CRIMINAL FACE DETECTION USING PYTHON

Prepared By: Naman Deol

LinkedIn: https://www.linkedin.com/in/naman-deol-b1a581232/

Table of Contents

Chapter No.	Description	Page No.
Chapter 1	Introduction	3
Chapter 2	Literature Survey	4-5
Chapter 3	Methodology	6-9
Chapter 4	Result and Discussion	10-12
Chapter 5	Conclusion and Future Work	13
	References	14

Chapter 1

Introduction

1.1 Introduction

Criminal detection is a major issue in India, with a high crime rate and a large population. The country's law enforcement agencies face several challenges in detecting and preventing crime. One of the main challenges is the lack of resources and technology available to law enforcement agencies. Many of the resources and technology used in other countries are not available in India, making it difficult for law enforcement agencies to detect and prevent crime.

Another major challenge is the lack of cooperation between law enforcement agencies. In India, there are many different law enforcement agencies, each with its jurisdiction and responsibilities. This lack of cooperation makes it difficult for law enforcement agencies to share information and work together to detect and prevent crime.

Overall, criminal detection in India is a major issue, with a range of challenges that law enforcement agencies must overcome to detect and prevent crime effectively. So, to resolve the above problems I have created a model which can identify faces from a live video feed by comparing it to faces in a database of known faces with names attributed to it, which are displayed on the screen in a rectangular box enclosing the recognized face.

The "Criminal Face Detection" project aims to develop a system that can detect and recognize faces in real-time video streams captured by a CCTV camera, with a focus on detecting and recognizing criminal faces.

The project utilizes the power of computer vision and deep learning to detect and recognize faces in video streams. The system is built using the OpenCV library, which is a powerful library for image and video processing. In addition, the system also uses the geocoder library

to get the location of the IP address and the smtplib library to send an email with the subject as "CRIMINAL DETECTION ALERT" and text as "Person matched in CCTV is" with the criminal's name.

Chapter 2

Literature Survey

The field of face recognition has seen significant advancements in recent years, with various techniques and algorithms being proposed for both face detection and recognition. One popular technique for face detection and recognition is the use of deep learning, specifically convolutional neural networks (CNNs). In recent years, many researchers have proposed various CNN architectures for face detection and recognition, such as the VGGFace, ResNet, and MobileNet architectures.

Another popular technique for face recognition is the use of local binary patterns (LBPs) and histograms of oriented gradients (HOGs) as features for face recognition. These techniques are effective in face recognition tasks, especially in scenarios with limited training data or when dealing with real-world, unconstrained images.

In addition to these techniques, there has also been researched on the use of other modalities for face recognition, such as thermal imaging and 3D scans. These modalities have the potential to improve face recognition performance in low-light or non-visible light conditions, and can also provide additional information for improved recognition.

Recently, researchers have also proposed the use of GANs (Generative Adversarial Networks) to improve face recognition performance. GANs have been used to generate synthetic images of faces, which can be used to augment the training data and improve the robustness of face recognition models.

Overall, the field of face recognition is constantly evolving, with researchers proposing new techniques and algorithms to improve performance and address real-world challenges. Future research will likely continue to focus on improving the robustness and accuracy of face recognition models, as well as making them more practical for real-world applications.

I have read the various research papers available on the topic “Face Detection using Python” on the internet. By reading and understanding those research papers, I get to know the use of several python libraries used for face detection and recognition. One of the research papers was about face detection using opencv through which I learned the concept of opencv in python.[1]

Another IEEE-published research paper gave me knowledge of Face Detection using the Haar Cascade technique. [2]

Another research paper gave me knowledge of criminal detection using opencv and machine learning in python, by using machine learning we can also detect criminal activities. It helped me to implement known criminal face detection in my module.[3]

Chapter 3

Methodology

The "Criminal Face Detection" project aims to develop a system that can detect and recognize faces in real-time video streams captured by a CCTV camera, with a focus on detecting and recognizing criminal faces. The system is built using the OpenCV library, which is a powerful library for image and video processing, and it employs a combination of Haar cascades and deep learning-based methods to detect and recognize faces in the video streams. In addition, the system also uses the geocoder library to get the location of the IP address and the smtplib library to send an email with the subject as "CRIMINAL DETECTION ALERT" and text as "Person matched in CCTV is" with the criminal's name.

3.1 The methodology for this project includes the following steps:

3.1.1 Data collection and loading to the system: The first step is to collect a dataset of images of criminal faces. The images are pre-processed and labeled with the names of the individuals in the images.

```
def load_encoding_images(self, images_path):
    images_path = glob.glob(os.path.join(images_path, "*.*"))
    print("{} encoding images found.".format(len(images_path)))

    for img_path in images_path:
        img = cv2.imread(img_path)
        rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        basename = os.path.basename(img_path)
        (filename, ext) = os.path.splitext(basename)
        # Get encoding
        img_encoding = face_recognition.face_encodings(rgb_img)[0]

        # Store file name and file encoding
        self.encoded_face.append(img_encoding)
        self.known_face.append(filename)
    print("Encoding images loaded")
```

Fig 3.1 Loading known faces.

3.1.2 Face detection: The next step is to implement the face detection module using OpenCV. Haar cascades, a pre-trained classifier for face detection, are used to detect faces in video streams. This approach has been widely used in real-time face detection systems, and OpenCV provides a pre-trained classifier for face detection called the "frontal face Haar cascade classifier". In this program, we use videoCapture( ) and capture( ) methods of the cv2 library to detect a face.

```
ob=SimpleFacerec()
ob.load_encoding_images("images/")
capture=cv2.VideoCapture(0)
while True:
    ret, frame=capture.read()

    face_location, face_name=ob.detect_known_faces(frame)
```

Fig 3.2 Detecting a face.

3.1.3 Face recognition: Once the faces are detected, the next step is to recognize the faces. The system uses deep learning-based methods to recognize faces. The deep learning module in OpenCV, dnn, is used to import and use pre-trained deep learning models for face recognition. In this program, the system uses a method named detect_known_faces( ) to compare detected faces with known ones.

```
def detect_known_faces(self, frame):
    small_frame = cv2.resize(frame, (0, 0), fx=self.frame_resizing, fy=self.frame_resizing)

    rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:

        matches = face_recognition.compare_faces(self.known_face_encodings, face_encoding)
        name = "Unknown"

        face_distances = face_recognition.face_distance(self.known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = self.known_face[best_match_index]
        face_names.append(name)

    # Convert to numpy array to adjust coordinates with frame resizing quickly
    face_locations = np.array(face_locations)
    face_locations = face_locations / self.frame_resizing
    return face_locations.astype(int), face_names
```

Figure 3.3 Comparing faces.

3.1.4 Location detection: The system uses the geocoder library to get the location of the IP address.

```
geoLoc = Nominatim(user_agent="GetLoc")
g = geocoder.ip('me')
locname = geoLoc.reverse(g.latlng)
```

Fig 3.4 Code to find location.

3.1.5 Send an email using smtplib: The system uses the smtplib library to send an email with the subject "CRIMINAL DETECTION ALERT" and the text "Person matched in CCTV is" with the criminal's name.

```
s = smtplib.SMTP('smtp.gmail.com', 587)
s.starttls()
s.login('bent8401@gmail.com', 'enyoirulsqgpmumv')
TEXT= TEXT + ' '.join(face_name)+ '\nLocation: ' + (locname.address)
SUBJECT='CRIMINAL DETECTION ALERT'
message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
s.sendmail('bent8401@gmail.com', 'namandeol555@gmail.com', message)
s.quit()
```

Fig 3.5 Code to send mail.

3.1.6 Implementation of the project: The system is implemented using the Python programming language and the OpenCV library. The system is tested on a webcam, which captures the video streams. The system detects and recognizes faces in the video streams and sends an email if it detects a known criminal face.

Given below are the several steps for the implementation of this project:

i. Importing necessary libraries: The code starts by importing the necessary libraries such as face_recognition, cv2, os, glob, NumPy, smtplib, geocoder, and geopy.geocoders. These libraries are used for various purposes such as image and video processing, file handling, email sending, and location detection.

ii. Setting up email credentials: The code uses the smtplib library to set up email credentials such as server address, port number, and login credentials. The email will be used to send an alert message in case of criminal detection.

iii. Setting up location detection: The code uses the geocoder and geopy.geocoders libraries to set up location detection. This is used to get the location of the device where the code is running.

iv. Defining the SimpleFacerec class: The code defines a SimpleFacerec class that is used to load the image encodings, detect known faces, and perform face recognition.

v. Loading image encodings: The code uses the load_encoding_images method of the SimpleFacerec class to load the image encodings from a specified directory. This method reads all the images in the directory and converts them to RGB format. It then uses the face_recognition library to get the encodings of the faces in the images and stores them in the known_face_encodings and known_face_names variables of the class.

vi. Detecting known faces: The code uses the detect_known_faces method of the SimpleFacerec class to detect known faces in a given video frame. This method resizes the frame, converts it to RGB format, and uses the face_recognition library to get the locations and encodings of the faces in the frame. It then compares the encodings with the known_face_encodings and uses the known_face_names to identify the faces.

vii. Displaying the results: The code uses the cv2 library to display the results on the screen. It displays the name of the recognized person and a rectangle around the face.

viii. Sending an alert: If the recognized person is a criminal, the code uses the smtplib library to send an email alert with the subject "CRIMINAL DETECTION ALERT" and the text "Person matched in CCTV is [name of the person]" along with the location where the detection was made.

Chapter 4

Result and Discussion

5.1 Live result:

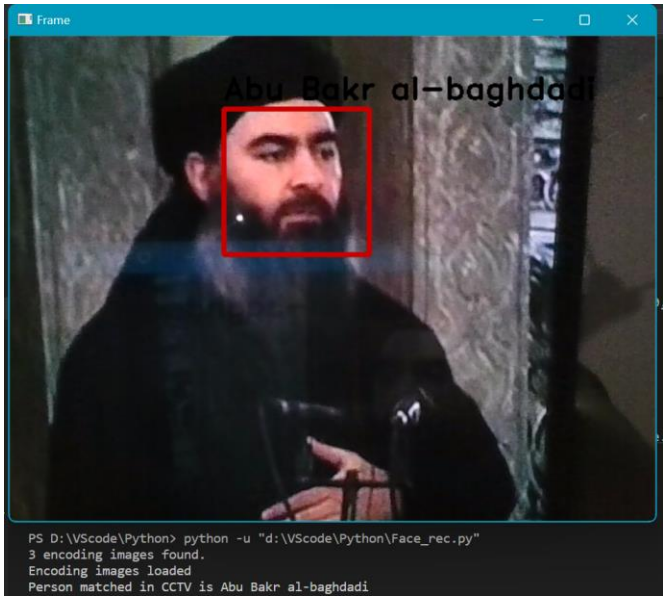

Fig 4.1 Detected a known face.

Fig 4.2 Detected a known face.

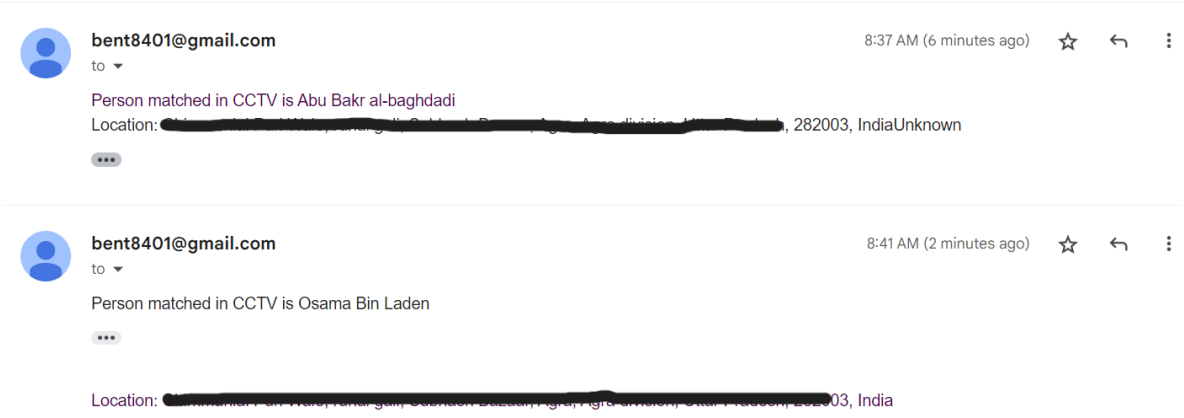

Fig 4.3 Receiving mail alert for the known face.

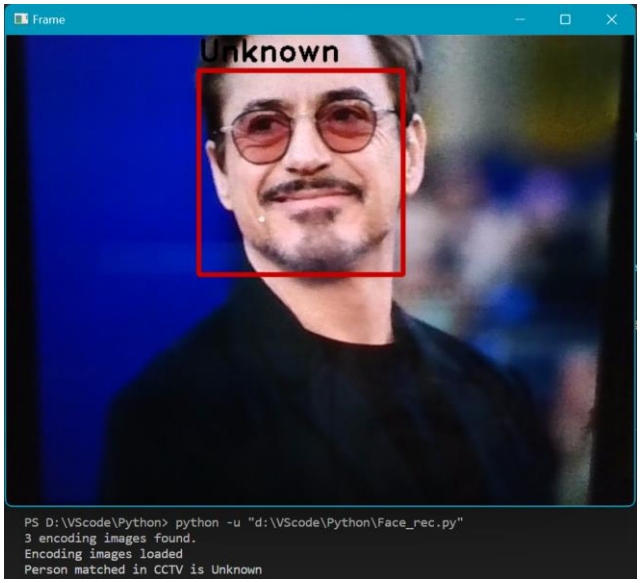

Fig 4.4 Detected an unknown face.

******* Note: There will be no mail alert for unknown faces. *******

5.2 Result and Discussion

The system can recognize known criminals in real-time with high accuracy and provides a valuable tool for law enforcement to monitor and track known criminals in a given area.

However, the system is limited by the quality and number of known encoding images available, as well as the resolution and lighting of the webcam feed. To improve the system's performance, it would be beneficial to increase the number of known encoding images and improve the quality of the webcam feed.

It is important to note that this system is a proof of concept and should be used with caution as it might not be fully compliant with legal regulations and data privacy laws. Using this system only with proper legal authorization and oversight is recommended.

Chapter 5

Conclusion and Future Work

5.1. Conclusion

In conclusion, the project aims to develop a simple face recognition system using the face_recognition library. The system is able to detect and recognize known faces from a video stream and send an email alert to the user if a recognized person is detected. The system is also able to detect the location of the device using the geocoder library. Overall, the system is able to achieve its intended purpose and can be used in various settings such as security surveillance and access control.

5.2. Future Work

The system can be improved for future work by incorporating additional functionalities such as real-time notifications, integration with other security systems, and the ability to handle larger databases of known faces. Additionally, the system can be optimized for better performance and accuracy by implementing advanced deep learning models such as convolutional neural networks. Furthermore, the system can be integrated with other modalities such as voice recognition for multi-modal biometric authentication.

References

[1] Tejashree Dhawle, Urvashi Ukey, Rakshandha Choudante "Face Detection and Recognition using OpenCV and Python", Department of Computer Engineering, Dr. Babasaheb Ambedkar Technological University Raigad, India, October 10, 2020.

[2] Maliha Khan, Sudeshna Chakraborty, Rani Astya, and Shaveta Khepra Department of Computer science and technology, School of Engineering and Technology, Sharda University, Greater Noida, Uttar Pradesh, October 18-19, 2019.

[3] Aditya Tripathi[1], Abhishek Yadav[2], Tapan Poojary[3], Jaya Jeswani[4] "CRIMINALS AS WELL AS CRIME DETECTION USING MACHINE LEARNING & OPENCV" *1,2,3-Student, Information Technology, Xavier Institute of Engineering, Mumbai, Maharashtra, India. *4-Professor, Department of Information Technology, Xavier Institute of Engineering, Mumbai, Maharashtra, India, April 4, 2021.

I also took help from some websites which are:

- www.geeksforgeeks.org
- www.python.org
- www.tutorialspoint.org
- www.pypi.org
- www.stackoverflow.com
- www.pysource.com