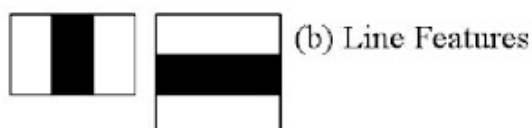
Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

- Importing the image
- Analysing and manipulating the image
- Output in which result can be altered image or report that is based on image analysis

Open CV built-in Haar Cascades:

It is an effective detection method and is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in below image are used. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.



Now all possible sizes and locations of each kernel are used to calculate plenty of features. For each feature calculation, we need to find the sum of the pixels under the white and black rectangles. To solve this, they introduced the integral images. It simplifies calculation of the sum of the pixels, how large may be the number of pixels, to an operation involving just four pixels.

To install OpenCV, type in command prompt

```
pip install opencv-python
```

I have tried various ways to install dlib on Windows but the easiest of all of them is via Anaconda. First, install Anaconda (here is a guide to install it) and then use this command in your command prompt:

```
conda install -c conda-forge dlib
```

Next to install face_recognition, type in command prompt

```
pip install face_recognition
```

Now that we have all the dependencies installed, let us start coding. We will have to create three files, one will take our dataset and extract face embedding for each face using dlib. Next, we will save these embedding in a file.

Tkinter GUI:

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications.

Creating a GUI using tkinter is an easy task.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. `Tk(screenName=None, baseName=None, className='Tk', useTk=1)`: To create a main window, tkinter offers a method `'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'`. To change the name of the window, you can change the `className` to the desired one.
2. `mainloop()`: There is a method known by the name `mainloop()` is used when your application is ready to run. `mainloop()` is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

Chapter-5

Framework and Algorithms Used

There has been a rapid development of the reliable face recognition algorithms in the last decade. The traditional face recognition algorithms can be categorised into two categories: holistic features and local feature approaches. The holistic group can be additionally divided into linear and nonlinear projection methods.

However, due to large variations in illumination conditions, facial expression and other factors, these methods may fail to adequately represent the faces. The main reason is that the face patterns lie on a complex nonlinear and non-convex manifold in the high-dimensional space. These methods project face onto a linear subspace spanned by the eigenface images. The distance from face space is the orthogonal distance to the plane, whereas the distance in face space is the distance along the plane from the mean image. These both distances can be turned into distances and given probabilistic interpretations.

The Gabor wavelet applied at fixed positions, in correspondence of the nodes of a square-meshed grid superimposed to the face image, is presented. Each subpattern of the partitioned face image is defined as the extracted Gabor features that belong to the same row of the square-meshed grid which are then projected to lower dimension space by Karhunen–Loeve transform. The obtained features of each subpattern, which are weighted using genetic algorithm (GA), are used to train a Parzen Window Classifier. Finally, matching process is done by combining the classifiers using a weighted sum rule.

Viola- Jones Algorithm

The Viola–Jones object detection framework is an object detection framework which was proposed in 2001 by Paul Viola and Michael Jones. Although it can be trained to detect a variety of object classes, it was motivated primarily by the problem of face detection. To make the task more manageable, Viola–Jones requires full view frontal upright faces. Thus in order to be detected, the entire face must point towards the camera and should not be tilted to either side. While it seems these constraints could diminish the algorithm's utility somewhat, because the detection step is most often followed by a recognition step, in practice these limits on pose are quite acceptable.

Before detecting a face, the image is converted into grayscale, since it is easier to work with and there's lesser data to process. The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the colored image.

Detection happens inside a detection window. A minimum and maximum window size is chosen, and for each size a sliding step size is chosen. Then the detection window is moved across the image as follows:

1. Set the minimum window size, and sliding step corresponding to that size.
2. For the chosen window size, slide the window vertically and horizontally with the same step. At each step, a set of N face recognition filters is applied. If one filter gives a positive answer, the face is detected in the current window.
3. If the size of the window is the maximum size stop the procedure. Otherwise increase the size of the window and corresponding sliding step to the next chosen size and go to the step 2.

Gaussian distribution

The algorithm is based on skin color clustering features and projecting distribution characteristic. Firstly, the approach utilizes the skin color features to transform the original image into a binary-image. Then projecting the binary-image to X-axes and Y-axes respectively, two curves are obtained. Their means and variances can be obtained. Gaussian distribution curves with same means and variances can be calculated. Finally, according to the solution of Gaussian equation, an accurate face region is found. Simulation shows that the method proposed is fast and needs less computation.

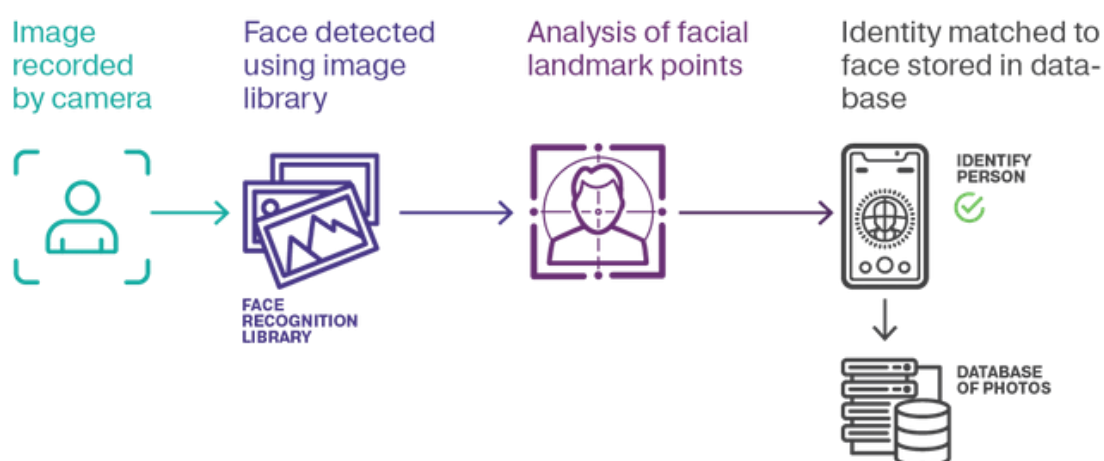


Figure 6

Comparing faces: Now that we have face embeddings for every face in our data saved in a file, the next step is to recognise a new image that is not in our data. So the first step is to compute the face embedding for the image using the same network we used above and then compare this embedding with the rest of the embeddings we have. We recognise the face if the generated embedding is closer or similar to any other embedding as shown below:



Figure 7

So we passed two images, one of the images is of Vladimir Putin and other of George W. Bush. In our example above, we did not save the embeddings for Putin but we saved the embeddings of Bush. Thus when we compared the two new embeddings with the existing ones, the vector for Bush is closer to the other face embeddings of Bush whereas the face embeddings of Putin are not closer to any other embedding and thus the program cannot recognise him.

Face Detection: The very first task we perform is detecting faces in the image or video stream. Now that we know the exact location/coordinates of face, we extract this face for further processing ahead.

Feature Extraction: Now that we have cropped the face out of the image, we extract features from it. Here we are going to use face embeddings to extract

the features out of the face. A neural network takes an image of the person's face as input and outputs a vector which represents the most important features of a face. In machine learning, this vector is called embedding and thus we call this vector as face embedding.

Uses of face detection

Although all facial recognition systems use face detection, not all face detection systems are used for facial recognition. Face detection can also be applied for facial motion capture, or the process of electronically converting a human's facial movements into a digital database using cameras or laser scanners. This database can be used to produce realistic computer animation for movies, games or avatars.

Face detection can also be used to auto-focus cameras or to count how many people have entered an area. The technology also has marketing applications -- for example, displaying specific advertisements when a particular face is recognized.

Another application for face detection is as part of a software implementation of emotional inference, which can, for example, be used to help people with autism understand the feelings of people around them. The program "reads" the emotions on a human face using advanced image processing.

An additional use is drawing language inferences from visual cues, or "lip reading." This can help computers determine who is speaking, which may be helpful in security applications. Furthermore, face detection can be used to help determine which parts of an image to blur to assure privacy.

FEATURES

- a. Enter Id-** This field is for to enter the id like enrolment number or admission number of the person.
- b. Enter Name-** This field for to enter the name of the person.
- c. Notification-** This is the area where various useful information is shown to the user like name and id and images train status.
- d. Take Images-** This button is used to take the images of the student or employee.
- e. Train Images-** This button is to train the images to the software for future verification.
- f. Track Images-** This button is to note down the attendance.
- g. Quit-** this button is to exit the software.
- h. Clear-** This button is used to clear the field like name and id.

Chapter-6

Result

The present system of attendance marking i.e., manually calling out the roll call by the faculty have quite satisfactorily served the purpose. With the change in the educational system with the introduction of new technologies in classroom such as virtual class room the traditional way of taking attendance may not be viable anymore. Even with rising number of course of study offered by universities, processing of attendance manually could be time consuming. Hence, in our project we aim at creating a system to take attendance using facial recognition technology in classrooms and creating an efficient database to record them.

We have seen that there are many software that take down the attendance using fingerprint system, retina system, and many more. So, there's a need for a system too that need to identify the people faces too. Therefore, we have decided to develop a program that automatically note down the attendance by detecting the facing without any physical involvement. This also reduces the work load of the teachers or staff who used to spend a much time in noting down the attendance.

Also, by seeing the today's scenario of COVID-19 we need to take more precaution about physical contact , that's why we have build a project that do not require any physical interaction.

Hereby we assure that our software Smart Face-py Monitor will help various institution a lot to take and manage the attendance of the employees or students. This will help them to make their work easier and also save their time. It will be proved helpful for the organisation which want low budget software for attendance marking because our software is not very expensive. In the times of COVID-19 , our software will lead to no contact based marking where as in fingerprint attendance system fingers are contacted to the scanner, but our

software provide fully non contact marking of attendance. In short, our application will be beneficial to every organisations.

Output

Interface-

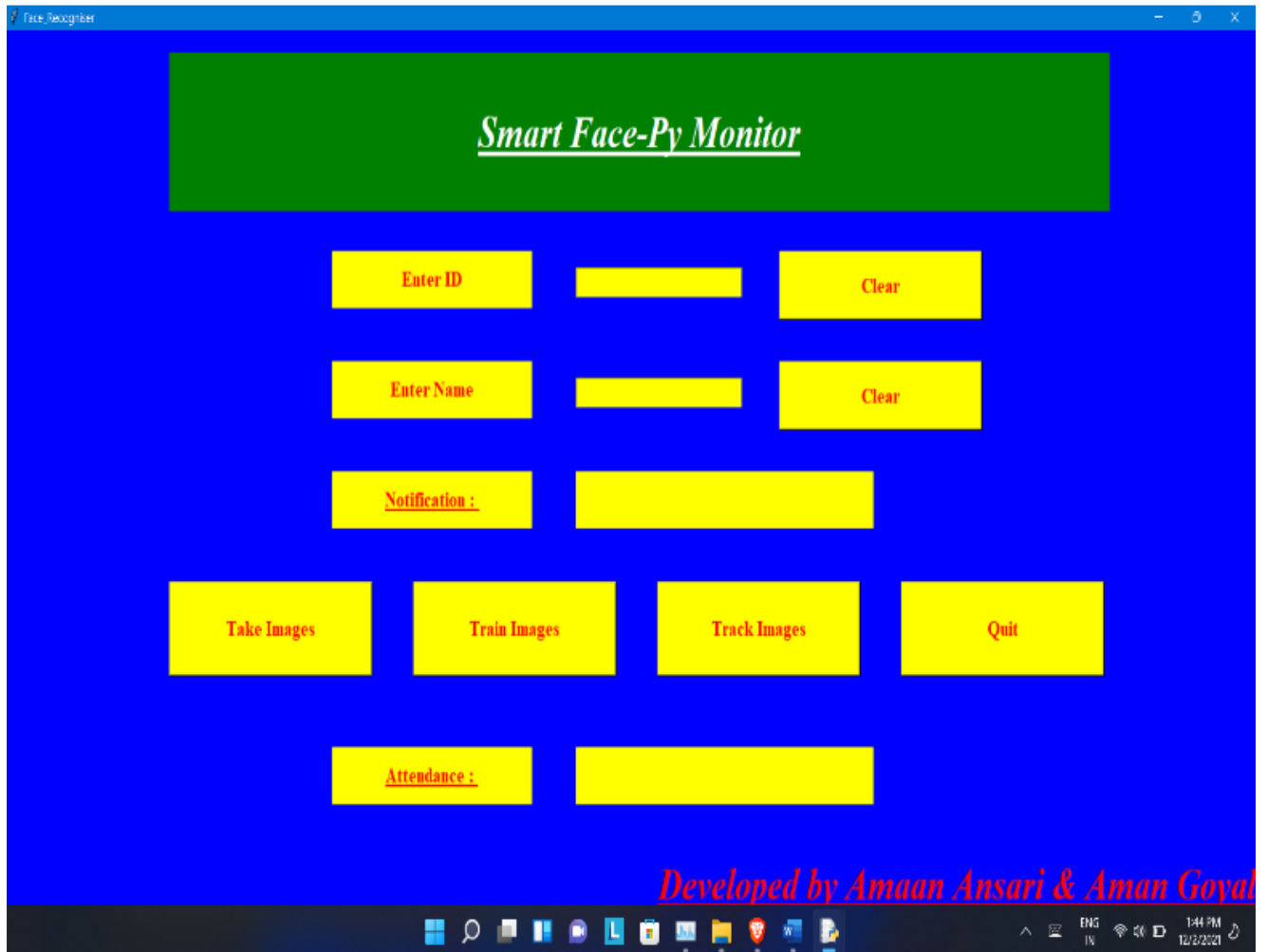
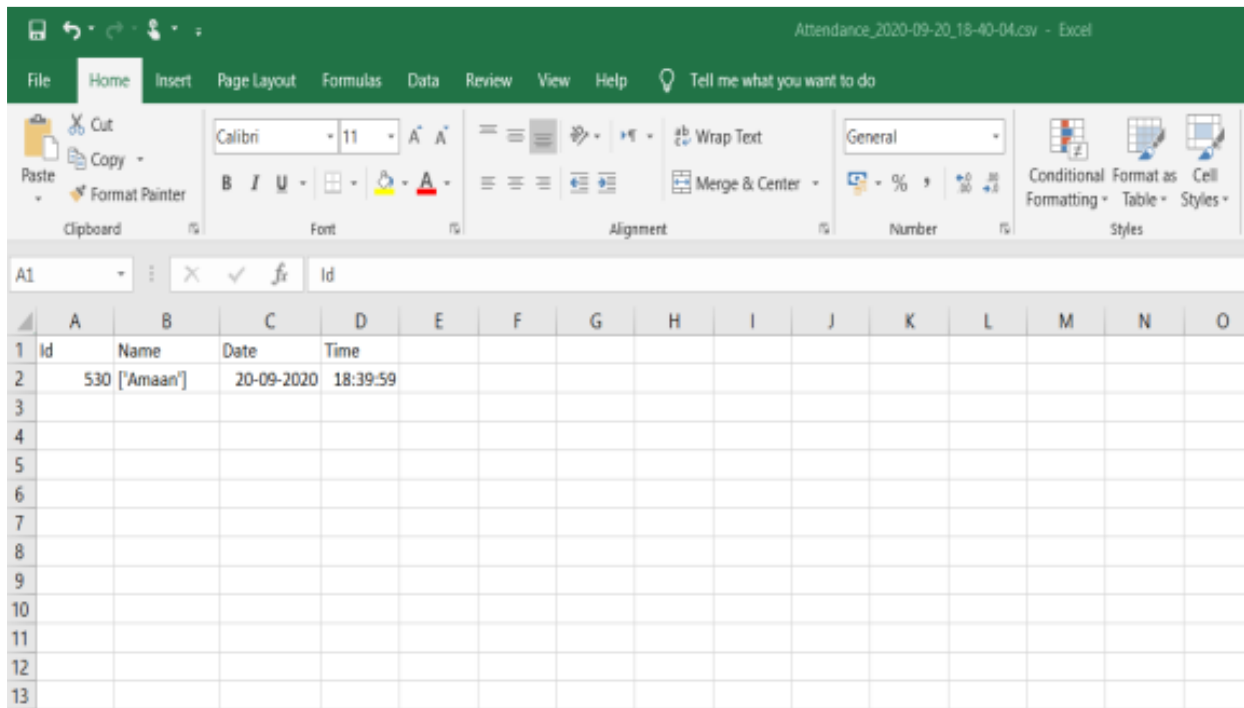


Figure 8

MS Excel (as database) –



The screenshot displays the Microsoft Excel interface with the 'Home' tab selected. The ribbon includes options for Clipboard, Font, Alignment, Number, and Styles. The spreadsheet shows a table with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Id	Name	Date	Time											
2	530	[Amaan]	20-09-2020	18:39:59											
3															
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															

Figure 9

Chapter-7

Conclusion

Hereby we assure that our software Smart Face-py Monitor will help various institution a lot to take and manage the attendance of the employees or students. This will help them to make their work easier and also save their time. It will be proved helpful for the organisation which want low budget software for attendance marking because our software is not very expensive. In the times of COVID-19 , our software will lead to no contact based marking where as in fingerprint attendance system fingers are contacted to the scanner, but our software provide fully non contact marking of attendance. In short, our application will be beneficial to every organisations.

Face recognition technology has come a long way in the last twenty years. Today, machines are able to automatically verify identity information for secure transactions, for surveillance and security tasks, and for access control to buildings etc. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy. However, next generation face recognition systems are going to have widespread application in smart environments -- where computers and machines are more like helpful assistants.

To achieve this goal computers must be able to reliably identify nearby people in a manner that fits naturally within the pattern of normal human interactions. They must not require special interactions and must conform to human intuitions about when recognition is likely. This implies that future smart environments should use the same modalities as humans, and have approximately the same limitations. These goals now appear in reach -- however, substantial research remains to be

done in making person recognition technology work reliably, in widely varying conditions using information from single or multiple modalities.

Face recognition is still a challenging problem in the field of computer vision. It has received a great deal of attention over the past years because of its several applications in various domains. Although there is strong research effort in this area, face recognition systems are far from ideal to perform adequately in all situations from real world. Paper presented a brief survey of issues methods and applications in area of face recognition. There is much work to be done in order to realise methods that reflect how humans recognise faces and optimally make use of the temporal evolution of the appearance of the face for recognition.

Chapter-8

Source 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("Face_Recogniser")

dialog_title = 'QUIT'

dialog_text = 'Are you sure?'
```

```
message = tk.Label(window, text="Smart Face-Py Monitor" ,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("TrainingImageLabel\Trainer.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
```

```
pillImage=Image.open(imagePath).convert('L')

#Now we are converting the PIL image into numpy array

imageNp=np.array(pillImage,'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("TrainingImageLabel\Trainer.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", "Developed by Amaan Ansari & Aman Goyal")
```

```
copyWrite.configure(state="disabled",fg="red" )
```

```
copyWrite.pack(side="left")
```

```
copyWrite.place(x=800, y=750)
```

```
window.mainloop()
```

Chapter-9

References

- [1]. <https://www.superdatascience.com/blogs/opencv-face-recognition>

- [2]. <https://www.slideshare.net/ShreyaDandavate/face-recognition-attendance-system-96913577>

- [3]. https://www.youtube.com/watch?v=mrDjwXZGxIab_channel=edureka%21