Project Report

*on*

Attendance System with  
Image Processing using KNN

In Subject: Design and Analysis of Algorithm

*by*

Aadesh Bajare (22210029)

Priyanshi Pawar (22210066)

Aayushi Ingle (22210258)

A picture containing icon

Description automatically generated

Department of Artificial Intelligence and Data Science

VIIT

**2024-2025**

**Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | TITLE | | Page No. |
| Chapter-1 | Introduction | |  |
| 1.1 | Project Overview | 3 |
| 1.2 | Key Objectives | 3 |
| 1.3 | Significance of the Project | 3 |
| Chapter-2 | Methodology | |  |
| 2.1 | Data Collection & Preprocessing | 4 |
| 2.2 | Face Recognition with KNN | 4 |
| 2.3 | Attendance Recording Process | 4 |
| 2.4 | CSV File Management & Security | 4 |
| 2.5 | User Interface & Usability | 4 |
| 2.6 | Hardware & Software Requirements | 5 |
| Chapter-3 | Implementation Details | |  |
| 3.1 | Code | 6 |
| 3.2 | Output | 8 |
| Chapter-4 | Conclusion | | 9 |
| Chapter-5 | References | | 9 |

**INTRODUCTION:**

**Project Overview:**

This project presents a face-recognition-based attendance system that utilizes image processing with the K-Nearest Neighbors (KNN) algorithm. The system consists of two main components: a registration module to add student faces to the database and an attendance verification module to mark attendance automatically in a CSV file. By capturing and recognizing facial features, the system eliminates manual attendance processes, ensuring a more accurate and efficient approach to attendance tracking. This solution not only enhances reliability but also offers a real-time approach to identifying students through facial recognition, making it a valuable addition to educational institutions.

**Key Objectives:**

The primary objective of this project is to design and implement an automated attendance system using facial recognition to reduce manual errors and save time. Key objectives include:

* Building a reliable facial recognition model using the KNN algorithm to ensure accurate student identification.
* Developing a user-friendly system for easy face data entry and real-time attendance tracking.
* Integrating a CSV file format for attendance records to maintain a reliable log.
* Ensuring secure storage of facial data to protect user privacy and comply with data security standards. This project also aims to promote contactless and efficient attendance management, suitable for modern educational settings.

**Significance of the Project:**

Traditional attendance methods, often involving roll calls or manual signatures, are time-consuming, prone to human error, and difficult to verify. This facial recognition-based attendance system addresses these issues by providing an automated, contactless solution that records attendance accurately in real-time. Using image processing and KNN, the system identifies students based on their unique facial features, reducing the possibility of proxy attendance and ensuring only authenticated students are marked present. The solution is scalable for use in schools, colleges, and workplaces where attendance monitoring is essential. This project demonstrates the practical application of artificial intelligence in simplifying administrative tasks, reducing paperwork, and fostering a more secure and efficient environment. Additionally, it highlights the potential for AI-driven automation in day-to-day institutional management, enhancing the overall efficiency of record-keeping processes.

**METHODOLOGY:**

1. **Data Collection and Preprocessing:**

* Utilizes OpenCV for video capture and facial detection through Haar Cascades, allowing robust face detection in real-time.
* The initial script captures 100 face images of each student, resizes them to a standard 50x50 pixel format, and stores them for consistency.
* Captured face data is saved using pickle for easy access and storage, with each student's images labeled for KNN model training.

1. **Face Recognition with KNN:**

* The K-Nearest Neighbors (KNN) algorithm is employed as the model for facial recognition due to its simplicity and effectiveness for small datasets.
* The second script loads pre-saved facial data and labels from the pickle files, which the KNN model then uses to classify incoming faces.
* This script continuously monitors live video feed for faces, identifying and labeling each detected face against the dataset using KNN’s distance-based classification.

1. **Attendance Recording Process:**

* When a student’s face is recognized, the system cross-references the identity and marks attendance in a CSV file for that day.
* The CSV file includes columns for ‘Name’ and ‘Time’ of attendance, and a timestamp is generated using datetime for each entry.
* To prevent multiple entries, each entry is appended only once after recognition.
* An auditory feedback mechanism is included, utilizing Windows Speech API to announce successful attendance registration.

1. **CSV File Management and Security:**
   * + Attendance is stored in daily CSV files, named by date to maintain organized records and enable quick retrieval.
     + The project uses structured file management by storing each student’s name along with a timestamp of attendance, ensuring a secure and reliable log.
     + Data security measures are minimal but sufficient for small-scale applications, ensuring user privacy by limiting file access to CSV storage only.
2. **User Interface and Usability:**

* The program employs basic OpenCV GUI functionality for real-time display, where detected faces and attendance labels are visually represented.
* Attendance marking is user-friendly, requiring minimal interaction with just a single key press to register attendance.
* With plans for scalability, the interface could support additional features like multi-student batch processing or remote access through networking.

1. **Hardware and Software Requirements:**

* **Hardware**: Any standard camera for facial capture, ideally with a resolution of at least 720p.
* **Software**: Python (version 3.x), OpenCV for image processing, Scikit-Learn for KNN algorithm, pickle for data serialization, and Windows Speech API for audio feedback.

**IMPLEMENTATION DETAILS:**

**Code:**

**Code File 1** [Add Students Data to Database]:

import cv2

import pickle

import numpy as np

import os

video=cv2.VideoCapture(0)

facedetect=cv2.CascadeClassifier('data/haarcascade\_frontalface\_default.xml')

faces\_data=[]

i=0

name=input("Enter Your Name: ")

while True:

    ret, frame = video.read()

    gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    faces=facedetect.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:

        crop\_img=frame[y:y+h, x:x+w, :]

        resized\_img=cv2.resize(crop\_img, (50,50))

        if len(faces\_data)<=100 and i%10==0:

            faces\_data.append(resized\_img)

        i=i+1

        cv2.putText(frame, str(len(faces\_data)), (50,50), cv2.FONT\_HERSHEY\_COMPLEX, 1, (50,50,255), 1)

        cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)

    cv2.imshow("Frame", frame)

    k=cv2.waitKey(1)

    if k==ord('a') or len(faces\_data)==50:

        break

video.release()

cv2.destroyAllWindows()

faces\_data=np.asarray(faces\_data)

faces\_data=faces\_data.reshape(100, -1)

if 'names.pkl' not in os.listdir('data/'):

    names=[name]\*100

    with open('data/names.pkl', 'wb') as f:

        pickle.dump(names, f)

else:

    with open('data/names.pkl', 'rb') as f:

        names=pickle.load(f)

    names=names+[name]\*100

    with open('data/names.pkl', 'wb') as f:

        pickle.dump(names, f)

if 'faces\_data.pkl' not in os.listdir('data/'):

    with open('data/faces\_data.pkl', 'wb') as f:

        pickle.dump(faces\_data, f)

else:

    with open('data/faces\_data.pkl', 'rb') as f:

        faces=pickle.load(f)

    faces=np.append(faces, faces\_data, axis=0)

    with open('data/faces\_data.pkl', 'wb') as f:

        pickle.dump(names, f)

**Code File 2** [Main File with KNN Implementation]:

from sklearn.neighbors import KNeighborsClassifier

import cv2

import pickle

import os

import csv

import time

from datetime import datetime

from win32com.client import Dispatch

def speak(str1):

    speak=Dispatch(("SAPI.SpVoice"))

    speak.Speak(str1)

video=cv2.VideoCapture(0)

facedetect=cv2.CascadeClassifier('data/haarcascade\_frontalface\_default.xml')

with open('data/names.pkl', 'rb') as f:

    LABELS=pickle.load(f)

with open('data/faces\_data.pkl', 'rb') as f:

    FACES=pickle.load(f)

knn=KNeighborsClassifier(n\_neighbors=5)

knn.fit(FACES, LABELS)

COL\_NAMES = ['NAME', 'TIME']

while True:

    ret, frame = video.read()

    gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    faces=facedetect.detectMultiScale(gray, 1.3, 5)

    for (x,y,w,h) in faces:

        crop\_img=frame[y:y+h, x:x+w, :]

        resized\_img=cv2.resize(crop\_img, (27,25)).flatten().reshape(1, -1)

        output=knn.predict(resized\_img)

        ts=time.time()

        date=datetime.fromtimestamp(ts).strftime("%d-%m-%Y")

        timestamp=datetime.fromtimestamp(ts).strftime("%H-%M-%S")

        exist=os.path.isfile("Attendance/Attendance\_" + date + ".csv")

        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 1)

        cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)

        cv2.rectangle(frame,(x,y-40),(x+w,y),(50,50,255),-1)

        cv2.putText(frame, str(output[0]), (x, y-15), cv2.FONT\_HERSHEY\_COMPLEX, 1, (255,255,255), 1)

        cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)

        attendance=[str(output[0]), str(timestamp)]

    cv2.imshow("Frame", frame)

    k=cv2.waitKey(1)

    if k==ord('a'):

        speak("Attendance Taken..")

        time.sleep(5)

        if exist:

            with open("Attendance/Attendance\_" + date + ".csv", "+a") as csvfile:

                writer=csv.writer(csvfile)

                writer.writerow(attendance)

            csvfile.close()

        else:

            with open("Attendance/Attendance\_" + date + ".csv", "+a") as csvfile:

                writer=csv.writer(csvfile)

                writer.writerow(COL\_NAMES)

                writer.writerow(attendance)

            csvfile.close()

    if k==ord('a'):

        break

video.release()

cv2.destroyAllWindows()

**Output:**

**CONCLUSION:**

The facial recognition attendance system offers an innovative solution for automated attendance marking, eliminating manual errors, and providing a streamlined, contactless process. Leveraging KNN for face recognition, it ensures reliable identification, making it a practical tool for educational and professional institutions. While developed as a prototype, this system demonstrates the potential of machine learning and image processing to simplify administrative tasks. Future developments could involve enhancing data security, improving model accuracy with larger datasets, and incorporating networking capabilities for multi-user functionality, further expanding its application scope and reliability.

**REFERENCES:**

1. Viola, P., & Jones, M. J. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features.
2. Zhang, Z. (2012). Microsoft Kinect Sensor and Its Effect. IEEE Multimedia.
3. Hu, W., & Tan, T. (2004). A survey on visual surveillance of object motion and behaviors.
4. OpenCV Documentation (2024).
5. Python Software Foundation. Python Language Reference, version 3.x. Available at: <https://www.python.org/>
6. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory.
7. Scikit-Learn Documentation (2024). Available from: <https://scikit-learn.org/>
8. Ranjan, R., Sankaranarayanan, S., & Chellappa, R. (2016). An all-in-one convolutional neural network for face analysis.
9. Daugman, J. G. (2003). Face and gesture recognition: Overview. IEEE Computer Society.
10. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation Learning: A Review and New Perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence.