AUTOMATIC ATTENDANCE MANAGEMENT SYSTEM USING UNCONSTRAINED VIDEO BASED FACIAL RECONGITION

A MINOR PROJECT REPORT

Submitted by

Sai Rishyanth Visinigiri [RA2111026010280]
Suhas Ganga [RA2111026010281]
Manisha Manoj [RA2111026010274]

Under the Guidance of

Dr. N. Kanimozhi

Assistant Professor, Department of Computational Intelligence

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

with specialization in Artificial Intelligence & Machine Learning

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR – 603203

MAY 2024

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203
CHENGALPATTU DISTRICT

BONAFIDE CERTIFICATE

Register No **RA2111026010281, RA2111026010280, RA2111026010274** certified to be the bonafide work done by **SUHAS GANGA, SAI RISHYANTH VISINIGIRI, MANISHA MANOJ** of III Year/VI Sem B. Tech Degree Course in **ARITIFICIAL INTELLIGENCE 18CSEC305J** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur during the academic year 2022 – 2023.

FACULTY-IN-CHARGE
Dr. N Kanimozhi
Assistant Professor,
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

HEAD OF THE DEPARTMENT
Dr. R Annie Uthra
Professor and Head,
Department of Computational Intelligence
SRM Institute of Science and Technology
Kattankulathur Campus, Chennai

ABSTRACT

Face recognition technology has revolutionized attendance management in educational institutions and workplaces, offering advantages such as reduced human errors and accurate tracking. Leveraging Python's robust capabilities and the Local Binary Patterns Histograms (LBPH) algorithm, this project presents a sophisticated attendance system. LBPH, known for its high accuracy in extracting facial features, is utilized for precise face recognition. Coupled with a Relational Database Management System (RDBMS) for storing attendance data, the system ensures seamless tracking and analysis. The system utilizes Python for its ease of use and extensive libraries.

The LBPH algorithm is employed for facial recognition due to its ability to extract discriminative features. An RDBMS is integrated to store attendance data, enabling easy retrieval and analysis. The system is implemented in Python, utilizing libraries such as OpenCV for image processing and face recognition. LBPH algorithm is trained on a dataset of facial images to recognize individuals. An RDBMS, such as SQLite or MySQL, is used to store attendance records securely. The face recognition attendance system demonstrates high accuracy and efficiency in marking attendance. It effectively reduces manual errors and ensures timely tracking.

The integration with an RDBMS facilitates easy generation of attendance reports and analysis of attendance patterns. The developed face recognition attendance system offers a reliable and efficient solution for attendance management in educational institutions and workplaces. Leveraging Python, LBPH algorithm, and RDBMS, the system streamlines the process, leading to improved organizational efficiency and productivity.

Future enhancements may include real-time attendance marking, integration with existing HR systems, and implementation of advanced algorithms for improved accuracy and performance. Additionally, exploring the application of machine learning techniques for attendance prediction and anomaly detection could further enhance the system's capabilities.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ABBR	FULL FORM
FR	FACE RECOGNITION
LBPH	LOCAL BINARY PATTERN HISTOGRAM
UAV	UNMANNED AERIAL VEHICLE
CNN	CONVOLUTIONAL NEURAL NETWORK
BMC-LBPH	BILATERAL MEDIAN CONVOLUTION – LBPH
LUDB	LAMAR UNIVERSITY DATABASE
SVM	SUPPORT VECTOR MACHINE
MTCNN	MULTI-TASK CASCADED CONVOLUTIONAL NETWORK
BMC	BILATERAL MEDIAN CONVOLUTION
IOU	INTERSECTION OVER UNION

CHAPTER 1

INTRODUCTION

Implementing a face recognition attendance system offers a significant improvement over traditional methods in terms of accuracy, efficiency, and convenience. Unlike manual attendance marking, which is prone to errors and can be time-consuming, the face recognition system automates the process.

By leveraging image processing techniques, the system captures students' facial features and matches them against a pre-existing database of students/staff. This process occurs in real-time, allowing for instantaneous attendance tracking without the need for manual intervention. As a result, teachers can focus more on instructional activities rather than administrative tasks.

Furthermore, the system enhances security and accountability by ensuring that attendance records are tied directly to individual students. This eliminates the possibility of students signing in on behalf of absent classmates or engaging in other forms of attendance fraud.

Additionally, the system provides valuable data for analysis and reporting purposes. Attendance records stored in the database can be easily retrieved and used to generate reports on individual student attendance, class attendance rates, and trends over time. This enables educators and administrators to identify patterns, intervene when necessary, and make data-driven decisions to improve overall attendance management.

Overall, the implementation of a face recognition attendance system represents a significant step forward in modernizing attendance management practices in educational institutions. It not only addresses the shortcomings of traditional methods but also offers numerous benefits in terms of accuracy, efficiency, and data analytics capabilities.

CHAPTER 2

LITERATURE SURVEY

The Smart Attendance Monitoring System proposed in [1] utilizes facial recognition technology to automate attendance tracking in classrooms. By considering factors such as facial expression and lighting conditions, this system enhances accuracy and reliability compared to manual methods.

In [2], the Attendance System Using Face Recognition and Class Monitoring System uploads facial images to a central database for automated attendance tracking. Integration with class monitoring systems reduces manual intervention, streamlining classroom management.

Meanwhile, the Automatic Attendance System Using Face Recognition [3] employs advanced algorithms like Viola-Jones and PCA for accurate attendance marking. By capturing images at class start and end times, this system ensures security and reliability throughout the session.

The Class Room Attendance System Using Facial Recognition System [4] introduces a 3D facial model for improved recognition accuracy, even in challenging environments. Real-time attendance recording enables timely evaluation of student performance.

In contrast, [5] presents an RFID-based attendance system reliant on RFID cards, offering convenience but raising security concerns.

Finally, the Wireless Iris Recognition Attendance Management System [6] introduces iris recognition technology for high accuracy, albeit facing challenges with iris topography variations. These systems showcase ongoing innovations in attendance management, promising enhanced efficiency and productivity in educational settings.

CHAPTER 3

SYSTEM ARCHITECUTRE AND DESIGN

The automatic attendance system proposed in the basis of face recognition consists of several components working together to achieve efficient attendance tracking. Here is an overview of the architecture and system design:

- **Camera Module:** The system begins with the camera module, which captures images of students as they come into view. This module is responsible for providing input images to the recognition and validation process.
- **Recognition and Validation Module**: Once an image is captured, it is processed by the recognition and validation module. This module utilizes face recognition algorithms to identify individuals from the captured images. Additionally, it performs validation checks to ensure the accuracy and authenticity of the recognition results.
- **Attendance Marking Module:** Upon successful recognition and validation of a student, the attendance marking module automatically marks the student's attendance. This module updates the attendance records in the system's database in real-time.
- **User Interface**: The system provides a user interface for interaction with users, typically accessed through a login interface. Users, such as teachers or administrators, can log in to the system to access its features and functionalities.
- **Home Page:** Upon successful login, the user is presented with the home page of the system. This page serves as the central hub for accessing various functionalities, including attendance reports, system settings, and other relevant information.

Block Diagram:

The proposed block diagram of the automatic attendance system illustrates the interaction between these components. It visually represents the flow of data and operations within the system, highlighting the dependencies and interactions between different modules.

The system architecture and design outlined above enable seamless and accurate attendance tracking using face recognition technology. Leveraging modern algorithms and user-friendly interfaces, the system provides an efficient solution for attendance management in educational

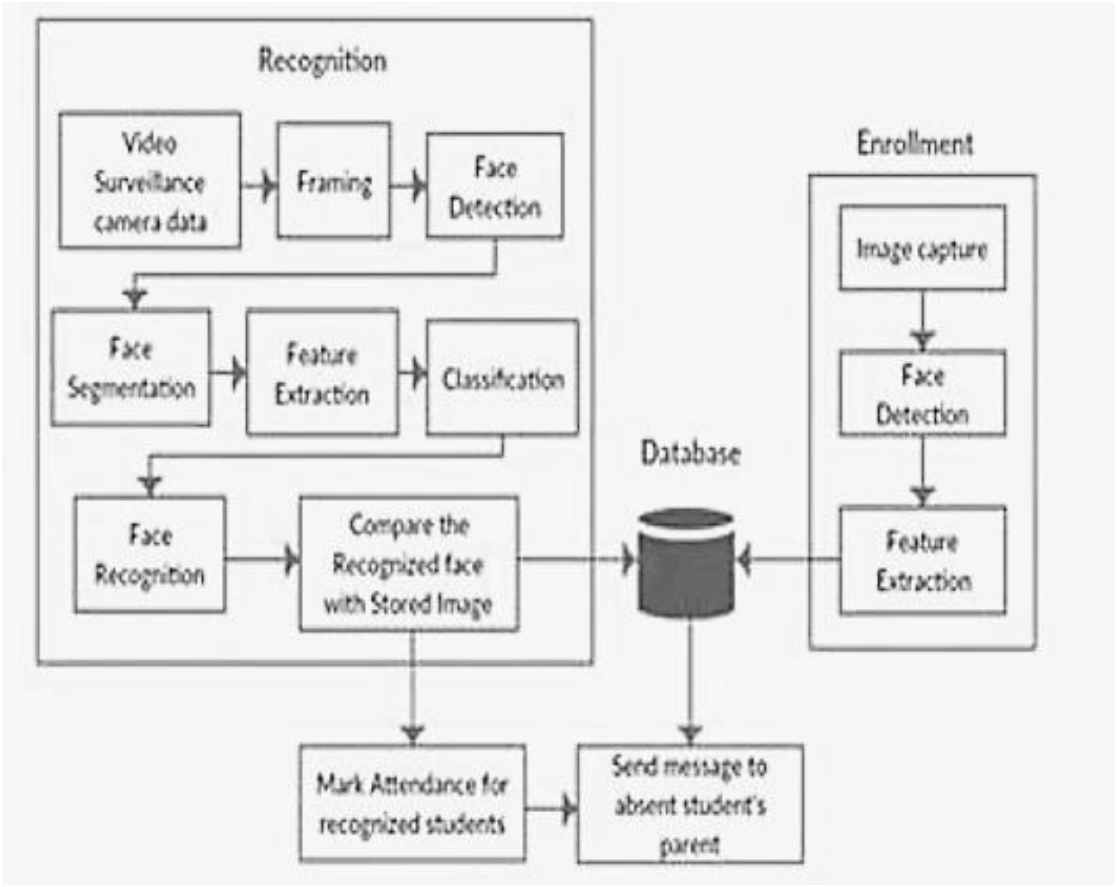

Figure 3.1 Bock Diagram for Automatic Attendance Management System

1. **Capturing the Image:**
   - A camera placed at the classroom entrance captures students' face images.

2. **Face Detection:**
   - Face detection algorithms identify the location and sizes of faces in the captured images.
   - Haar cascade classifier is used to detect faces within the images.

3. **Image Preprocessing:**
   - Preprocessing techniques are applied to enhance the quality of the captured images.
   - Conversion to grayscale improves image quality for better recognition accuracy.

4. **Training Set:**
   - A training set is prepared by comparing faces to be recognized with similar faces.
   - Algorithms are trained using this set to associate faces with specific individuals.

5. **Face Recognition:**
    - The system employs face recognition algorithms to identify and verify individuals from captured images.
    - Automatic identification and verification of persons are performed based on facial features.

6. **Attendance Marker:**
    - Upon successful recognition of a student's face, the attendance marker updates the attendance records.
    - A list of students present in the class is compiled based on matched faces from the date folder.
    - Students not matched with the present list are marked as absent in the attendance records.

Face Detection using Haar Cascade Classifier:

Paul Viola and Michael Jones proposed the Haar cascade classifier as an effective object detection method, which is based on machine learning. This approach involves analyzing positive and negative images to create a cascade that can be used to detect objects in other images. For face detection, a large dataset of positive and negative face images is required.

Haar features are used to analyze the characteristics present in an image. These features are evaluated within a 24x24 window in each operation. The sum of white region pixels is subtracted from the sum of black region pixels within the window, resulting in an integer value that determines the validity of the corresponding feature.

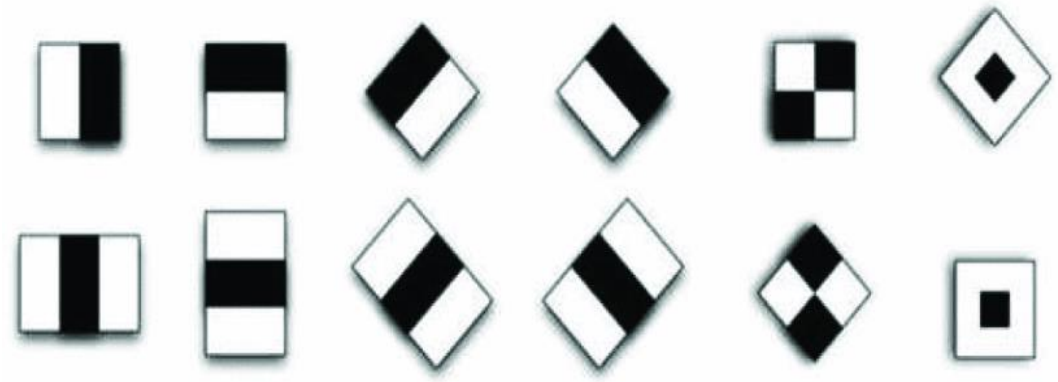

Figure 3.2 Haar Features

Figure 3.3 Haar Characteristics Applied on an Image

Face Recognition using Local Binary Pattern Histogram:

The local binary pattern (LBP) histogram method is employed for face recognition. This algorithm generates a new histogram for the input image and compares it with other histograms to find the best match. The label associated with the best match histogram is returned as the recognized face. To generate the LBP histogram, a 3x3 window moves across the image. At each position, the center pixel is compared with its neighboring pixels. If a neighbor pixel's intensity value is less than or equal to the center pixel, it is denoted as 1; otherwise, it is denoted as 0. Reading these values in a clockwise order within the 3x3 window results in a binary pattern, representing a local area of the image. This process is repeated for the entire image, resulting in a list of local binary patterns that are used for face recognition.

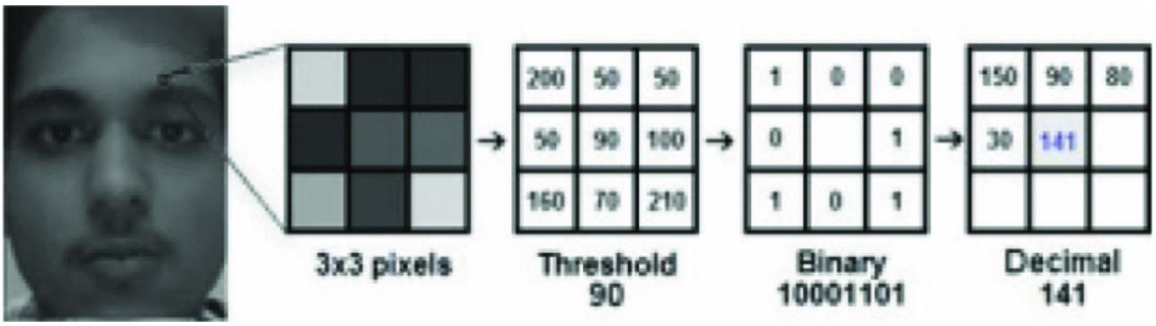

Figure 3.4 LBPH Algorithm for Face Identification

$$\int_{l}^{t} LBP\left(gp_x, gp_y\right) \sum_{p=0}^{p-1} s(gp - gc) \times 2^p$$

$$s(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases}$$

Figure 3.5 Mathematical Expression for LBPH Algorithm

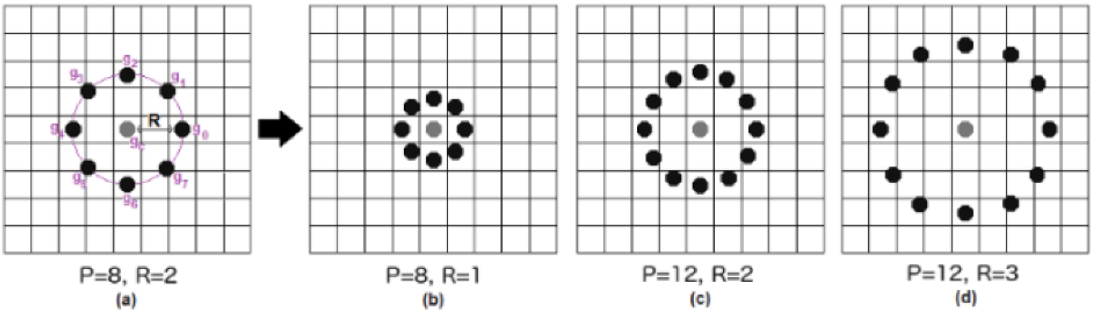

Figure 3.6 LBP Operation Radius Change

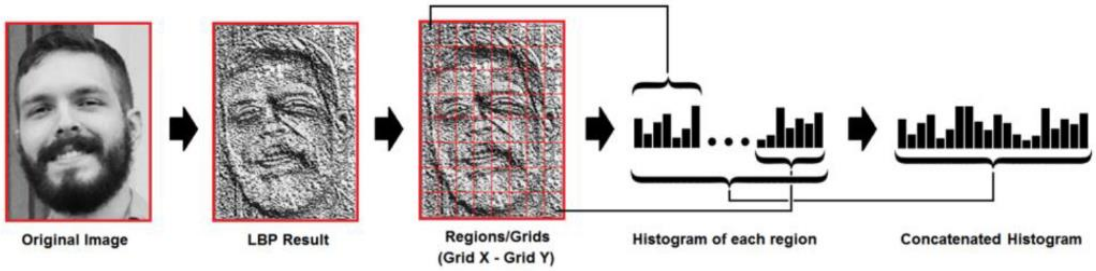

Figure 3.7 Extracting Histogram

CHAPTER 4

METHODOLOGY

1. **Input Image Capture:**

   A camera positioned at the classroom entrance captures images of students as they enter, providing input data for the attendance system.

2. **Convert Color Image to Grayscale:**

   The captured color images are converted to grayscale using appropriate image processing techniques. This conversion simplifies subsequent processing steps while retaining essential facial information.

3. **Face Detection using Haar Cascade Classifier:**

   Grayscale images undergo face detection utilizing the Haar cascade classifier algorithm. This algorithm analyzes the image data to identify regions likely to contain faces based on predefined patterns.

4. **Face Recognition using Local Binary Pattern Histogram:**

   Once faces are detected, the system performs face recognition using the local binary pattern (LBP) histogram method.

   Local binary patterns are computed for each detected face region, capturing texture information.

   These patterns are compared with pre-trained patterns stored in the system's database to determine potential matches.

5. **Face Matching with Trained Ones:**

   The recognized faces are matched with pre-trained faces stored in the system's database.

   Matching involves comparing features extracted from the detected face with features of known individuals.

   Matching scores or distances are calculated to assess the similarity between the detected face and the pre-trained faces.

6. **Attendance Marking:**

   If a recognized face matches a pre-trained face above a certain threshold, the system marks the corresponding student as "PRESENT" in the attendance data sheet.

   If no match is found or the similarity score falls below the threshold, the student is marked as "ABSENT."

   Attendance marking occurs in real-time as students enter the classroom.

7. **Generate Report:**

   The system generates comprehensive attendance reports based on the marked attendance data.

   Reports may include details such as attendance percentages, individual student attendance records, and trends over time.

   These reports assist educators and administrators in monitoring attendance patterns and identifying areas for improvement.

8. **Update Attendance:**

   The attendance data sheet is updated with the marked attendance, ensuring accurate and up-to-date records.

   This updated data serves as a valuable resource for various administrative and academic purposes.

9. **Continue Process:**

   The system continues to process incoming students, repeating steps 3 to 8 for each individual.

10. **Stop:**

    The attendance marking process concludes once all students have been processed, and the system halts until the next session.

Advantages:

- By utilizing advanced face detection and recognition techniques, the proposed system offers enhanced accuracy and efficiency in attendance management.
- Automation reduces the burden on educators and administrators, saving time and effort in manual attendance tracking.
- Detailed reports provide valuable insights into attendance patterns, facilitating data-driven decision-making and improving overall educational outcomes.

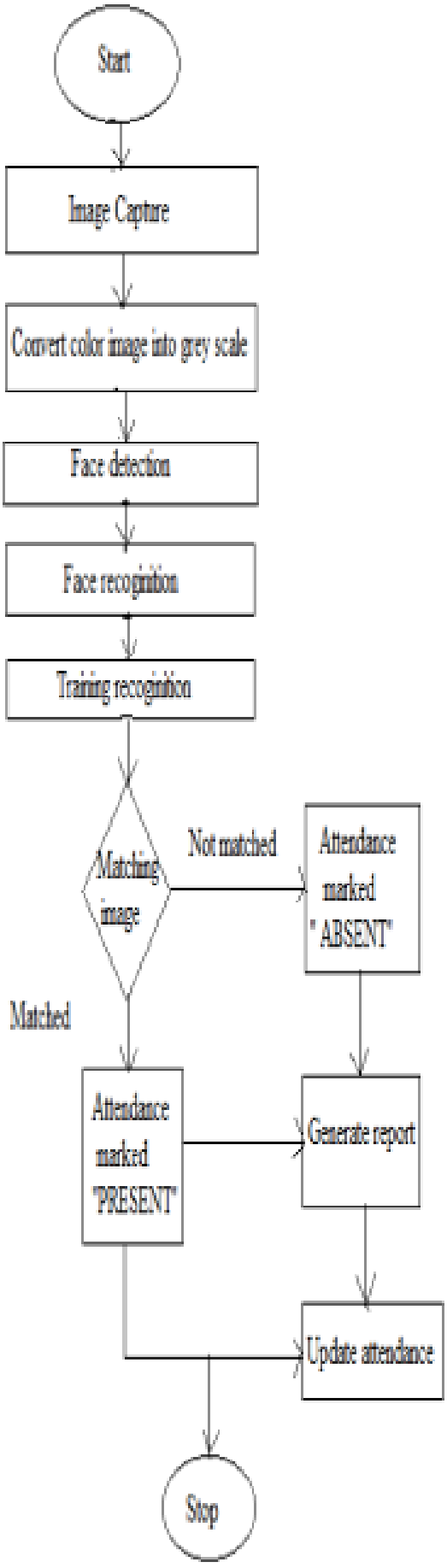

Figure 4.1 System Flow Diagram

CHAPTER 5

CODING AND TESTING

Required Files:

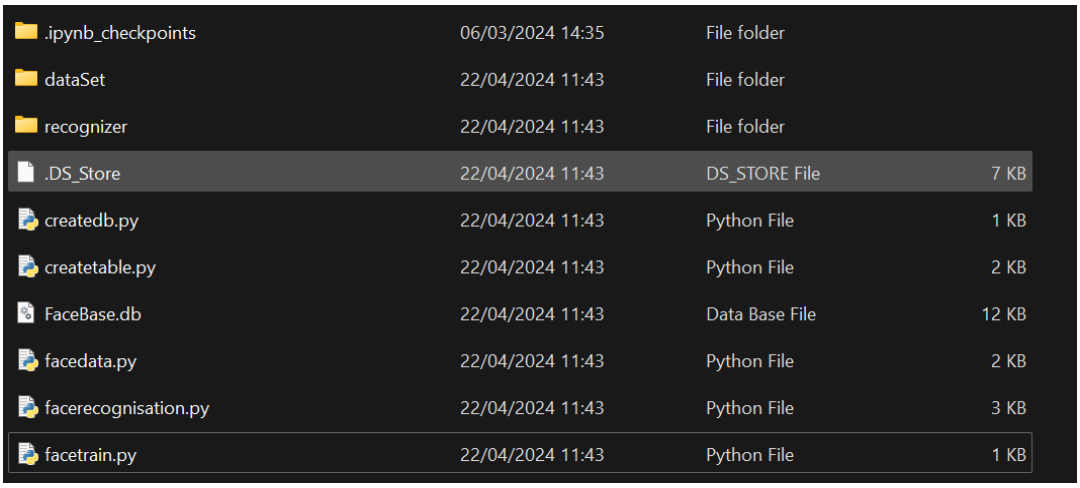

Figure 5.1 Required Files

- createdb.py and createtable.py: These files likely contain Python code to create a database and tables to store facial recognition data for users (students or employees).
- FaceBase.db: This is the database file created by the Python scripts to store facial data.
- facedata.py, facerecognisation.py, and facetrain.py: These files likely contain the core logic for facial recognition. They might be responsible for:
  Loading and processing images of faces (facedata.py).
  Recognizing faces in images or videos (facerecognisation.py).
  Training a facial recognition model on a dataset of user faces (facetrain.py).
- Other files:
  ipynb_checkpoints: This file is a checkpoint file created by Jupyter Notebook and not essential for the system.
  .DS_Store: This is a hidden file created by macOS and not relevant to the Python project.

Establishing Connection with FaceBase.db

In [1]:
```
import sqlite3
from sqlite3 import Error


def create_connection(db_file):
    """ create a database connection to a SQLite database """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print(sqlite3.version)
    except Error as e:
        print(e)
    finally:
        if conn:
            conn.close()


if __name__ == '__main__':
    create_connection("FaceBase.db")
```

```
2.6.0
```

Create Table in Connect Established Database

In [2]:
```
import sqlite3
from sqlite3 import Error


def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return conn
```

```
def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)
```

```
def main():
    database = "FaceBase.db"

    sql_create_projects_table = "CREATE TABLE IF NOT EXISTS People
    (ID text PRIMARY KEY,Name text NOT NULL,Age text NOT NULL,Gender text NOT NULL,CR text NOT NULL);"


    # create a database connection
    conn = create_connection(database)

    # create tables
    if conn is not None:
        # create projects table
        create_table(conn, sql_create_projects_table)

        # create tasks table
        # create_table(conn, sql_create_tasks_table)
    else:
        print("Error! cannot create the database connection.")


if __name__ == '__main__':
    main()
```

Implementation of Haar Cascade Classifier for Face Detection

The below Python script captures images of individuals using a webcam and detects their faces using the Haar cascade classifier provided by OpenCV. It then stores these face images in a dataset directory for further processing in an automatic attendance management system. Additionally, the script interacts with a SQLite database to insert or update information about individuals, such as their name, age, gender, and disciplinary records. Overall, this script serves as a data collection tool for building a face recognition-based attendance system, enabling the creation of a dataset for training and testing face recognition algorithms while also managing student information in a database.

```
n [1]: import cv2
       import numpy as np
       import sqlite3

       #faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
       faceDetect = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

       cam=cv2.VideoCapture(0);

       def insertOrUpdate(Id,Name,Age,Gen,CR):
           conn=sqlite3.connect("FaceBase.db")
           cmd="SELECT * FROM People WHERE ID="+str(Id)
           cursor=conn.execute(cmd)
           isRecordExist=0
           for row in cursor:
               isRecordExist=1
           if(isRecordExist==1):
               cmd="UPDATE People SET Name="+str(Name)+"WHERE ID="+str(Id)
               cmd2="UPDATE People SET Age="+str(Age)+"WHERE ID="+str(Id)
               cmd3="UPDATE People SET Gender="+str(Gen)+"WHERE ID="+str(Id)
               cmd4="UPDATE People SET CR="+str(CR)+"WHERE ID="+str(Id)
               conn.execute(cmd)
           else:
               params = (Id,Name,Age,Gen,CR)
               cmd="INSERT INTO People(ID,Name,Age,Gender,CR) Values(?, ?, ?, ?, ?)"
               cmd2=""
               cmd3=""
               cmd4=""
               conn.execute(cmd, params)

           conn.execute(cmd2)
           conn.execute(cmd3)
           conn.execute(cmd4)
           conn.commit()
           conn.close()

       Id=input('Enter User Reg No: ')
       name=input('Enter User Name: ')
       age=input('Enter User Age: ')
       gen=input('Enter User Gender: ')
       cr=input('Enter User Disciplinary Records: ')
       insertOrUpdate(Id,name,age,gen,cr)
       sampleNum=0
       while(True):
           ret,img=cam.read();
           gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
           faces=faceDetect.detectMultiScale(gray,1.3,5);
           for(x,y,w,h) in faces:
               sampleNum=sampleNum+1;
               cv2.imwrite("dataSet/User."+str(Id)+"."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
               cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
               cv2.waitKey(100);
           cv2.imshow("Face",img);
           cv2.waitKey(1);
           if(sampleNum>50):
               break;
       cam.release()
       cv2.destroyAllWindows()
```

Entering Student Details Manually into the Database

```
Enter User Reg No: RA2111026010280
Enter User Name: Sai Rishyanth Visinigiri
Enter User Age: 20
Enter User Gender: Male
Enter User Disciplinary Records: None
```

LBPH Face Recognition

Step 1: Import required libraries and modules

```
import os
import cv2
import numpy as np
from PIL import Image
import pysql
```

Step 2: Connect to the database

```
db = pymysql.connect(host='localhost', user='root', password='', db='attendance')
```

Step 3: Initialize Face Recognizer

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
path='dataSet'
```

Step 4: Load Trained Data for Face Recognition

```
def getImagesWithID(path):
    imagepaths=[os.path.join(path,f) for f in os.listdir(path)]
    faces=[]
    IDs=[]
    for imagepath in imagepaths:
        if ".DS_Store" in imagepath:
            print("Junk!")
        else:
            faceImg=Image.open(imagepath).convert('L');
            faceNp=np.array(faceImg,'uint8')
            ID=int(os.path.split(imagepath)[-1].split('.')[1])
            faces.append(faceNp)
            IDs.append(ID)
            cv2.imshow("training",faceNp)
            cv2.waitKey(10)
    return np.array(IDs),faces

IDs,faces=getImagesWithID(path)
recognizer.train(faces,IDs)
recognizer.save('trainningData.yml')
cv2.destroyAllWindows()
```

Step 5: Initialize Video Capture Object and Face Detector

```
def getProfile(id):
    conn=sqlite3.connect("FaceBase.db")
    cmd="SELECT * FROM People WHERE ID="+str(id)
    cursor=conn.execute(cmd)
    profile=None
    for row in cursor:
        profile=row
    conn.close()
    return profile

while(True):
    ret,img=cam.read();
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray,1.3,5);
    for(x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
        id,conf=rec.predict(gray[y:y+h,x:x+w])
        print(id)
        profile=getProfile(id)
        if(profile!=None):
            cv2.putText(img,"Name : "+str(profile[1]),(x,y+h+20),fontface, fontscale, fontcolor);
            cv2.putText(img,"Age : "+str(profile[2]),(x,y+h+45),fontface, fontscale, fontcolor);
            cv2.putText(img,"Gender : "+str(profile[3]),(x,y+h+70),fontface, fontscale, fontcolor);
            cv2.putText(img,"Criminal Records : "+str(profile[4]),(x,y+h+95),fontface, fontscale, fontcolor);

    cv2.imshow("Face",img);
    if(cv2.waitKey(1)==ord('q')):
        break;
cam.release()
cv2.destroyAllWindows()
```

Confidence Values of Face Identification

```
161
161
161
161
161
161
161
161
161
161
161
161
161
161
154
154
123
161
```

Step 6: Predict the face using LBPH and mark attendance

```
In [ ]: for (x, y, w, h) in faces:
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = img[y:y+h, x:x+w]
            id_, confidence = recognizer.predict(roi_gray)

            if confidence >= 45: # confidence threshold
                cursor = db.cursor()
                cursor.execute(f"SELECT name FROM students WHERE id={id_}")
                name = cursor.fetchone()[0]
                cv2.putText(img, name, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

                # insert attendance record in database
                cursor.execute(f"INSERT INTO attendance (student_id, date) VALUES ({id_}, CURRENT_DATE())")
                db.commit()

            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)

        cv2.imshow('Attendance System', img)
        if cv2.waitKey(1) == ord('q'): # press 'q' to quit
            break
```

Step 7: Release the video capture object and close the database connection

```
In [ ]: cap.release()
        cv2.destroyAllWindows()
        db.close()
```

This algorithm assumes that you have a relational database called attendance with two tables students and attendance. The students table has columns id and name, and the attendance table has columns student_id and date. You also need to train the LBPH face recognizer on a dataset of student's faces and save the trained data as trained_data.yml.

Dataset Description for Automatic Attendance Management System

Type: Image dataset

- **Image format:** Likely common formats like JPEG, PNG.
- **Image resolution:** Should be high enough for accurate facial recognition (typically at least 300x300 pixels).
- **Background:** Ideally, images should have a plain background for better face detection.
- **Data size**: The size of the dataset will depend on the number of individuals and the number of images per person. A larger dataset can improve the accuracy of the facial recognition model.
- **Data labeling:** While not always necessary, some systems might benefit from labeling the images with the individual's ID for easier identification during training.

Figure 5.2 Image Dataset for Automatic Attendance Management System

CHAPTER 6

SCREENSHOTS AND RESULTS

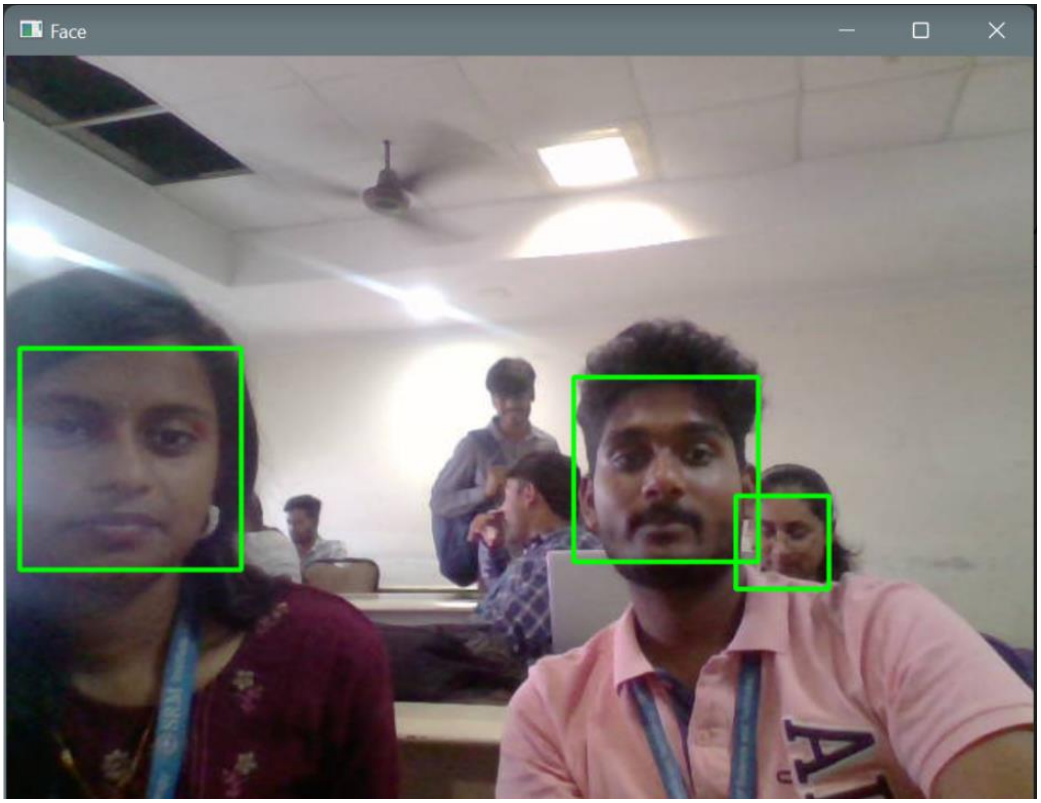

Figure 6.1 Haar Cascade Classifier for Face Detection

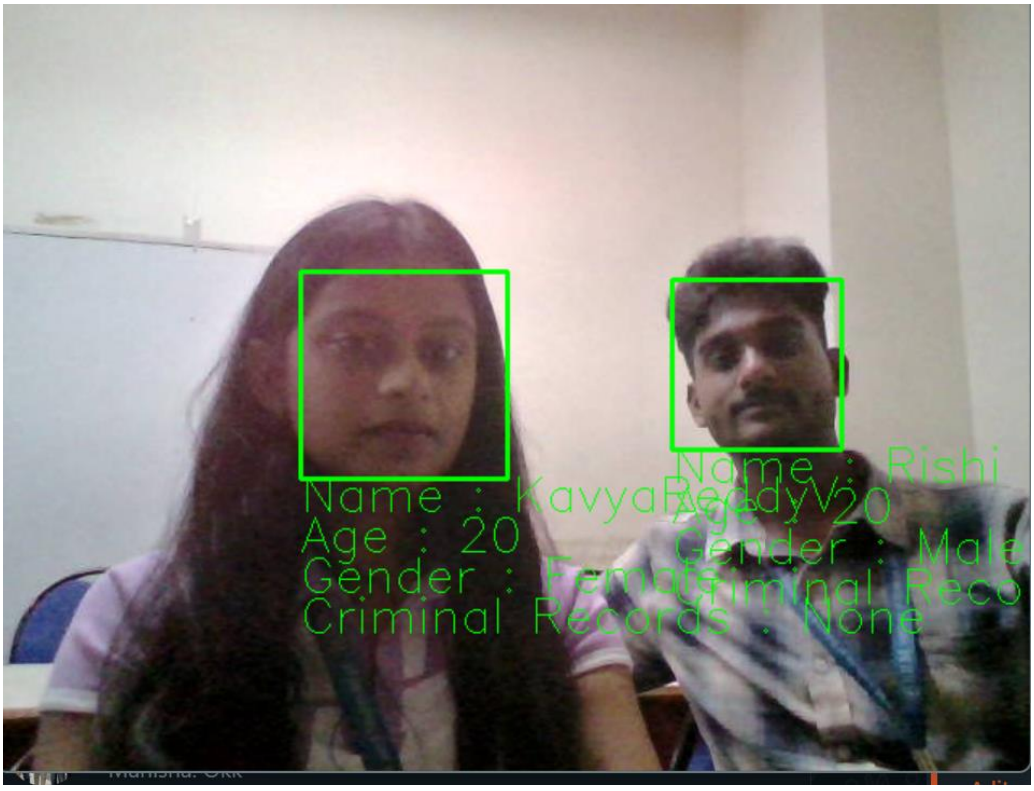

Figure 6.2.1 Face Recognition and Identification

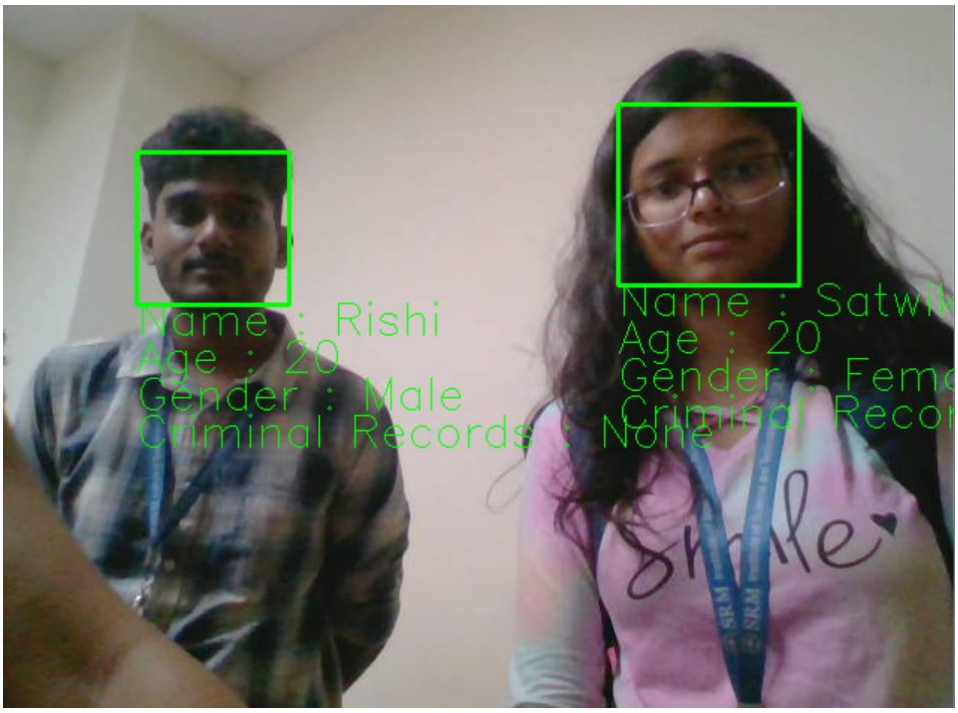

Figure 6.2.2. Face Recognition and Identification

A	B	C	D	E	F	G	H
StudentID	FullName	RollNumber	Gender	Department	Branch	Section	Status
1	Sai Rishyanth Visinigiri	RA2111026010280	Male	CINTEL	CSE AIML	Y1	Present
2	Suhas Ganga	RA2111026010281	Male	CINTEL	CSE AIML	Y1	Absent
3	Kavya Reddy Vutukuri	RA2111026010261	Female	CINTEL	CSE AIML	Y1	Present
4	Manisha Manoj	RA2111026010274	Female	CINTEL	CSE AIML	Y1	Absent
5	Sai Satwika	RA2111026010227	Female	CINTEL	CSE AIML	Y1	Present
6	Shanmukh Srinikeeth	RA2111026010217	Male	CINTEL	CSE AIML	Y1	Absent

Figure 6.3. Attendance stored in csv file

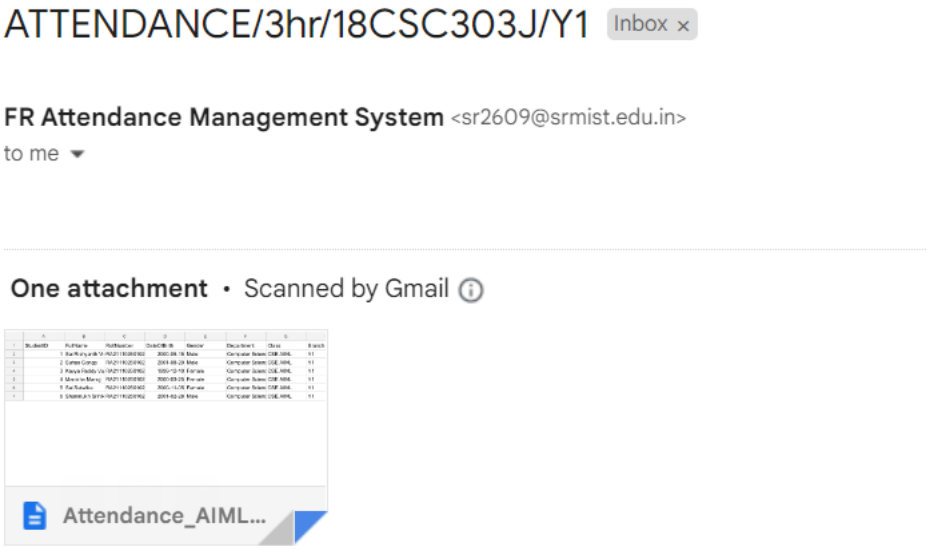

Figure 6.4 Auto-mail of Attendance to Faculty Members

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

The implementation of the attendance management system utilizing facial recognition technology has significantly streamlined the process of marking attendance for both lectures and students. By leveraging Python programming language and facial recognition techniques, the system offers improved accuracy and efficiency, particularly beneficial in scenarios involving a large number of students and lectures.

The system not only reduces the time and effort required for attendance marking but also enhances the overall management of student records. The integration of facial recognition technology ensures reliable attendance tracking, which can be further utilized for exam-related issues and academic monitoring.

Future Enhancements:

Despite the successful implementation of the attendance management system, there are several opportunities for further enhancement and expansion:

- **SMS Alert System:** Integrating a GSM module for sending SMS alerts to students regarding their attendance status can enhance communication and accountability. This feature ensures that students are promptly informed about their attendance, facilitating better engagement and accountability.
- **Parental Notification:** Extending the SMS alert system to notify parents or guardians about their child's attendance can foster stronger parent-school communication. Parents can stay informed about their child's attendance patterns and take proactive steps to support their academic progress.
- **Real-time Monitoring:** Implementing real-time monitoring capabilities allows administrators and educators to track attendance data continuously. This feature enables timely intervention in case of attendance discrepancies or issues, ensuring proactive management of attendance records.
- **Data Analysis and Reporting:** Implementing advanced data analysis tools and reporting features enables administrators to gain insights into attendance trends, identify patterns, and make data-driven decisions to improve attendance management and student engagement.

CHAPTER 8

REFERENCES

[1] Shubhobrata Bhattacharya, Gowtham Sandeep Nainala, Prosenjit Das and Aurobinda Routray, “Smart Attendance Monitoring System : AFace Recognition based Attendance System for Classroom
Environment”, 2018 IEEE 18th International Conference on Advanced Learning Technologies, pages 358-360,2018.

[2] Arun Katara, Mr. Sudesh, V.Kolhe,” Attendance System using Face Recognition and Class Monitoring System”, IJRITCC, Volume: 5,Issues: 2 ,February 2017.

[3] Ashish Choudhary, Abhishek Tripathi, Abhishek Bajaj, Mudit Rathi, and B.M Nandini, "Auttomatic Attendance System Using Face Recognition", International Journal of Modern Trends in Engineering
and Research , Volume: 03, Issue: 04, ISSN (Online):2349–9745; ISSN (Print):2393-8161,April 2016.

[4] Abhishek Jha, "Class room attendance system using facial recognition system", The International Journal of Mathematics, Science, Technology and Management (ISSN : 2319-8125) Volume: 2,Issue:
3,2014.

[5] T. Lim, S. Sim, and M. Mansor , "RFID based attendance system", Industrial Electronics Applications, 2009. ISIEA 2009. IEEE Symposium on, volume: 2, pages 778-782, IEEE 2009.

[6] S. Kadry and K. Smaili, “A design and implementation of a wireless iris recognition attendance management system”, Information Technology and control, volume: 36, no: 3, pages 323–329, 2007.

CHAPTER 9

APPENDIX (CODE)

createddb.py

```
import sqlite3
from sqlite3 import Error

def create_connection(db_file):
    """ create a database connection to a SQLite database """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print(sqlite3.version)
    except Error as e:
        print(e)
    finally:
        if conn:
            conn.close()

if __name__ == '__main__':
    create_connection("/Users/suhas/AML/Facial Recog/image-detection/FaceBase.db")
```

createtable.py

```
import sqlite3
from sqlite3 import Error
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return conn
def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

def main():
    database = "/Users/suhas/AML/Facial Recog/image-detection/FaceBase.db"

    sql_create_projects_table = "CREATE TABLE IF NOT EXISTS People (ID text
PRIMARY KEY,Name text NOT NULL,Age text NOT NULL,Gender text NOT NULL,CR
text NOT NULL);"

   # create a database connection
    conn = create_connection(database)

    # create tables
    if conn is not None:
        # create projects table
        create_table(conn, sql_create_projects_table)

        # create tasks table
        # create_table(conn, sql_create_tasks_table)
    else:
        print("Error! cannot create the database connection.")

if __name__ == '__main__':
    main()
```

facedata.py

```
import cv2
import numpy as np
import sqlite3

#faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
faceDetect = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

cam=cv2.VideoCapture(0);

def insertOrUpdate(Id,Name,Age,Gen,CR):
    conn=sqlite3.connect("FaceBase.db")
    cmd="SELECT * FROM People WHERE ID="+str(Id)
    cursor=conn.execute(cmd)
    isRecordExist=0
    for row in cursor:
        isRecordExist=1
    if(isRecordExist==1):
        cmd="UPDATE People SET Name="+str(Name)+"WHERE ID="+str(Id)
        cmd2="UPDATE People SET Age="+str(Age)+"WHERE ID="+str(Id)
        cmd3="UPDATE People SET Gender="+str(Gen)+"WHERE ID="+str(Id)
        cmd4="UPDATE People SET CR="+str(CR)+"WHERE ID="+str(Id)
        conn.execute(cmd)
    else:
        params = (Id,Name,Age,Gen,CR)
        cmd="INSERT INTO People(ID,Name,Age,Gender,CR) Values(?, ?, ?, ?, ?)"
        cmd2=""
        cmd3=""
        cmd4=""
        conn.execute(cmd, params)

    conn.execute(cmd2)
    conn.execute(cmd3)
    conn.execute(cmd4)
    conn.commit()
    conn.close()

Id=input('Enter User Id: ')
name=input('Enter User Name: ')
age=input('Enter User Age: ')
gen=input('Enter User Gender: ')
cr=input('Enter User Criminal Records: ')
insertOrUpdate(Id,name,age,gen,cr)
sampleNum=0
while(True):
    ret,img=cam.read();
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray,1.3,5);
    for(x,y,w,h) in faces:
        sampleNum=sampleNum+1;
        cv2.imwrite("dataSet/User."+str(Id)+"."+str(sampleNum)+".jpg",gray[y:y+h,x:x+w])
```

```
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
        cv2.waitKey(100);
    cv2.imshow("Face",img);
    cv2.waitKey(1);
    if(sampleNum>50):
        break;
cam.release()
cv2.destroyAllWindows()
```

facerecognisation.py

```
import cv2
import numpy as np
import sqlite3

#faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
faceDetect = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
cam=cv2.VideoCapture(0);
#rec=cv2.createLBPHFaceRecognizer();
rec = cv2.face.LBPHFaceRecognizer_create()
rec.read("recognizer/trainningData.yml")
#font=cv2.cv.InitFont(cv2.cv.CV_FONT_HERSHEY_COMPLEX,0.4,1,0,1)

fontface = cv2.FONT_HERSHEY_SIMPLEX
fontscale = 1
fontcolor1 = (0, 255, 0)
fontcolor2 = (0, 0, 255)
#cv2.putText(im, str(Id), (x,y+h), fontface, fontscale, fontcolor)

def getProfile(id):
    conn=sqlite3.connect("FaceBase.db")
    cmd="SELECT * FROM People WHERE ID="+str(id)
    cursor=conn.execute(cmd)
    profile=None
    for row in cursor:
        profile=row
    conn.close()
    return profile

while(True):
    ret,img=cam.read();
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray,1.3,5);
    for(x,y,w,h) in faces:
        id,conf=rec.predict(gray[y:y+h,x:x+w])
        print(id,conf)
        if conf<=23:
            profile=getProfile(id)
            if(profile!=None):
                cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
                cv2.putText(img,"Name : "+str(profile[1]),(x,y+h+20),fontface, fontscale,
fontcolor1);
                cv2.putText(img,"Age : "+str(profile[2]),(x,y+h+45),fontface, fontscale,
```

```
fontcolor1);
            cv2.putText(img,"Gender : "+str(profile[3]),(x,y+h+70),fontface, fontscale,
fontcolor1);
            cv2.putText(img,"Criminal Records : "+str(profile[4]),(x,y+h+95),fontface,
fontscale, fontcolor1);
        else:
            cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
            cv2.putText(img,"Name : Unknown",(x,y+h+20),fontface, fontscale, fontcolor2);
            cv2.putText(img,"Age : Unknown",(x,y+h+45),fontface, fontscale, fontcolor2);
            cv2.putText(img,"Gender : Unknown",(x,y+h+70),fontface, fontscale, fontcolor2);
            cv2.putText(img,"Criminal Records : Unknown",(x,y+h+95),fontface, fontscale,
fontcolor2);
    cv2.imshow("Face",img);
    if(cv2.waitKey(1)==ord('q')):
        break;
cam.release()
cv2.destroyAllWindows()
```

facetrain.py

```
import os
import time
import cv2
import numpy as np
from PIL import Image

#recognizer=cv2.createLBPHFaceRecognizer();
recognizer = cv2.face.LBPHFaceRecognizer_create()
path='dataSet'

def getImagesWithID(path):
    imagepaths=[os.path.join(path,f) for f in os.listdir(path)]
    faces=[]
    IDs=[]
    for imagepath in imagepaths:
        if ".DS_Store" in imagepath:
            print("Junk!")
        else:
            faceImg=Image.open(imagepath).convert('L');
            faceNp=np.array(faceImg,'uint8')
            ID=int(os.path.split(imagepath)[-1].split('.')[1])
            faces.append(faceNp)
            IDs.append(ID)
            cv2.imshow("training",faceNp)
            time.sleep(0.01)
            cv2.waitKey(10)
    return np.array(IDs),faces

IDs,faces=getImagesWithID(path)
recognizer.train(faces,IDs)
recognizer.save('recognizer/trainningData.yml')
cv2.destroyAllWindows()
```