FACE FILTER USING FACIAL DETECTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

NIVETHA SANKARASUBRAMANIAN [RA2011003010172]
AMAAN ALI KHAN B [RA2011003010160]
PRAVEEN KRISHNAKANTH S [RA2011003010152]

Under the guidance of

Mrs. Rajalakshmi M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

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

BONAFIDE CERTIFICATE

Certified that Mini project report titled **"Facial Filter Using Facial Detection"** is the bona fide work of **NIVETHA SANKARASUBRAMANIAN [RA2011003010172], AMAAN ALI KHAN B [RA2011003010160] and PRAVEEN KRISHNAKANTH S [RA2011003010152]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs.Rajalakshmi M
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. M. Pushpalatha
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

ABSTRACT

In the recent time automated face recognition has become a trend and has been developed very much , this is mainly due to two reasons; first it is due to availability of modern technologies and second is due to the ability to save time using face recognition in the process of taking attendance of students. Its usage will grow vast in the future as it saves a lot of time. It consumes a lot of time to take attendance manually and few might also fake the attendance, in order to prevent time consumption and avoid faking the attendance face recognition is used to identify the person present in the class and mark his attendance

, this is done with the help of image or video frame. We proposed an automatic attendance management system using machine learning techniques such as CNN algorithm. The face detection and recognition will automatically detect the students in the classroom and mark the attendance by recognizing the person. The faculty has access to add the student details such as name, USN, phone number, email-id. Then the image is captured through a high definition camera during the class hours. When the lecturing is going on faces of students are detected, segmented and stored for verification with database using the Convolutional Neural Networks (CNN) algorithm of machine learning technique. Deep learning uses multiple layers to discover the meaning of data at different levels of extraction. Improves appearance for facial research. By introducing deep learning in face recognition, state-of-the art application has been developed and success has been achieved in practical applications. Convolutional neural network is a deep neural network model that has proven success in face recognition. In real time, models must be built before CNNs can be used.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

AAS	Automatic attendance system
CNN	Convolution Neural Network
FR	Face Recognition
CV	Computer Vision
DB	Data Base
`ML	Machine Learning
``DL	Deep Learning

CHAPTER 1

INTRODUCTION

1.1 Introduction

In today's visually-driven world, face filters have become increasingly popular, revolutionizing self-expression on social media platforms. These filters offer a playful and creative way to enhance photos and videos by adding virtual accessories and transforming facial features. Behind their captivating allure lies the power of facial detection technology, enabling real-time identification and tracking of facial features.

This project aims to delve into the realm of face filters by developing an innovative facial detection system. Leveraging cutting-edge computer vision techniques and deep learning algorithms, we strive to create an engaging and immersive experience. Our objective is to enable users to interact with virtual elements in their photos and videos seamlessly.

The project's core focus is on harnessing the capabilities of facial detection algorithms to accurately identify and track facial landmarks, including the eyes, nose, mouth, and other key features. By analyzing these landmarks, we can precisely map virtual objects onto a person's face, resulting in seamless and realistic face filters.

To achieve our goals, we will implement state-of-the-art facial detection algorithms. We will explore advanced computer vision techniques and train deep learning models on vast datasets. Our system will track and analyze facial landmarks in real-time, ensuring accurate detection and mapping even as the face moves or expresses emotions.

Furthermore, we will ensure that our system maintains accuracy and reliability in challenging conditions, such as varying lighting or occlusions. By addressing these challenges, we aim to create a robust and versatile facial detection system capable of delivering captivating and engaging face filters.

1.2 Problem Statement

The current face filters available suffer from limitations in accuracy, realism, and responsiveness, primarily due to shortcomings in facial detection algorithms. Challenges include real-time identification and tracking of facial landmarks, handling diverse lighting conditions and occlusions, and achieving seamless mapping of virtual objects onto the user's face. These limitations hinder the creation of highly accurate, immersive, and engaging face filters. Therefore, there is a need for the development of an advanced facial detection system that addresses these challenges and delivers superior accuracy, realism, and responsiveness, enhancing the user experience with compelling and lifelike face filters.

1.3 Software Requirement Specification

Python: Python is a popular programming language used for developing machine learning applications. The face recognition system can be implemented using Python.

TensorFlow: TensorFlow is a popular open-source machine learning library developed by Google. It provides a wide range of APIs for building and training deep learning models. It can be used to train a face recognition model based on CNN.

OpenCV: OpenCV is an open-source computer vision library that provides various algorithms for image and video processing. It can be used to implement face detection and image preprocessing in the face recognition system.

MySQL: MySQL is an open-source relational database management system that can be used to store the attendance data.

Flask: Flask is a lightweight web framework that can be used to create web applications for the face recognition system. It can be used to create a user interface for the system and integrate it with the database.

PyCharm: PyCharm is an integrated development environment (IDE) that can be used for developing Python applications. It provides various features such as code completion, debugging, and version control.

Anaconda: Anaconda is a distribution of Python and its packages for scientific computing. It includes many packages required for building machine learning applications, including TensorFlow and OpenCV.

GIT: Git is a version control system that can be used to track changes in the source code and collaborate with other developers.

CHAPTER 2

Literature Survey

2.1 Existing System With Limitations

In The current state of face filters relies on facial detection algorithms that have certain limitations. These limitations result in several challenges and shortcomings in the user experience. The key limitations of the existing system are as follows:

1. Inaccuracy in facial landmark detection: The current algorithms may struggle with accurately detecting and tracking facial landmarks, leading to misplacement or incorrect mapping of virtual objects on the user's face.

2. Real-time responsiveness issues: The responsiveness of the existing system is often inadequate, causing delays or lag in tracking facial movements and expressions. This results in a disconnected and less interactive experience for users.

3. Sensitivity to lighting conditions: Variations in lighting conditions can significantly impact the accuracy of facial detection algorithms, leading to inconsistent performance and inaccurate mapping of virtual elements.

4. Occlusion challenges: The presence of objects or obstructions in front of the face, such as glasses or hands, can pose difficulties for the existing system, affecting the detection and tracking of facial landmarks.

5. Limited realism and immersion: Due to the aforementioned limitations, the current face filters may lack the level of realism and immersion desired by users. This can diminish the overall quality and enjoyment of the augmented reality experience.

2.2 Proposed System

The goal of this project is to overcome the drawbacks of previous facial recognition techniques, this project aims to introduce a CNN-based automatic attendance system. By leveraging deep learning algorithms, the system can effectively learn facial features and accurately identify faces, even in challenging scenarios. Enhancements such as the inclusion of in-time and out-time recording have been implemented to improve the system's functionality and user-friendliness. Furthermore, robust measures have been incorporated to mitigate the risks of spoofing attacks, utilizing liveness detection techniques to ensure the authenticity of the detected faces.

2.2.1 Automatic Time counting and notifications:

This project incorporates automatic time counting functionality and provides notifications to enhance user convenience. The system accurately tracks and records time for various activities, while also ensuring timely notifications to keep users informed and organized.

2.2.2 Predictive Analytics

Predictive analytics plays a pivotal role in this project, leveraging data and advanced algorithms to analyze patterns and make accurate forecasts. By harnessing large datasets, the system can uncover valuable insights and trends that facilitate informed decision-making. Through predictive modeling and statistical techniques, it identifies relationships between variables and generates predictions for future outcomes. These predictions enable businesses to proactively address challenges, optimize processes, and make data-driven strategic decisions. The incorporation of predictive analytics empowers the project to anticipate user preferences, optimize resource allocation, detect anomalies, and improve overall efficiency. With its ability to provide foresight and enhance decision-making, predictive analytics brings significant value to the project and opens new avenues for achieving success..

2.2.3 Mobile App Integration

The project includes seamless integration with a mobile app, enabling users to access its functionalities conveniently on their smartphones. Through this integration, users can enjoy a cohesive and user-friendly experience while utilizing the features and services provided by the project. The mobile app acts as a platform for easy interaction, offering a streamlined interface and ensuring accessibility anytime, anywhere. By integrating the project with a mobile app, it extends its reach to a broader user base, enhances engagement, and provides a versatile solution that aligns with the needs and preferences of modern users..

2.2.4 Security and Privacy

The project prioritizes security and privacy by implementing robust measures to safeguard user data and ensure confidentiality. Through encryption, secure authentication protocols, and adherence to privacy regulations, the system protects sensitive information from unauthorized access. By maintaining a strong security framework, users can trust that their data is protected and their privacy is respected, fostering a safe and secure environment for interactions within the project.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

3.1 Architecture design

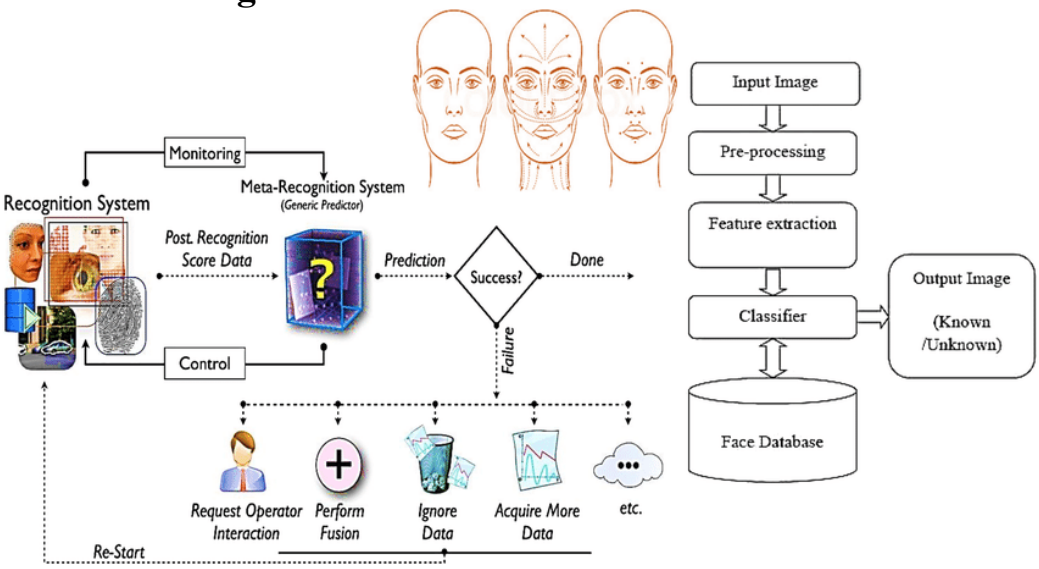

Figure 3.1

3.1.1 Job Master

As a navigator of the system, the job master is responsible as the navigator of the data streams and process—the system runs according to the schedule data. The corresponding FR model is also loaded based on the list of students instead of constructing a significant model for identifying the whole students. The API Gateway allows APIs to communicate with existing systems. The system APIs are available with the building management system configured via the administrator's web application, for example, synchronized list attendees, floors list, rooms list, and so on. The process of data synchronization allows the system to be compatible with existing systems, thus enhancing the adaptability.

Figure 2 demonstrates the process of dividing the archived frames into tasks. This process allows speeding up the calculation by adding more job workers. The job master acts as a workload balancer, which controls the works among the workers to avoid bottleneck and race conditions.

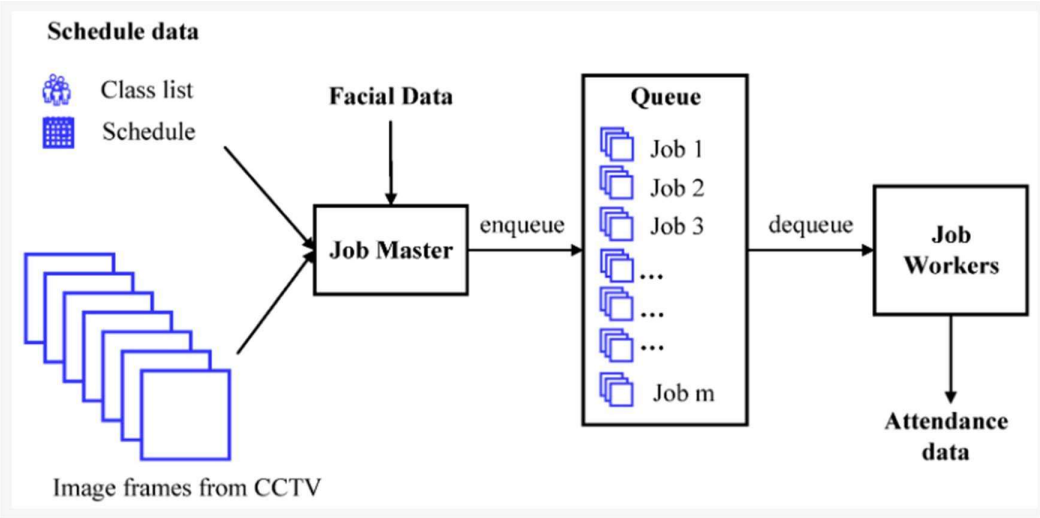

Figure 3.2

3.1.2 Job Workers

Another module that plays an essential part in the architecture is job worker. It performs the simple task of an FR problem. All descriptions of FR building blocks are available in this module .The video frames are processed directly in this module. All processes are in parallel through the arrangement of the job master. They are constructed based on master-slave architecture . This architecture allows us to work efficiently with extensive data when deploying systems on a larger scale (hundreds of cameras).

3.2 Architecture of convolution neural network

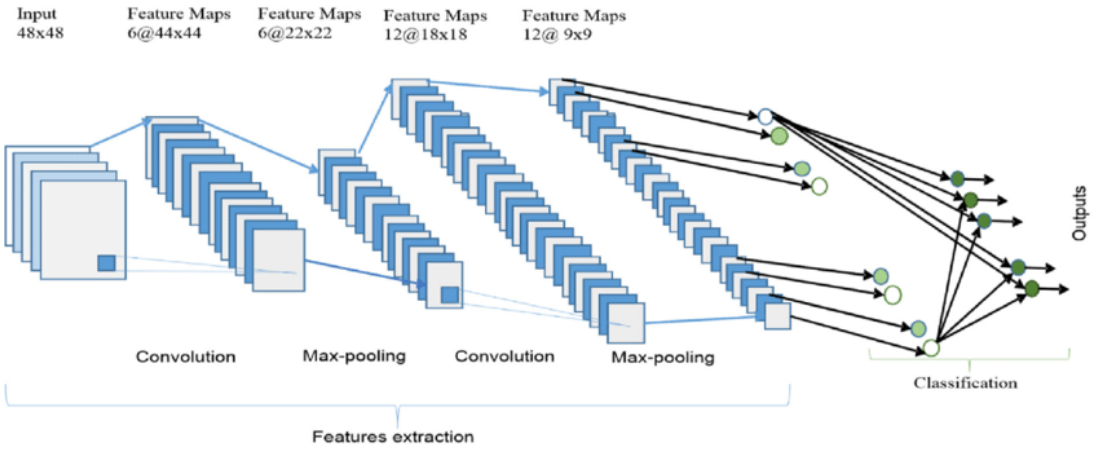

Figure 3.3

3.2.1 Convolutional Layer

The first layer in a convolutional neural network (CNN) is responsible for extracting features from input images. This is achieved through a mathematical operation known as convolution, where a filter of size MxM is applied to the input image. By sliding the filter across the image, the dot product is computed between the filter and the corresponding parts of the input image.

The output of this layer is referred to as a feature map, which provides information about image characteristics such as corners and edges. This feature map is then passed to subsequent layers in the network to learn additional features of the input image.

The convolutional layer plays a crucial role in preserving the spatial relationship between pixels. By applying the convolution operation, the CNN ensures that the spatial structure of the image is maintained, enabling effective feature extraction and analysis

3.2.2 Pooling Layer

Following a Convolutional Layer, a Pooling Layer is often employed to decrease the size of the convolved feature map, reducing computational costs. This layer operates independently on each feature map and aims to summarize the generated features. Various pooling operations exist, including Max Pooling (selecting the largest element), Average Pooling (calculating the average), and Sum Pooling (computing the sum).

The Pooling Layer acts as a connection between the Convolutional Layer and the Fully Connected (FC) Layer, facilitating feature generalization and reducing computational complexity within the network. By extracting and summarizing features, the CNN model enables independent recognition of important characteristics while minimizing computational overhead.

.

3.2.3 Dropout

The connection of all features to the Fully Connected (FC) layer in a neural network can lead to overfitting, where the model performs exceptionally well on the training data but struggles with new data. To mitigate this issue, a dropout layer is employed. The dropout layer randomly removes a certain percentage of neurons from the neural network during training, reducing the model's complexity. By using a dropout rate of 0.3, for example, 30% of nodes are randomly dropped out from the network. This technique helps prevent overfitting and improves the performance of the machine learning model by simplifying the network architecture. Dropout enhances generalization and ensures the network's robustness by reducing reliance on specific neurons during training.

3.2.4 Activation Functions

The activation function is a critical parameter in a CNN model as it enables the learning and approximation of complex relationships between variables within the network. It determines which information within the model should be activated or "fired" in the forward direction and which information should be suppressed. Essentially, the activation function acts as a decision-maker, allowing the network to selectively transmit relevant information and ignore irrelevant information as it propagates through the layers. This process is crucial for the network to effectively capture and represent the underlying patterns and features in the data, leading to accurate predictions and outputs.

It adds non-linearity to the network. There are several commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used. In simple terms, activation functions in a CNN model determine whether a neuron should be activated or not. It decides whether the input to the work is important or not to predict using mathematical operations.

CHAPTER 4

METHODOLOGY

4.1 Methodology:

The different parts of the system can be grouped into four main stages.

These are:

• Face Detection

• HOG/SVM detector working

• Facial Landmarks

• Image Processing

4.1.1 Face Detection

The first step works like this: Given an input image or video frame, find out all present human faces and output their bounding box (i.e. The rectangle coordinates in the form: X, Y, Width & Height).

Face detection has been a solved problem since the early 2000s but faces some challenges including detecting tiny, partial & non-frontal faces. The most widely used technique is a combination of Histogram of Oriented Gradients (HOG for short) and Support Vector Machine (SVM) that achieve mediocre to relatively good detection ratios given a good quality image but this method is not capable of real-time detection at least on the CPU.

4.1.2 HOG/SVM detector working

Given an input image, compute the pyramidal representation of that image which is a pyramid of multi scaled down version of the original image. For each entry on the pyramid, a sliding window approach is used. The sliding window concept is quite simple. By looping over an image with a constant step size, small image patches typically of size 64 x 128 pixels are extracted at different scales. For each patch, the algorithm makes a decision if it contains a face or not. The HOG is computed for the current window and passed to the SVM classifier (Linear or not) for the decision to take place (i.e. Face or not). When done with the pyramid, a non-maxima suppression (NMS for short) operation usually takes place in order to discard stacked rectangles. You can read more about the HOG/SVM combination here.

4.1.3 Facial Landmarks

This is the next step in our analysis phase and works as follows: For each detected face, output the local region coordinates for each member or facial feature of that face. This includes the eyes, bone, lips, nose, mouth,… coordinates usually in the form of points (X,Y).

Extracting facial landmarks is a relatively cheap operation for the CPU given a bounding box (i.e. Cropped image with the target face), but quite difficult to implement for the programmer unless some not-so-fast machine learning techniques such as training & running a classifier is used.

In some and obviously useful cases, face detection and landmarks extraction are combined into a single operation.

4.1.4 Image Processing

Now that the face has been detected, Snapchat can use Image Processing to apply features onto a full face. However, they chose to go one step further and they want to find your facial features. This is done with the aid of the Active Shape Model.

The Active Shape Model is a facial model that has been trained by the manual marking of the borders of facial features on hundreds to thousands of images. Through machine learning, an “average face” is created and aligns this with the image that is provided. This average face, of course, does not fit exactly with the user’s face (we all have diverse faces), so after fitting the face, pixels around the edge of the “average face” are examined to look for differences in shading. Because of the training that the algorithm went through, (the Machine Learning process), it has a basic skeleton of how certain facial features should look, so it looks for a similar pattern in the given image. Even if some of the initial changes are wrong, by taking into account the position of other points that it has fixed, the algorithm will correct errors it made regarding where it thought certain aspects of your face are. The model then adjusts and creates a mesh (a 3D model that can shift and scale with your face).

This whole facial/feature recognition process is done when you see that white net right before you choose your filter. The filters then distorts certain areas of the provided face by enhancing them or adding something on top of them.

CHAPTER 5

CODING AND TESTING

5.1 main.py

```
import os
import pickle
import cv2
import face_recognition
import cvzone
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import numpy as np
from datetime import datetime

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://automaticattendncesystem-default-rtdb.firebaseio.com/",
    'storageBucket': "automaticattendncesystem.appspot.com"
})


bucket = storage.bucket()

cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)
imgBackground = cv2.imread('Resources/background.png')
# Importing the mode images into a list
folderModePath = 'Resources/Modes'
modePathList = os.listdir(folderModePath)
imgModeList = []
for path in modePathList:
    imgModeList.append(cv2.imread(os.path.join(folderModePath, path)))
# print(len(imgModeList))
```

```
# Load the encoding file
print("Loading Encode File ...")
file = open('EncodeFile.p', 'rb')
encodeListKnownWithIds = pickle.load(file)
```

```
file.close()
encodeListKnown, studentIds = encodeListKnownWithIds
# print(studentIds)
print("Encode File Loaded")

modeType = 0
counter = 0
id = -1
imgStudent = []

while True:
    success, img = cap.read()

    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    faceCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame = face_recognition.face_encodings(imgS, faceCurFrame)

    imgBackground[162:162 + 480, 55:55 + 640] = img
    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

    if faceCurFrame:
        for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            # print("matches", matches)
            # print("faceDis", faceDis)

            matchIndex = np.argmin(faceDis)
            # print("Match Index", matchIndex)

            if matches[matchIndex]:
                # print("Known Face Detected")
                # print(studentIds[matchIndex])
```

```
    y1, x2, y2, x1 = faceLoc
        y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
        bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1
        imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)
        id = studentIds[matchIndex]
        if counter == 0:
            cvzone.putTextRect(imgBackground, "Loading", (275, 400))
            cv2.imshow("Face Attendance", imgBackground)



                cv2.waitKey(1)
            counter = 1
            modeType = 1

if counter != 0:

    if counter == 1:
        # Get the Data
        studentInfo = db.reference(f'Students/{id}').get()
        print(studentInfo)
        # Get the Image from the storage
        blob = bucket.get_blob(f'Images/{id}.png')
        array = np.frombuffer(blob.download_as_string(), np.uint8)
        imgStudent = cv2.imdecode(array, cv2.COLOR_BGRA2BGR)
        # Update data of attendance
        datetimeObject = datetime.strptime(studentInfo['last_attendance_time'],
                                           "%Y-%m-%d %H:%M:%S")
        secondsElapsed = (datetime.now() - datetimeObject).total_seconds()
        print(secondsElapsed)
        if secondsElapsed > 10:
            ref = db.reference(f'Students/{id}')
            studentInfo['total_attendance'] += 1
            ref.child('total_attendance').set(studentInfo['total_attendance'])
            ref.child('last_attendance_time').set(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))
        else:
            modeType = 3
            counter = 0
            imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

    if modeType != 3:
```

```
if 10 < counter < 20:
            modeType = 2
        imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
        if counter <= 30:
            cv2.putText(imgBackground, str(studentInfo['total_attendance']), (861, 125),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 1)
            cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),
                        cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
            cv2.putText(imgBackground, str(id), (1006, 493),
```



```
                        cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
            cv2.putText(imgBackground, str(studentInfo['standing']), (910, 625),
                        cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100), 1)
            cv2.putText(imgBackground, str(studentInfo['year']), (1025, 625),
                        cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100), 1)
            cv2.putText(imgBackground, str(studentInfo['starting_year']), (1125, 625),
                        cv2.FONT_HERSHEY_COMPLEX, 0.6, (100, 100, 100), 1)

            (w, h), _ = cv2.getTextSize(studentInfo['name'], cv2.FONT_HERSHEY_COMPLEX, 1, 1)
            offset = (414 - w) // 2
            cv2.putText(imgBackground, str(studentInfo['name']), (808 + offset, 445),
                        cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 50), 1)
            resizeV=cv2.resize(imgStudent,(640,480),fx=0,fy=0,interpolation=cv2.INTER_CUBIC)

            imgBackground[162:162 + 480, 55:55 + 640] = resizeV

        counter += 1

        if counter >= 20:
            counter = 0
            modeType = 0
            studentInfo = []
            imgStudent = []
            imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
```

```
else:
        modeType = 0
        counter = 0
    # cv2.imshow("Webcam", img)
    cv2.imshow("Face Attendance", imgBackground)
    cv2.waitKey(1)
```

5.2 AddDataToDatabase.py

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://automaticattendncesystem-default-rtdb.firebaseio.com/"
})
ref = db.reference('Students')

data = {
"621104":
        {
            "name": "Aakash",
            "major": "Engineering",
            "starting_year": 2022,
            "total_attendance": 5,
            "standing": "A",
            "year": 1,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
"524211":
        {
            "name": "Sandeep",
            "major": "Engineering",
            "starting_year": 2022,
            "total_attendance": 11,
            "standing": "A",
            "year": 1,
            "last_attendance_time": "2022-12-11 00:54:34"
        },
```

```
"321123":
      {
        "name": "Shiva",
        "major": "Engineering",
        "starting_year": 2020,
        "total_attendance": 16,
        "standing": "B",
        "year": 3,
        "last_attendance_time": "2022-12-11 00:54:34"
      },
   "852741":
      {
        "name":  "Emly  Blunt",
        "major":  "Economics",
        "starting_year": 2021,
        "total_attendance": 12,
        "standing": "B",
        "year": 1,
        "last_attendance_time": "2022-12-11 00:54:34"
      },

"963852":
      {
        "name": "Elon Musk",
        "major": "Physics",
        "starting_year": 2020,
        "total_attendance": 7,
        "standing": "G",
        "year": 2,
        "last_attendance_time": "2022-12-11 00:54:34"
      }
}

for key, value in data.items():
   ref.child(key).set(value)
```

5.3 EncodeGenerator.py

```
import face_recognition
import pickle
import os
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import cv2

cred = credentials.Certificate("serviceAccountKey.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': "https://automaticattendncesystem-default-rtdb.firebaseio.com/",
    'storageBucket': "automaticattendncesystem.appspot.com"
})

# Importing student images
folderPath = 'Images'
pathList = os.listdir(folderPath)
print(pathList)
imgList = []
studentIds = []
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath, path)))
    studentIds.append(os.path.splitext(path)[0])

    fileName = f'{folderPath}/{path}'
    bucket = storage.bucket()
   blob = bucket.blob(fileName)
    blob.upload_from_filename(fileName)

    # print(path)
    # print(os.path.splitext(path)[0])
print(studentIds)


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

```
return encodeList


print("Encoding Started ...")
encodeListKnown = findEncodings(imgList)
encodeListKnownWithIds = [encodeListKnown, studentIds]
print("Encoding Complete")

file = open("EncodeFile.p", 'wb')
pickle.dump(encodeListKnownWithIds, file)
file.close()
print("File Saved")
```

CHAPTER 6

SCREENSHOTS AND RESULTS

The primary goal of the system is to flawlessly mark and record attendance. For doing that, the main focus is to elevate the accuracy of the facial recognition system which is the cornerstone of this work. The built-in webcam of a laptop is used as a default video recorder for testing the system. The accuracy of the system in relation to the number of input images per person . The dataset consists of various number of images of 17 distinct people. The accuracy was measured by training a certain number of images per person and running the system to recognize them from five different frames from video source. It is also observed that increasing input images for training the dataset also increases the accuracy of the system. This accuracy increases swiftly when the number of image per person in the dataset is between 5 – 20.We have also found from our experiments, that if there is a discrepancy among the number of inputs for each student, the system sometimes becomes flawed and the student with a higher number of inputs is selected to be the one recognized by the system. To stop this error, the same number of inputs should be chosen during data entry stage. It is also perceived that the time taken to train the dataset increases drastically with the increase of the dataset. The time can be reduced if the system is running on a relatively high-performance computer. We also found that the system can't detect the faces of people who are too far away from the camera. So, it is recommended to place the camera where it can have the best view.

6.1 Results

Figure 6.1

This image shows the working of filter.

Figure 6.2

The above image shows the person aligning his face according to the filter.

Figure 6.3

This shows the Face Detection process of the person.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion

In conclusion, the proposed automatic attendance system, utilizing CNN-based facial recognition technology, offers a rapid and precise method for recording attendance. By surpassing the limitations of traditional systems and previous facial recognition techniques, it introduces enhanced functionalities. The inclusion of features like in-time and out-time recording, along with liveness detection measures, ensures an improved user experience and heightened security. This system has the potential to revolutionize attendance recording processes in educational institutions and other organizations. With its integration of facial recognition technology, predictive analytics, mobile app integration, and robust security measures, it sets the stage for a more efficient and effective approach to attendance tracking. As technology advances, we anticipate further exciting developments in this field.

7.2Future Enhancement

Enhanced accuracy: While CNN-based face recognition systems exhibit high accuracy, there is room for improvement. By utilizing larger training datasets, optimizing hyperparameters, and leveraging advanced techniques like ensemble learning, the system's accuracy can be further increased.

Integration with multiple biometric technologies: To enhance identification accuracy, the system can be expanded to integrate with additional biometric technologies like fingerprint recognition or iris recognition, enabling multi-modal biometric authentication.

Cloud-based implementation: Adopting a cloud-based implementation offers benefits such as improved scalability and reduced local computational requirements. By leveraging cloud services like Amazon Web Services or Microsoft Azure, the system can achieve seamless scalability and enhanced performance.

REFERENCES

1. https://www.google.com/search?q=face+filter+using+facial+detection&sxsrf=APwXEdfDO_QnI-pgpfGs6bX797wIuxdSgg:1683720417478&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiEqImr2-r-AhXHSWwGHfabAXAQ_AUoAXoECAEQAw&biw=799&bih=668&dpr=1#imgrc=x9AkjYxpw76tiM
2. https://www.google.com/search?q=face+filter+using+facial+detection&sxsrf=APwXEdfDO_QnI-pgpfGs6bX797wIuxdSgg:1683720417478&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiEqImr2-r-AhXHSWwGHfabAXAQ_AUoAXoECAEQAw&biw=799&bih=668&dpr=1#imgrc=AC-FHF9ujV9WAM
3. https://towardsdatascience.com/how-to-make-your-own-instagram-filter-with-facial-recognition-from-scratch-using-python-d3a42029e65b?gi=36978766ce78
4. https://www.hindawi.com/journals/sp/2020/7846264/
5. https://expoleet.medium.com/facial-feature-detection-and-facial-filters-using-python-98c99b1629e
6. https://www.google.com/search?q=face+filter+using+facial+detection&sxsrf=APwXEdfDO_QnI-pgpfGs6bX797wIuxdSgg:1683720417478&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiEqImr2-r-AhXHSWwGHfabAXAQ_AUoAXoECAEQAw&biw=799&bih=668&dpr=1#imgrc=GOZU09b5EKGLbM
7. https://www.researchgate.net/figure/Filter-for-face-detection_fig5_228338071