International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 01 | January - 2023 Impact Factor: 7.185 ISSN: 2582-3930

ATTENDANCE BASED SYSTEM USING IMAGE RECOGNITION

Julakanti Vishnu Vardhan [1], Mrs. Doddi Srilatha[2], Ittamalla Samuel Moses [3],
Pinnapuralla Vijay Kumar [4]
[1,3,4] B.TECH Scholars, Dept.of Computer Science and Engineering Hyderabad-501301, India
[2] Associate Professor, Dept. of Computer Science and Engineering, SNIST, Hyderabad-501301,India

***Abstract*: - The Attendance System using Real-Time Facial Recognition is a reliable solution that can be used on a daily basis to manage student attendance. It involves using facial recognition technology to identify and recognize students' faces for attendance tracking, rather than relying on biometric information. In this particular face recognition project, the computer system is able to quickly and accurately detect and recognize human faces in images or videos captured by a surveillance camera. To improve the performance of the facial recognition system, numerous algorithms and techniques have been developed, with the LBPH approach being used in this case. This technique involves converting video frames into images, allowing the system to easily recognize students' faces and track their attendance, updating the attendance database automatically. The database used for storing the attendance records is SQLite, which is known for its efficiency, high performance, portability, reliability, accessibility, and compatibility with the Python software. Python is a suitable programming language for this attendance system due to the availability of extensive libraries and a large community for support and potential future developments.**

I INTRODUCTION

Attendance is a necessary parameter which is required is most of schools and colleges. On average this attendance is carried out to have accurate count of people or students seating in a particular classroom or any practice area. The traditional method of taking this attendance is carried out by humans where the lecturer or teacher manually counts each and every student with their required data like candidates name, serial number, status etc. This process is time consuming and maintenance of collected data is difficult. Thus we can digitize the process of taking attendance and make it simple and accurate. Our project will digitize the process of attendance collection by using image processing and will automatically update real time data of by accessing the database. In this project it is mostly software based with some hardware components. Initially we use a stationary camera which is situated above the classroom's entrance so that we can scan each and every students face without any interruption. Camera only provides captured real time image as an input to computer node 1 which is acting as a master processor. We use MATLAB as a platform for processing our input image. This image is processed and compared with preloaded image and as soon as image gets identified our algorithm indicated the master processor to transmit the identified person data to the computer node 2 which is faculties data base via IOT with help of Wi-Fi. As this is digitized process so the real time attendance is updated continuously at faculties data base.

II BACKGROUND STUDY (LITERATURE)

Attendance doesn't necessarily have to be narrowed to schools but can also be applied to offices, shopping centers, supermarkets. With the growing population, it is becoming increasingly difficult to keep track of everyone. Now in the digital era where time is essential, we have to discover methods of automating menial tasks so that we can better utilize the time somewhere else. Attendance is one such task that can easily be automated through the use of Image Recognition. It is present virtually everywhere. Right from classrooms from schools and institutions to offices and even factories. Attendance is important to maintain as it logs the amount of time spent doing an activity. Attendance is done in a manual manner which is time and energy consuming, therefore demanding the need for a way to take attendance. It becomes increasingly difficult to keep track as the number of attendees grow.

III METHODOLOGY

EXISTING SYSTEM

Attendance doesn't necessarily have to be narrowed to schools but can also be applied to offices, shopping centers, supermarkets. There are several existing systems which are used take attendance. We have the Fingerprint based attendance system. This is where a portable fingerprint device is used to mark the attendance. The fingerprints are configured with the students fingerprint beforehand. However, there is a severe disadvantage that the finger has to be completely dry and not be covered with any substance, otherwise the fingerprint scanner may not recognize the finger. In the Iris based attendance system, the student needs to stand in front of a camera, so that the camera will scan the Iris of the student. This is time consuming as the camera will take some time to gain focus and take a clear shot. Also the attendee has to remain still which itself is quite difficult in these days. Another method is to use RFID based tags in which it will be scanned similar to an ID card. This method is simple and efficient but can be easily exploited. A person can take any number of cards and scan the attendance for others.

PROPOSED SYSTEM

In the Proposed Attendance based System using Image Recognition we use the model which has been implemented through the Scale Invariant Feature Transform. The scale-invariant feature transform (SIFT) is a computer vision algorithm to detect, describe, and match local features in images. Although this model has numerous applications we limit to facial recognition.

IV ALGORITHM

In the SIFT framework, keypoints are the first points of interest that we find. The image is convolved using various Gaussian filter settings, and the difference of subsequent Gaussian-blurred images is subsequently captured. The maxima and minima of the Difference of Gaussians (DoG) that occur at various scales are then used as keypoints.

The K-Nearest Neighbor (KNN) algorithm, which is used most frequently for classification issues, is the second algorithm in use. During the training phase, KNN just stores the dataset without doing anything to it. When an input is provided for classification, the system sorts the input data into related data. The DoG is given by,

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma),$$

where L(x,y,kσ) is the convolution of the original image. The value of k is chosen so that we get a fixed number of convolved images per octave and the convolved images are grouped by octave. Following that, each octave, adjacent Gaussian-blurred photos are used to create the Difference-of-Gaussian images. Keypoints are determined as local minima/maxima of the DoG images across scales after DoG images have been collected.

The next step in the algorithm is to perform a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures. This information allows us to reject the points which are of low contrast or poorly localized along an edge.

First, for each candidate keypoint, interpolation of nearby data is used to accurately determine its position. This approach calculates the interpolated location of the extremum, It significantly enhances stability and matching. The interpolation is done using the quadratic Taylor expansion of the Difference-of-Gaussian scale-space function.

$$D(\mathbf{x}) = D + \frac{\partial D}{\partial \mathbf{x}}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

To discard the keypoints with low contrast, the value of the second-order Taylor expansion is computed at the offset. If the value is less than 0.03 the keypoint is discarded. In order to increase stability, we need to eliminate the keypoints that have poorly determined locations but have high edge responses. Each keypoint is assigned one or more orientations based on local image gradient directions. This is the key step in chieving invariance to rotation as the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation.

V ARCHITECTURE

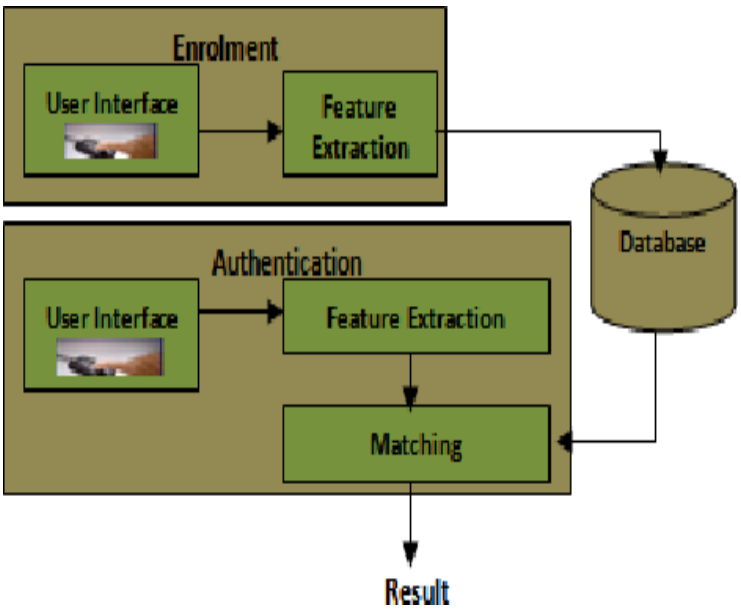

Fig 5.1 System architecture

Steps involved:

Step-1: Collecting the dataset.
Step-2: Pre-processing the dataset which involves data normalization encoding the dataset.
Step -3: Extracting the features that the models need to be trained on is done using attributes which have a correlation > 0.5 with the target attribute.
Step-4: Take a photo of the livefeed from the user interface
Step-5: Feed the input to the trained model and compare the faces already existing in the database.
Step-6: Update the necessary information in the database.

VI IMPLEMENTATION

6.1 UML DIAGRAMS

6.1.1 Use Case diagram

The Attendance based System using Image Recognition use case diagram represents how users interact with the system. It also represents how each module interacts with the other modules and also the relationbetween those modules

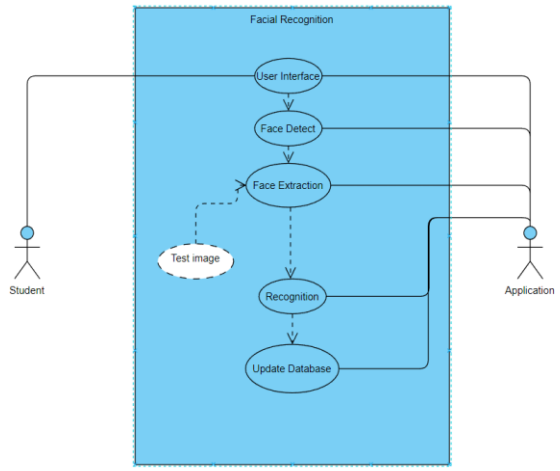

Fig 6.1.1 Use case diagram

6.1.2 Sequence diagram

The Attendance based System using Image Recognition sequence diagram shows the sequence in which each process occurs.

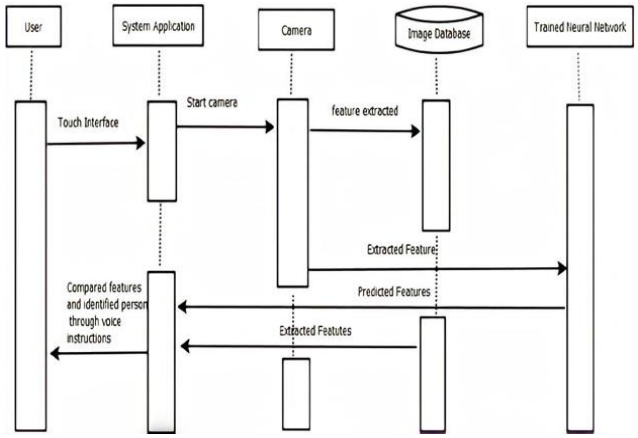

Fig 6.1.2 Sequence diagram

6.1.3 Class diagram

The Attendance based System using Image Recognition class diagrams are entirely used to represent the dynamic elements of a system. It represents a system's overall behaviour and explains the users and functionality of the system.

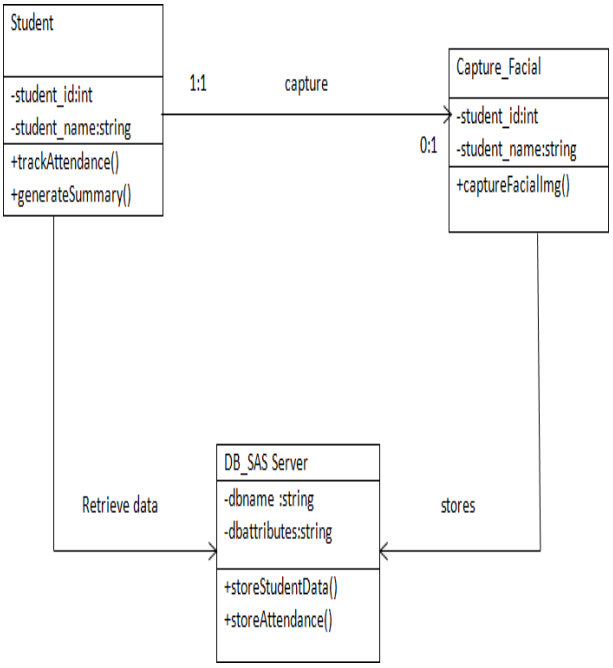

Fig 6.1.3 Class diagram

6.1.4 Activity diagram

The Attendance based System using Image Recognition uses an activity diagram. An activity diagram is a specific type of state chart diagram that shows the transition from one action to the next. The diagram that shows how a system behaves is called a behavioral diagram. The action is described as a system operation.

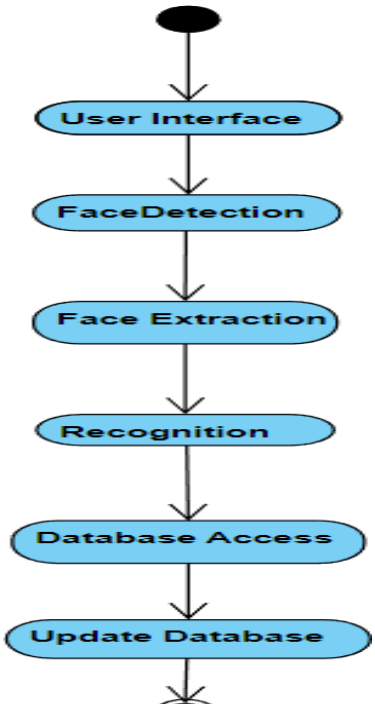

Fig 6.1.4 Activity diagram

6.2 Training module:

6.2.1 Gathering and Importing all the necessary libraries

```
from tkinter import *
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
from tkinter import ttk
import sqlite3
```

Fig 6.2.1 Importing Libraries

6.2.2 Declaring and Defining the global variables

```
nameList = []
infoList = {}
i = 1
```

Fig 6.2.2 Global Variable Declaration

6.2.3 Defining the directory which already contains the images of the students which is used for checking.

```
def hell():
    path = "ImagesAttendance"
    images = []
    classNames = []
    myList = os.listdir(path)
    global nameList
    global infoList
    for cl in myList:
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])
```

Fig 6.2.3 Directory Connection

6.2.4 Creation of Encoding function

```
def findEncodings(images):
    encodeList = []
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        encode = face_recognition.face_encodings(img)[0]
        encodeList.append(encode)
    return encodeList
```

Fig 6.2.4 Creating the Encoding Function

6.2.5 Main Function

```
encodeListKnown = findEncodings(images)
cap = cv2.VideoCapture(0)
while True:
    success, img = cap.read()
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)
    facesCurFrame = face_recognition.face_locations(imgS)
    encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)
    for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
        matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
        faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
        matchIndex = np.argmin(faceDis)
        if matches[matchIndex]:
            name = classNames[matchIndex].upper()
            y1, x2, y2, x1 = faceLoc
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
            markAttendaces(name)
    cv2.imshow('Webcam', img)
    cv2.waitKey(1)
    key = cv2.waitKey(50)
    if key == ord('s'):
        break
```

Fig 6.2.5 Main Function

6.2.6 Connecting to the SQLite Database and fetching data

```
conn = sqlite3.connect('attend.db')
c = conn.cursor()
c.execute("select NAME from attendance")
Dname = c.fetchall()
DnameSet = []
for name in Dname:
    DnameSet.append(name[0])
DnameSet = set(DnameSet)
ScanName = set(infoList.keys())
real_names = DnameSet.intersection(ScanName)
```

Fig 6.2.6 Connecting to SQLite Database and fetching data

6.2.7 Update database with new information

```
for name, time in infoList.items():
    subTime = time.split(':')
    hours, minutes = int(subTime[0]), int(subTime[1])
    if (minutes > 10):
        attended = 'ABSENT'
    else:
        attended = 'PRESENT'
    if (hours < 12):
        period = hours - 9 + 1
    elif (hours < 15):
        period = hours - 12
    else:
        period = "NULL"
    if name in real_names:
        c.execute(f'''UPDATE attendance SET TIME=? WHERE NAME=?''', (time, name))
        c.execute(f'''UPDATE attendance SET PERIOD=? WHERE NAME=?''', (period, name))
        c.execute(f'''UPDATE attendance SET PRESENT=? WHERE NAME=?''', (attended, name))
    else:
        c.execute(f'INSERT INTO attendance VALUES("{i}","{name}","{time}","{period}","{attended}")')
        i += 1
c.execute("select * from attendance")
d = c.fetchall()
print("RollNumber\t\tName\t\tTime\t\tPeriod")
for row in d:
    roll_number, name, time, period, present = row
    print(f'{str(roll_number)}\t\t\t\t{name.capitalize()}\t\t{time}\t{str(period)}\t{present}')
    conn.commit()
    conn.close()
```

Fig 6.2.7 Update database

6.2.8 User Interface

```
root = Tk()
root.geometry("300x200")
root.configure(bg='cadet blue')
label2 = Label(root, text="Attendance System", bg='cadet blue', font=("Arial", 20), foreground='purple')
label2.place(x=35, y=10)
button1 = Button(root, text="Take Attendance", command=hell, bg='cadet blue', activebackground='orange')
button1.place(x=50, y=80)
button2 = Button(root, text="Exit", command=root.destroy, bg='cadet blue', activebackground='red')
button2.place(x=145, y=120)
button3 = Button(root, text="Show Attendance", command=View, bg='cadet blue', activebackground='green')
button3.place(x=175, y=80)
root.mainloop()
```

Fig 6.2.8 Update database

6.2.9 Displaying Database Information

```
def View():
    root1 = Tk()
    root1.title("Database Attendance")
    root.title("Attendance")
    tree = ttk.Treeview(root1, show='headings')
    tree.column("#1", anchor=CENTER)
    tree.heading("#1", text="Roll Number")
    tree.column("#2", anchor=CENTER)
    tree.heading("#2", text="Name")
    tree.column("#3", anchor=CENTER)
    tree.heading("#3", text="Time")
    tree.column("#4", anchor=CENTER)
    tree.heading("#4", text="Period")
    tree.column("#5", anchor=CENTER)
    tree.heading("#5", text="Present")
    tree.pack()
    con1 = sqlite3.connect("attend.db")
    cur1 = con1.cursor()
    cur1.execute("SELECT * FROM attendance")
    rows = cur1.fetchall()
    for row in rows:
        tree.insert("", END, values=row)
        con1.close()
        root1.mainloop()
```

Fig 6.2.9 Update database

VII OUTPUTS

A. User Interface

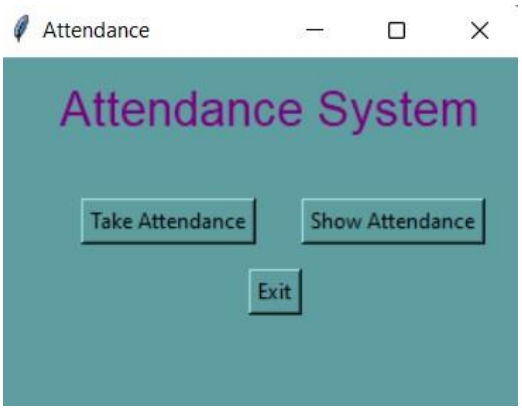

Fig 7.1 User interface

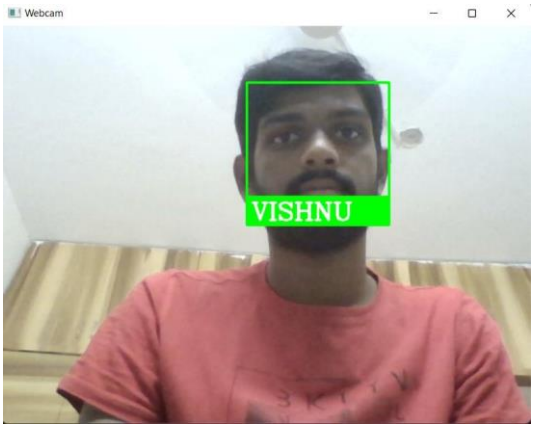

Fig 7.2 Working demo of facial recognition

Roll Number	Name	Time	Period	Present
1	Vishnu	20:27:17	NULL	ABSENT
1	Elon	19:45:29	7	ABSENT
1	Jack ma	09:57:05	1	ABSENT
1	Srikanth	11:52:06	3	ABSENT
2	Anish	11:51:40	3	ABSENT
1	Bill gates	20:27:04	NULL	ABSENT

Fig 7.3 Database Information

VIII CONCLUSION AND FUTURE WORK

The proposed attendance system using image recognition is inherently much more efficient than it's fingerprint or RFID based counterparts due to no requirement of manual intervention. It also provides various benefits:

- It saves ample time as the system computes the attendance for several people concurrently.
- The program is proportionally lightweight and can be run on affordable hardware.
- Requires very minimal hardware.

This system can be a stand alone system or be a specialized component which is part of a much larger system. The input device could be integrated to a Arduino or a Raspberry Pi system. It could also be part of an IoT system or any other device like a mobile phone or a webcam.

Future enhancements could be:

- Recognition of people wearing glasses, masks, tattoos etc
- Very High Recognition Accuracy
- Distinguish between physical and digital faces