CHAPTER 1
INTRODUCTION

Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection and Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face. Thus, I thought to use the same technique for Automatic Attendance System for my own organization.

The face is our primary focus of attention in social life playing an important role in conveying identity and emotions. We can recognize a number of faces learned throughout our lifespan and identify faces at a glance even after years of separation. This skill is quite robust despite of large variations in visual stimulus due to changing condition, aging and distractions such as beard, glasses or changes in hairstyle. Computational models of face recognition are interesting because they can contribute not only to theoretical knowledge but also to practical applications.

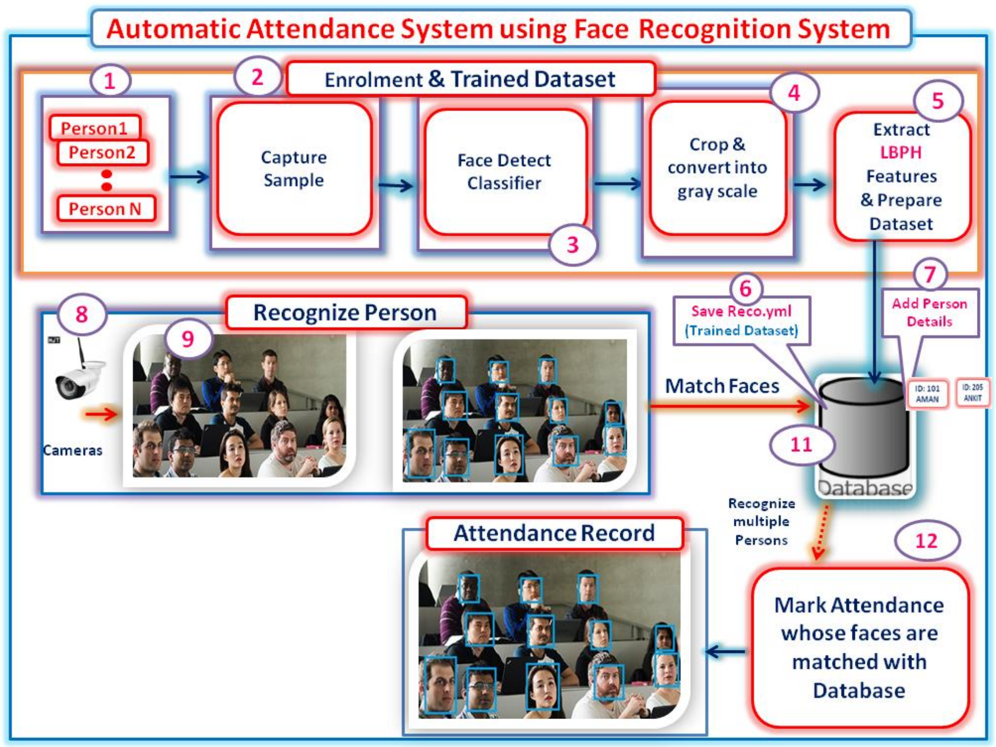

Automatic Attendance System using Face Recognition System
Enrolment & Trained Dataset
1
Person1
Person2
Person N
2
Capture Sample
Face Detect Classifier
3
4
Crop & convert into gray scale
5
Extract LBPH Features & Prepare Dataset
6
Save Reco.yml (Trained Dataset)
7
Add Person Details
8
Recognize Person
9
Cameras
Match Faces
11
Database
Recognize multiple Persons
12
Attendance Record
Mark Attendance whose faces are matched with Database

Figure 1.1: Project Flowchart

Earlier attendance used to be taken by the concerned teacher manually; facing a lot of problems. Then we came with a technology of biometric based fingerprint attendance system; which does not works well cut, mehndi designs and have other problems as well. So, I have used the most demanding technology i.e. Machine Learning that use face recognition technique to mark attendance of respective present and absent students. Also, computers that detect and recognize faces could be applied to a wide variety of tasks including criminal identification, security system, image and film processing, identity verification, tagging purposes and human-computer interaction. Face detection is used in many places now a day’s especially the websites hosting images like picassa, photobucket and facebook. The automatically tagging feature adds a new dimension to sharing pictures among the people who are in the picture and also gives the idea to other people about who the person is in the image.

MAIN MODULE

1.1 MODULES

Automatic Attendance Management System using face recognition is the task of identifying already enrolled students as a present or absent face. Often the problem of face recognition is confused with the problem of face detection, Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

The project consists of major modules:

1) Create/Reset Database
2) Capture Samples
3) Train Faces
4) Face Detection
5) Face Recognition
6) Mark Attendance

7) View Attendance
8) Flowchart and Presentation(using Pyhton only)
9) Technology Stack
10) About Us

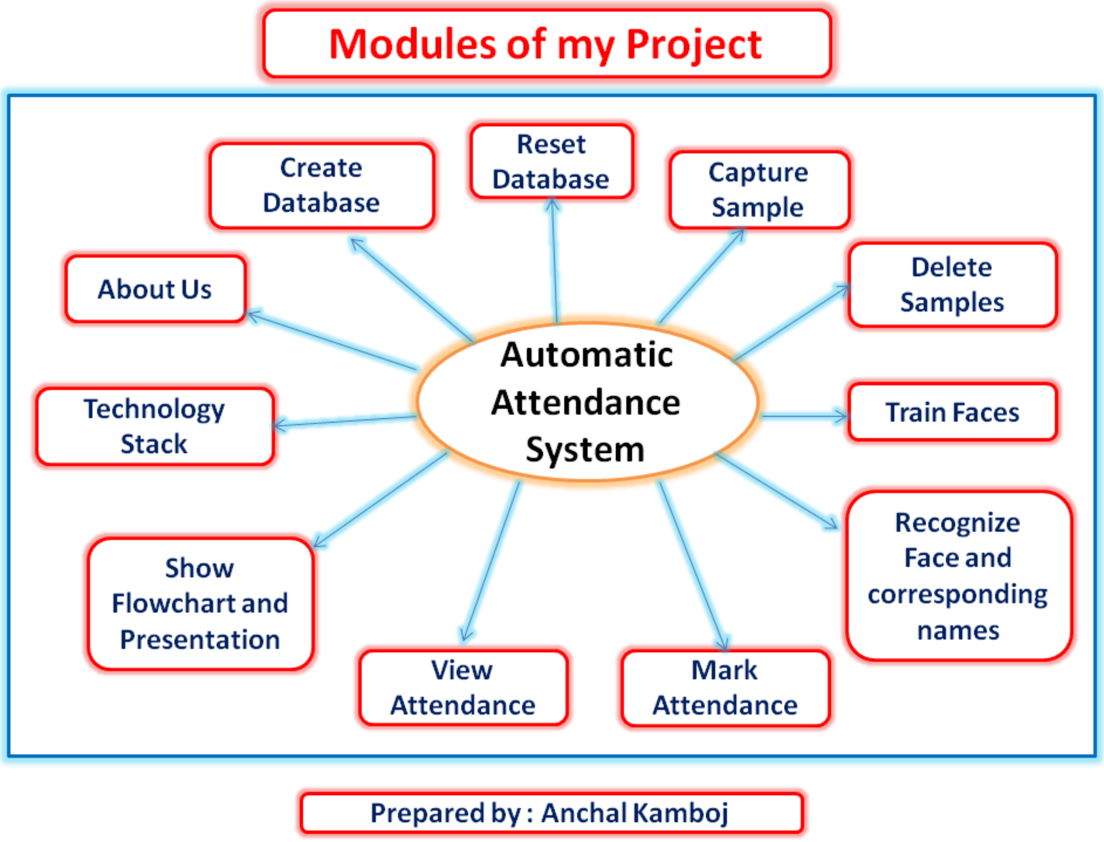


Figure 1.2: Project Modules

1.2 FACE DETECTION:

The main function of this step is to determine whether human faces appear in a given image, and where these faces are located at. The expected outputs of this step are patches containing each face in the input image. In order to make further face recognition

system more robust and easy to design, face alignment are performed to justify the scales and orientations of these patches. Besides serving as the pre-processing for face recognition, face detection could be used for region of interest detection, retargeting, video and image classification, etc.

1.3 FEATURE EXTRACTION AND RECOGNITION:

After the face detection step, human-face patches are extracted from images. Directly using these patches for face recognition have some disadvantages, first, each patch

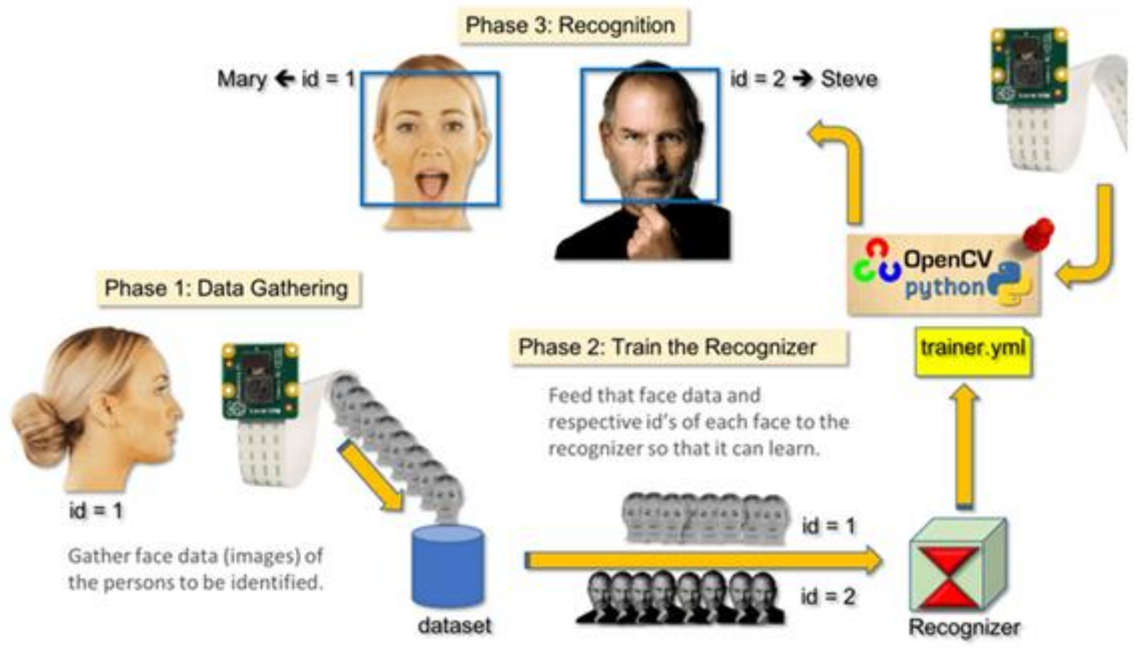

Phase 3: Recognition
Mary ← id = 1
id = 2 → Steve
OpenCV
python
Phase 1: Data Gathering
id = 1
Gather face data (images) of the persons to be identified.
dataset
Phase 2: Train the Recognizer
Feed that face data and respective id's of each face to the recognizer so that it can learn.
id = 1
id = 2
trainer.yml
Recognizer

Fig 1.3: Face Detection Procedure

usually contains over 1000 pixels, which are too large to build a robust recognition system1. Second, face patches may be taken from different camera alignments, with different face expressions, illuminations, and may suffer from occlusion and clutter. To overcome these drawbacks, feature extractions are performed to do in-formation packing, dimension reduction, salience extraction, and noise cleaning. After this step, a face patch is usually transformed into a vector with fixed dimension or a set of fiducial points and their corresponding locations.

For each person, several images are taken and their features are extracted and stored in the database. Then when an input face image comes in, we perform face detection and feature extraction, and compare its feature to each face class stored in the database. There have been many researches and algorithms proposed to deal with this classification problem. There are two general applications of face recognition, one is called identification and another one is called verification.

1.4 INTERFACE IN TKINTER:

The interface has been made in python using tKinter framework. This includes the above mentioned modules.

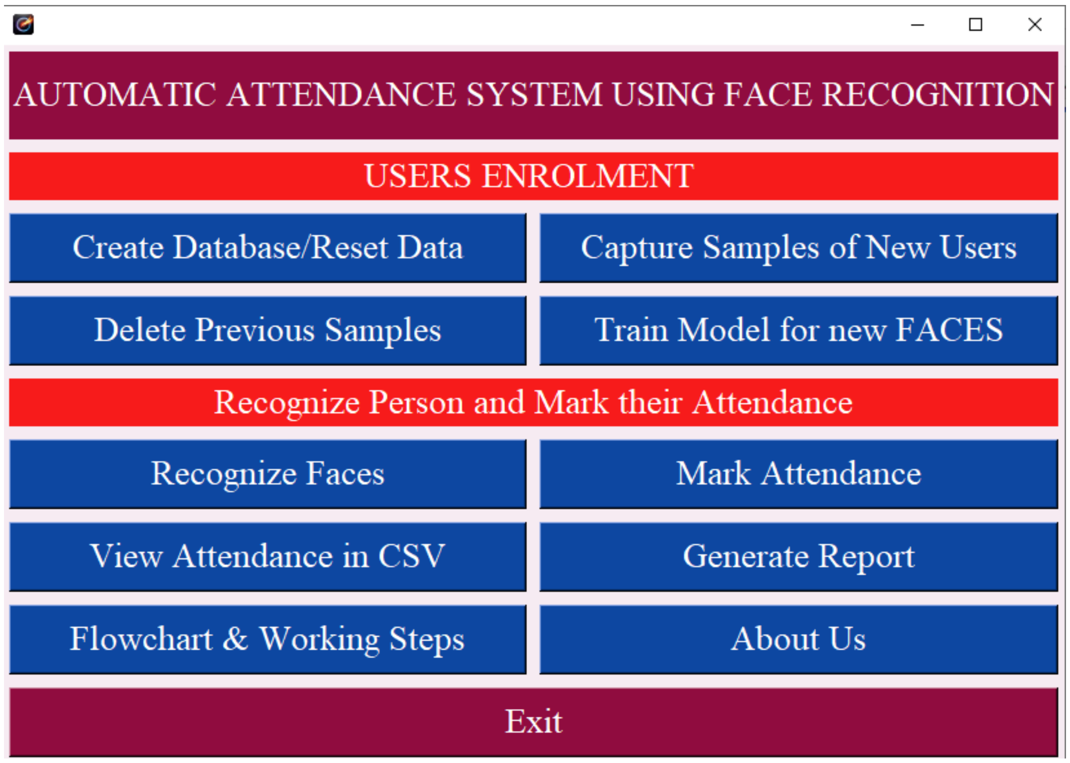

AUTOMATIC ATTENDANCE SYSTEM USING FACE RECOGNITION
USERS ENROLMENT
Create Database/Reset Data
Capture Samples of New Users
Delete Previous Samples
Train Model for new FACES
Recognize Person and Mark their Attendance
Recognize Faces
Mark Attendance
View Attendance in CSV
Generate Report
Flowchart & Working Steps
About Us
Exit

Fig 1.4: Project interface in tKinter

CHAPTER 3

RESULTS AND DISCUSSION

OBJECTIVE: Basic aim of the software is to computerized the campus of the college. I have designed the software for automate the attendance management in campus using Machine Learning i.e. Face Recognition.

3.1 SYSTEM DESIGN

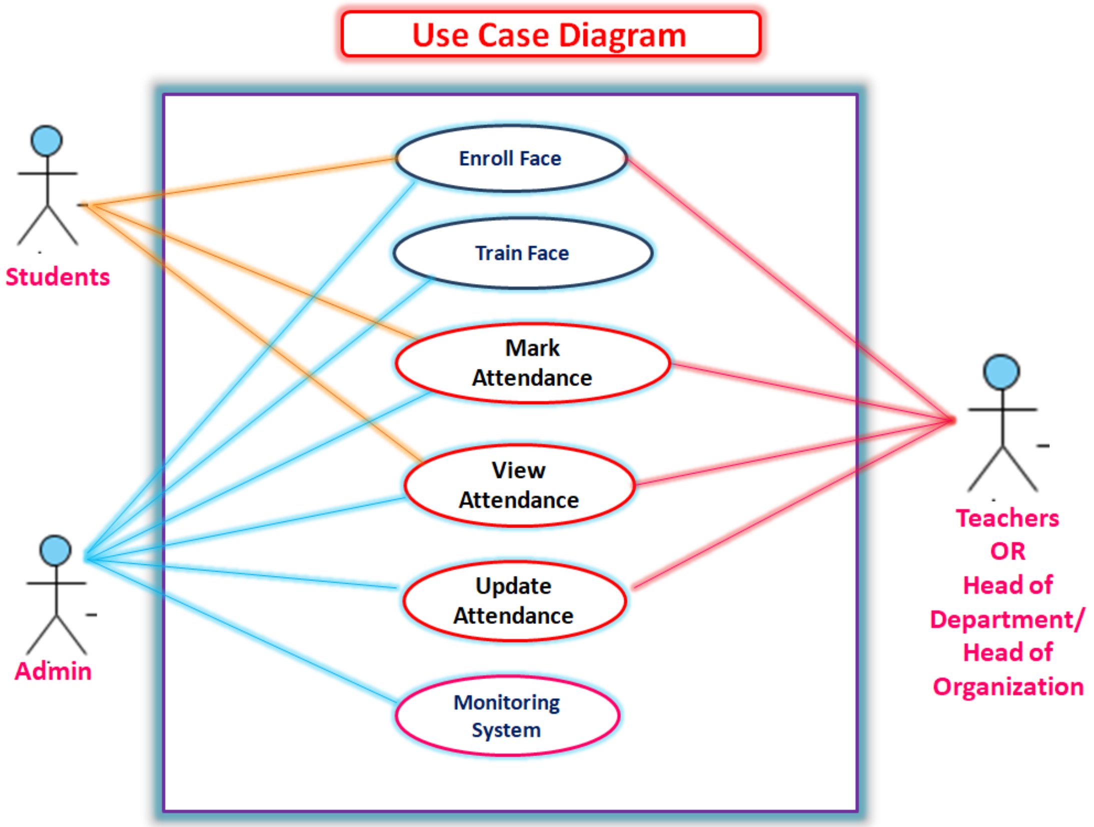


Figure 3.1: Use Case Diagram

3.2 MODULES

Automatic Attendance Management System using face recognition is the task of identifying already enrolled students as a present or absent face. Often the problem of face recognition is confused with the problem of face detection, Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

The project consists of major modules:

1) Create/Reset Database
2) Capture Samples
3) Train Faces
4) Face Detection
5) Face Recognition
6) Mark Attendance
7) View Attendance
8) Flowchart and Presentation(using Pyhton only)
9) Technology Stack
10) About Us

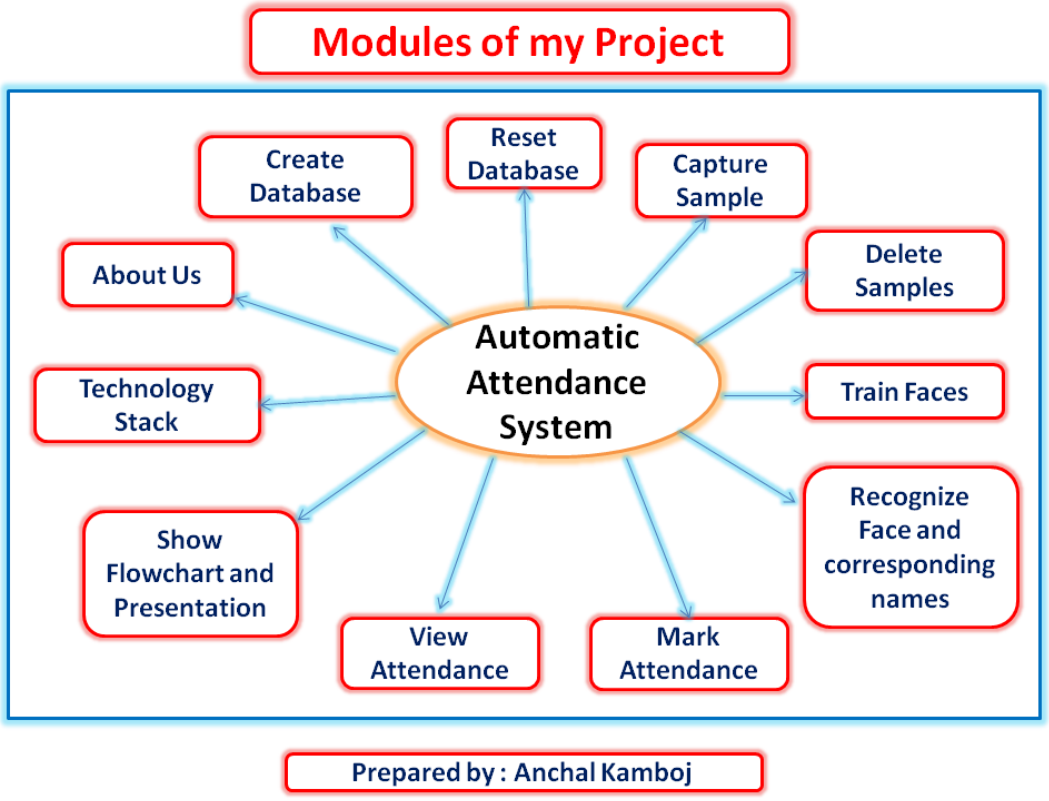
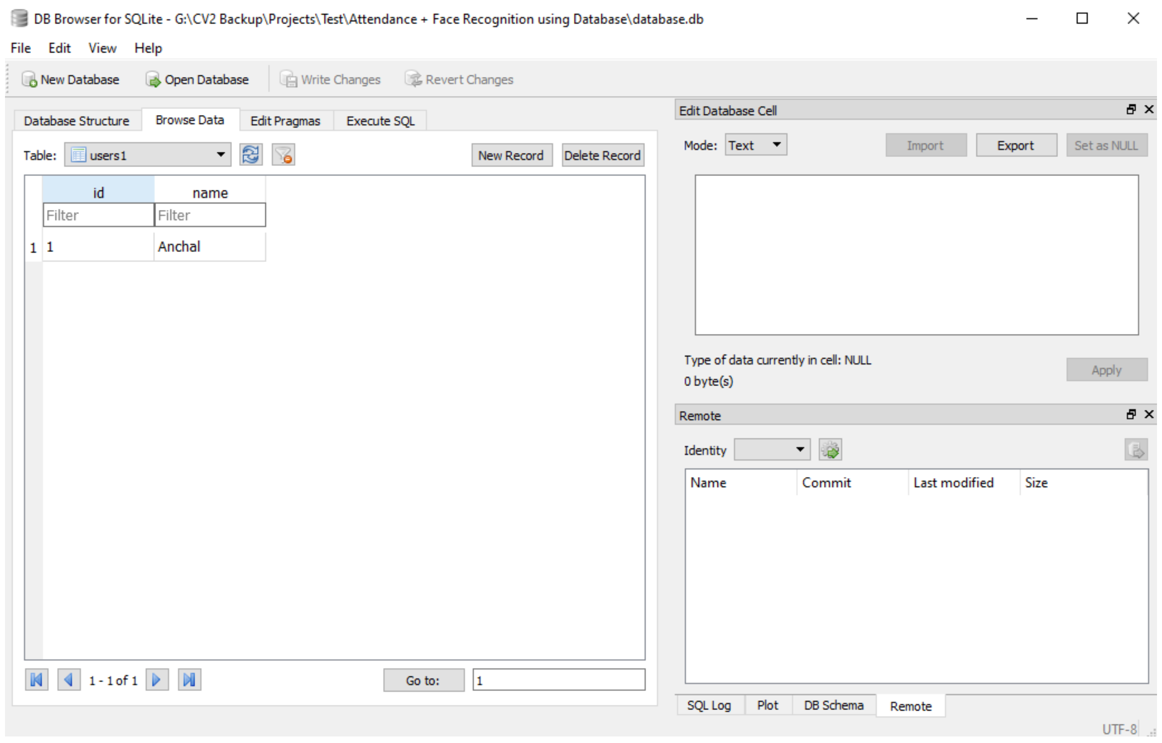


Figure 3.2: Project Modules

3.2.1 Create/Reset Database:

In this process, basically we create a Database named “database.db” in SQLite3. This database consists of one table named “users”. It contains two fields: name of the student and its Roll Number. This helps in marking attendance at the time of face recognition. This is a onetime process used to maintain record of all the students to get present or absent marked in all the lectures. In the picture shown below, there is only one entry. Similarly, all the students can be enrolled accordingly with their samples.



	id	name
	Filter	Filter
1	1	Anchal

Figure 3.3: Create Database in SQLite

3.2.2 Enroll Faces:

In this process, we capture the samples of all the students. These samples are required to enroll all the students, so that, those students can be recognized further to mark their respective attendance. We take at least 20 samples of each and every student for better accuracy.

3.2.3 Train Faces

After enrolling i.e. taking samples of various students; we train the system for those student's faces. So that, when the students mark their respective attendance; our Automatic Attendance Management System recognizes that person.

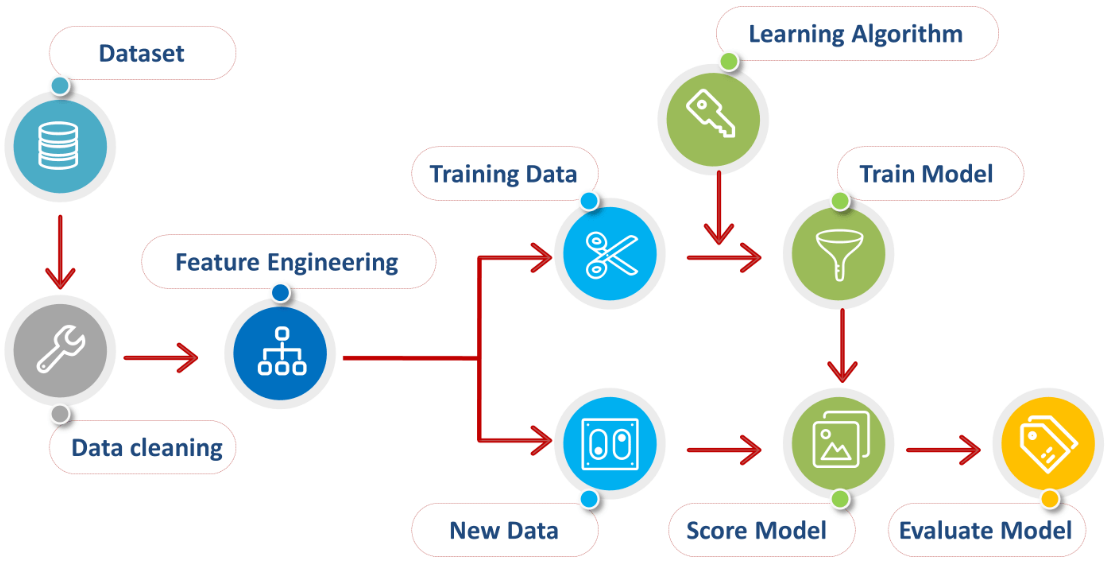


Figure 3.5: Method to Train Dataset

3.2.4 Face Detection:

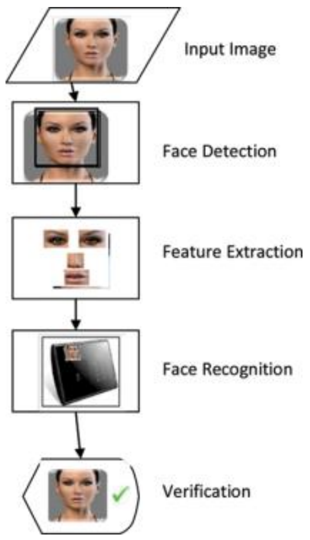


Figure 3.6: Face Detection & Recognition Flowchart

Face detection can consider a substantial part of face recognition operations. According to its strength to focus computational resources on the section of an image holding a face.

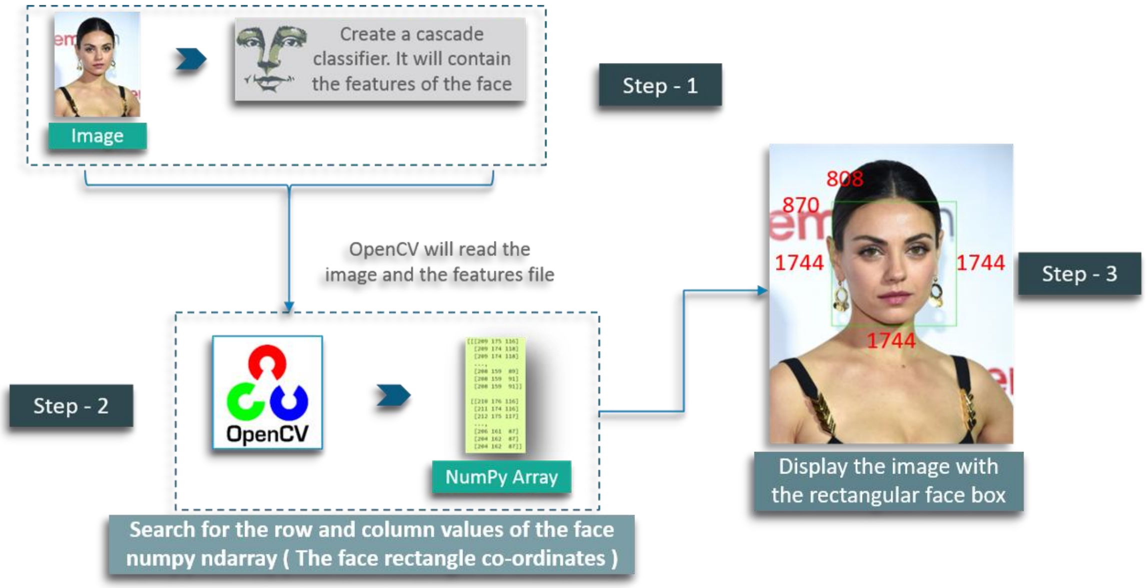


Image
Create a cascade classifier. It will contain the features of the face
Step - 1
OpenCV will read the image and the features file
Step - 2
OpenCV
NumPy Array
Search for the row and column values of the face numpy ndarray (The face rectangle co-ordinates)
808
870
1744
1744
1744
Step - 3
Display the image with the rectangular face box

Figure 3.7: Face Detection Process

The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin colour, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution.

Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is used to detect faces in real time for surveillance and tracking of person or objects. An ideal face detector would therefore be able to detect the presence of any face under any set of lighting conditions, upon any background. The face detection task can be broken down into two steps. The **first step** is a classification task that takes some arbitrary image as input and outputs a binary value of yes or no, indicating whether there are any faces present in the image. The **second step** is the face localization task that aims to take an image as input and output the location of any face or faces within that image as some bounding box with (x, y, width, height).

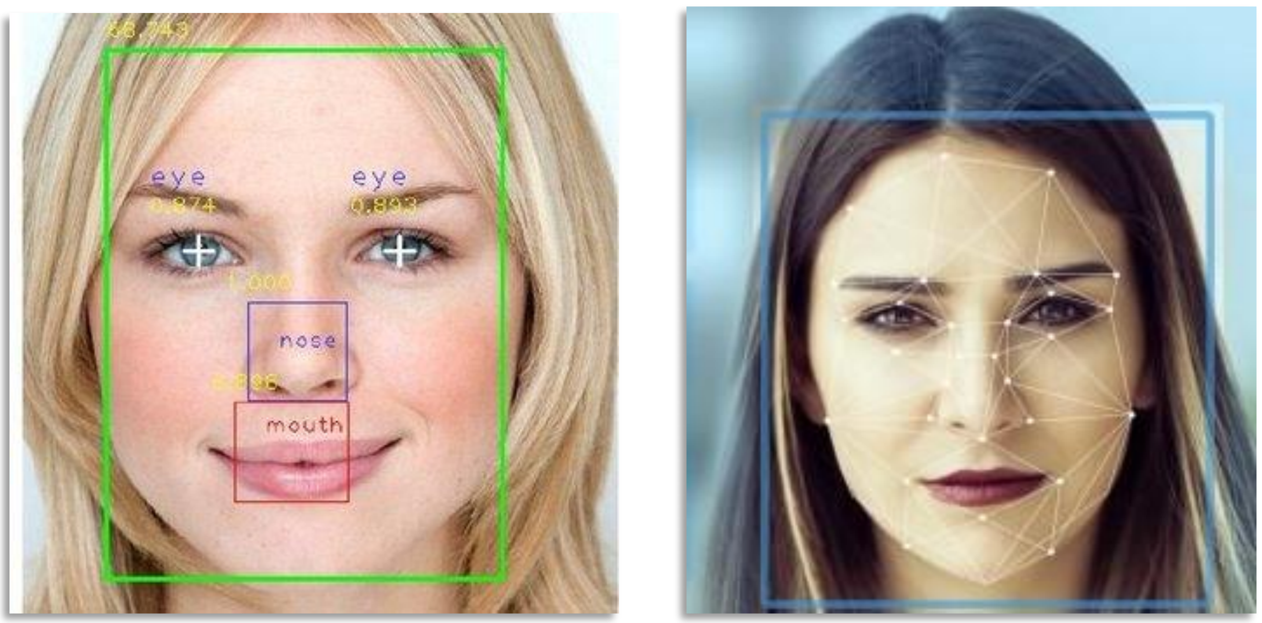


Figure 3.8: Extract Face Feature & Draw Rectangle

How the Face Detection Works:-

There are many techniques to detect faces, with the help of these techniques; we can identify faces with higher accuracy. These techniques have an almost same procedure for Face Detection such as OpenCV, Neural Networks, Matlab, etc. The face detection work as to detect multiple faces in an image. Here we work on OpenCV for Face Detection, and there are some steps that how face detection operates, which are as follows-

Face Recognition Algorithm used: **- Harrcascade algorithm**

Step1:- Firstly the **image is imported** by providing the location of the image. Then the picture is **transformed** from **RGB to Grayscale** because it is **easy to detect faces** in the grayscale.

Step2:- After that, the **image manipulation** used, in which the **resizing, cropping, blurring** and **sharpening** of the images done if needed.

Step3:- The next step is **image segmentation**, which is used for **contour detection** or segments the multiple objects in a single image so that the classifier can **quickly detect the objects and faces** in the picture.

Step4:- The next step is to use **Haar-Like features algorithm**. This algorithm used for **finding** the **location of the human faces** in a frame or image. All human faces shares some universal properties of the human face like the eyes region is darker than its neighbour pixels and nose region is brighter than eye region.

Step5:- The haar-like algorithm is also used for feature selection or **feature extraction** for an object in an image, with the help of edge detection, line detection, centre detection for **detecting eyes, nose, mouth, etc.** in the picture. It is used to select the essential features in an image and extract these features for face detection.

Step6:- The next step is to give **the coordinates of x, y, w, h** which makes a rectangle box in the picture to **show the location of the face** or we can say that to show the region of interest in the image. After this, it can make a **rectangle box** in the **area of interest** where it detects the face.

There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.

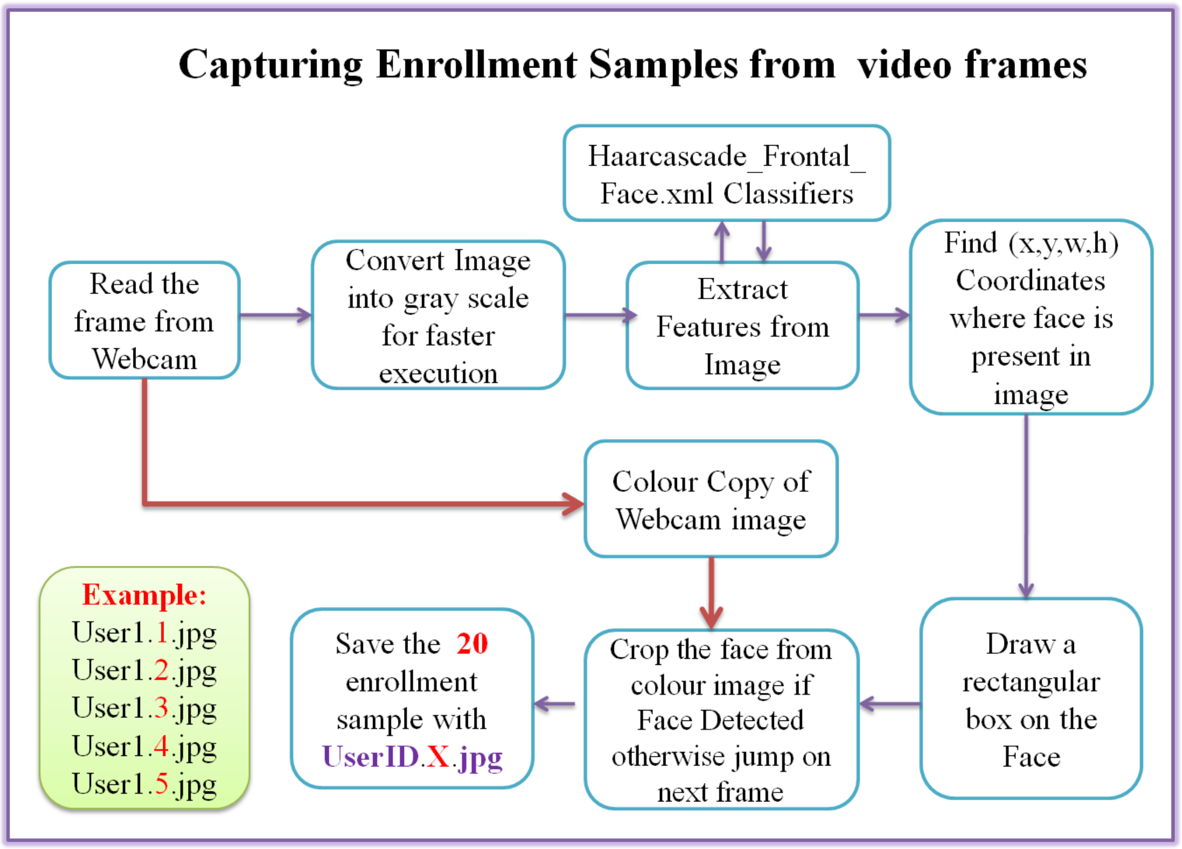

Capturing Enrollment Samples from video frames
Haarcascade_Frontal_ Face.xml Classifiers
Read the frame from Webcam
Convert Image into gray scale for faster execution
Extract Features from Image
Find (x,y,w,h) Coordinates where face is present in image
Colour Copy of Webcam image
Example:
User1.1.jpg
User1.2.jpg
User1.3.jpg
User1.4.jpg
User1.5.jpg
Save the 20 enrollment sample with UserID.X.jpg
Crop the face from colour image if Face Detected otherwise jump on next frame
Draw a rectangular box on the Face

Figure 3.9: Flowchart of Capturing Enrollment Samples

How to Run Face Detector in Real-Time (Webcam):-

(Requirement for Running the code- Python, OpenCV, Webcam, Numpy)

```
#import libraries
import cv2
import numpy as np

#import classifier for face and eye detection
face_classifier =
cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

# Import Classifier for Face and Eye Detection
face_classifier =
cv2.CascadeClassifier(‘Haarcascades/haarcascade_frontalface_default.xml’)
eye_classifier = cv2.CascadeClassifier (‘Haarcascades/haarcascade_eye.xml’)
def face_detector (img, size=0.5):

# Convert Image to Grayscale
gray = cv2.cvtColor (img, cv2.COLOR_BGR2GRAY)
faces = face_classifier.detectMultiScale (gray, 1.3, 5)
If faces is ():
return img

# Given coordinates to detect face and eyes location from ROI
for (x, y, w, h) in faces
x = x — 100
w = w + 100
y = y — 100
h = h + 100
cv2.rectangle (img, (x, y), (x+w, y+h), (255, 0, 0), 2)
roi_gray = gray[y: y+h, x: x+w]
roi_color = img[y: y+h, x: x+w]
eyes = eye_classifier.detectMultiScale (roi_gray)
for (ex, ey, ew, eh) in eyes:
cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,0,255),2)
roi_color = cv2.flip (roi_color, 1)
return roi_color

# Webcam setup for Face Detection
cap = cv2.VideoCapture (0)
while True:
ret, frame = cap.read ()
cv2.imshow (‘Our Face Extractor’, face_detector (frame))
if cv2.waitKey (1) == 13:   #13 is the Enter Key
break

# When everything done, release the capture
cap.release ()
cv2.destroyAllWindows ()
```

3.2.5 Face Recognition:

There are two predominant approaches to the face recognition problem: Geometric (feature based) and photometric (view based). As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature. Face recognition is the task of identifying an already detected object as a known or unknown face. Often the problem of face recognition is confused with the problem of face detection and Face Recognition on the other hand is to decide if the "face" is someone known, or unknown, using for this purpose a database of faces in order to validate this input face.

There are two predominant approaches to the face recognition problem:

1. **Geometric** (feature based)
2. **Photometric** (view based).

As researcher interest in face recognition continued, many different algorithms were developed, three of which have been well studied in face recognition literature. Recognition algorithms can be divided into two main approaches:

1.Geometric:

It is based on geometrical relationship between facial landmarks, or in other words the spatial configuration of facial features. That means that the main geometrical features of the face such as the eyes, nose and mouth are first located and then faces are classified on the basis of various geometrical distances and angles between features

2. Photometric stereo:

Used to recover the shape of an object from a number of images taken under different lighting conditions. The shape of the recovered object is defined by a gradient map, which is made up of an array of surface normal.

__Face Recognition Algorithm used:-__

Local Binary Patterns Histograms (LBPH).

LBPH considers texture descriptor which is useful to symbolize Faces. Because face data can be split as compositions of patterns of micro textures. Basically LBPH is carried out in 3 stages they are:-

1. Feature extraction

2. Matching

3. Classification

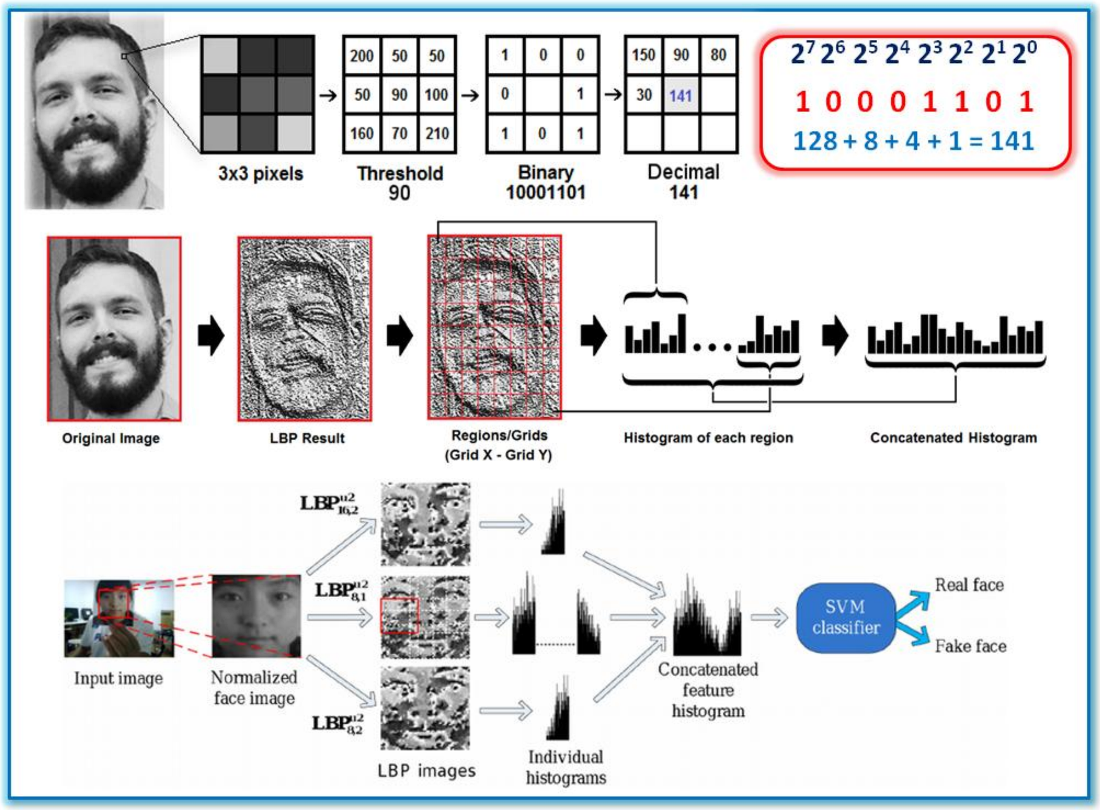


Figure 3.10: LBPH Working

The face recognition is carried out as stages first stage the image capturing and converting into grey scale then the haar features are checked if the features are their then it is considered as face if not non face, after that the pixels are mapped and checked the face. Today we discuss about one of the oldest (not the oldest one) and more popular face recognition algorithms: **Local Binary Patterns Histograms (LBPH).**

Local Binary Pattern(LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with a simple data vector. As LBP is a visual descriptor it can also be used for face recognition tasks, as can be seen in the following step-by-step explanation.

1) Step-by-Step

Now that we know a little more about face recognition and the LBPH, let’s go further and see the steps of the algorithm:

1. Parameters: the LBPH uses 4 parameters:

•**Radius:** the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.

•**Neighbors:** the number of sample points to build the circular local binary pattern.

•Sample points you include, the higher the computational cost. It is usually set to 8.

•**Grid X:** the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

•**Grid Y:** the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

2. Training the Algorithm: Here, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.

3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

2) OpenCV Library

Open CV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and Open CL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

3) Haar-cascade Detection in OpenCV

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 are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

How to Run Face Recognition in Real-Time (Webcam):-

(Requirement for Running the code- Python, OpenCV, Webcam, Numpy)

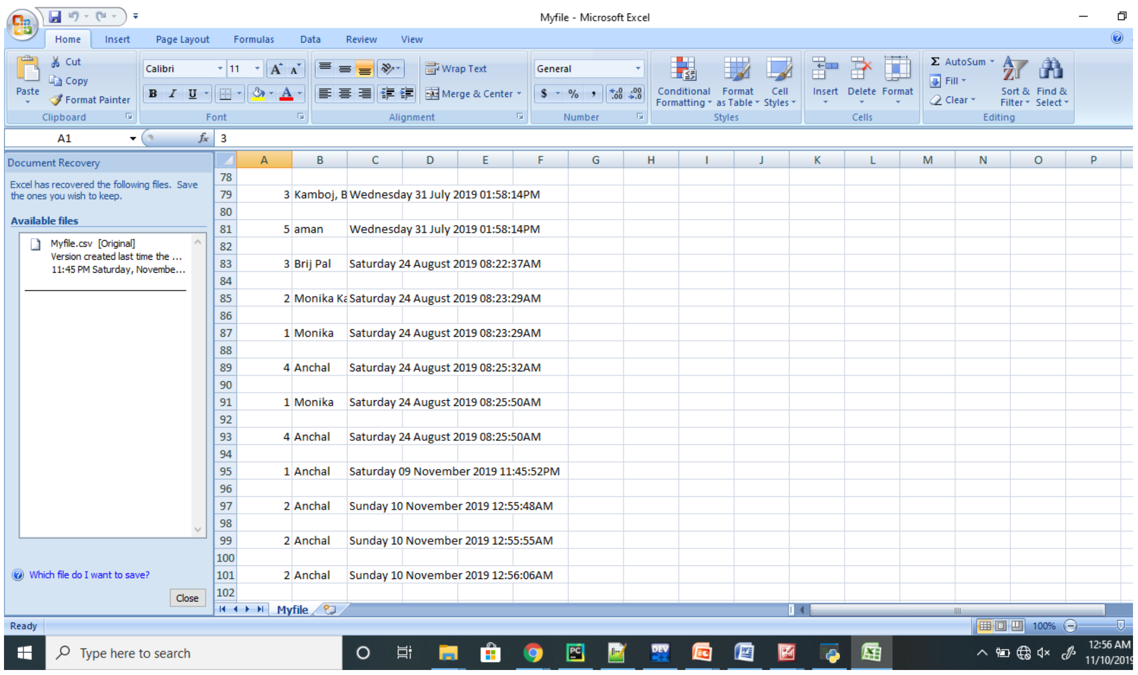
```
Import cv2
import numpy as np
import sqlite3
import os

conn = sqlite3.connect('database.db')
c = conn.cursor()

fname = "recognizer/trainingData.yml"
if not os.path.isfile(fname):
  print("Please train the data first")
  exit(0)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read(fname)

while True:
  ret, img = cap.read()
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  faces = face_cascade.detectMultiScale(gray, 1.3, 5)
  for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
    cv2.line(img, (x + 7, y - 9), (x + w - 10, y - 9), (0,              255, 0), 20)
    ids,conf = recognizer.predict(gray[y:y+h,x:x+w])
    c.execute("select name from users1 where id = (?);", (ids,))
    result = c.fetchall()
    name = result[0][0]
    if conf < 70:
      cv2.putText(img, name, (x + 4, y - 3), cv2.FONT_HERSHEY_DUPLEX, .6, (0, 0, 255),
lineType=cv2.LINE_AA)
    else:
      cv2.putText(img, 'No Match', (x + 4, y - 3),     cv2.FONT_HERSHEY_DUPLEX, .6, (255, 255, 255),
lineType=cv2.LINE_AA)
  cv2.imshow('Face Recognizer',img)

 if(cv2.waitKey(1)==27):
      break;

cap.release()
cv2.destroyAllWindows()
```

3.2.6 Mark Attendance:

Here mainly, after recognition of any student's face; it marks attendance of that particular student in the Database. The attendance is marked by student's Name, Roll Number and Timestamp. This is done in a CSV file. Thus, we can maintain records of all the users as per the date and time so as to ensure the lecture or time of entry of a particular person. This file can be provided with further restrictions, such that, only admin can access it. So as to make it secure and make further changes (if any).

3.2.7 View Attendance:

Here the students, faculty, parents, Head of Department, Head of Organization or any authorized person can view the attendance record of students. This attendance record is in CSV file. The attendance is marked by student's Name, Roll Number and Timestamp. Thus, we can maintain records of all the users as per the date and time so as to ensure the lecture or time of entry of a particular person. This file can be provided with further restrictions, such that, only admin can access it. So as to make it secure and make further changes.



A1 | 3

Document Recovery

Excel has recovered the following files. Save the ones you wish to keep.

Myfile.csv [Original]
Version created last time the ...
11:45 PM Saturday, Novembe...

	A	B	C
78			
79	3	Kamboj, B	Wednesday 31 July 2019 01:58:14PM
80			
81	5	aman	Wednesday 31 July 2019 01:58:14PM
82			
83	3	Brij Pal	Saturday 24 August 2019 08:22:37AM
84			
85	2	Monika Ka	Saturday 24 August 2019 08:23:29AM
86			
87	1	Monika	Saturday 24 August 2019 08:23:29AM
88			
89	4	Anchal	Saturday 24 August 2019 08:25:32AM
90			
91	1	Monika	Saturday 24 August 2019 08:25:50AM
92			
93	4	Anchal	Saturday 24 August 2019 08:25:50AM
94			
95	1	Anchal	Saturday 09 November 2019 11:45:52PM
96			
97	2	Anchal	Sunday 10 November 2019 12:55:48AM
98			
99	2	Anchal	Sunday 10 November 2019 12:55:55AM
100			
101	2	Anchal	Sunday 10 November 2019 12:56:06AM
102			

Figure 3.13: View Attendance in CSV/Excel File

3.2.8 Flowchart and Presentation

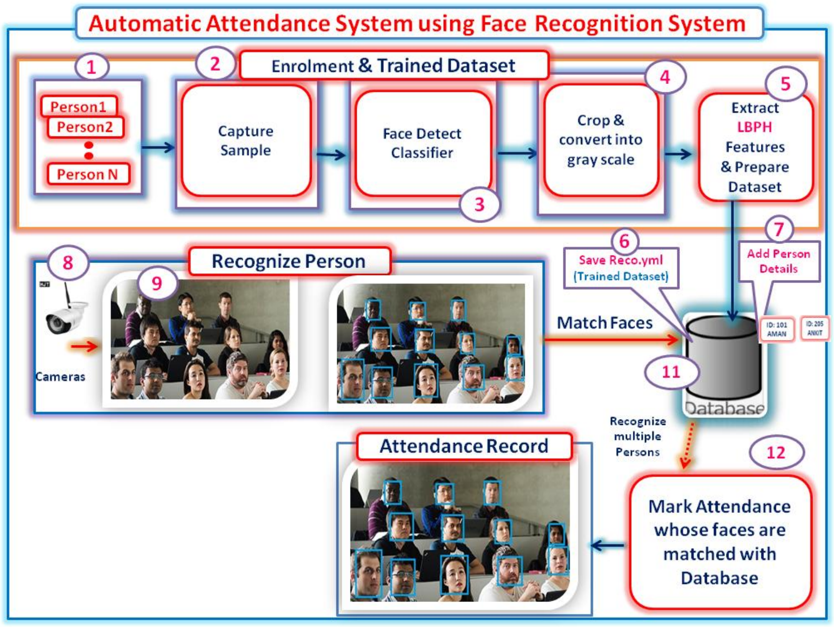

Automatic Attendance System using Face Recognition System
1
Person1
Person2
Person N
2
Enrolment & Trained Dataset
Capture Sample
Face Detect Classifier
3
4
Crop & convert into gray scale
5
Extract LBPH Features & Prepare Dataset
6
Save Reco.yml (Trained Dataset)
7
Add Person Details
8
Recognize Person
9
Cameras
Match Faces
11
Database
Recognize multiple Persons
12
Attendance Record
Mark Attendance whose faces are matched with Database

Figure 3.14: Flowchart and Presentation

This is a pictorial representation of our whole project in brief. This slideshow is made in Python itself. Even it doesn't require any presentation tool like Microsoft PowerPoint or anything else.

3.2.9 Technology Stack

In this project we have used:

Front End: tKinter using Python
Back End: SQLite
Reports: CSV Files
Versions: Python 3.6, SQLite 3, Windows 10
Packages Used: OpenCV, tKinter, Numpy

3.2.10 About Us

In this project, I have made a project on Automatic Attendance Management System under the supervision of Er. Brij Pal Kamboj. This project can be practically used inorder to mark attendance in various organizations. Yes, it can be further modified to make it more practical so as to sell it in the market to meet the future needs.