Recorded Video Face Recognition System Using OpenCV and Face-Recognition

Ayesha Ahmed Baig
20K-0172
BCS-6C
FAST-NUCES
Karachi Campus

Tahreem Fatima
20K-0483
BCS-6C
FAST-NUCES
Karachi Campus

Kanza Batool
20K-0423
BCS-6C
FAST-NUCES
Karachi Campus

***Abstract*—Face recognition is an important technology with various applications, including security, surveillance, and access control. This project presents a face recognition system that uses OpenCV and face-recognition library in Python. The system detects and recognizes faces in pre-recorded video files. The project uses Haar cascades to detect faces and extracts features using the face-recognition library. The features are then compared with the stored encodings of known faces to recognize individuals. The system displays the recognized person's name in real-time on the video stream. The project also provides a way to generate face encodings from images for use in the recognition system.**

I. Introduction

Face recognition has gained significant attention due to its increasing demand in security and surveillance systems. It is a complex process that involves detecting, extracting, and comparing facial features to identify individuals. In recent years, several techniques have been developed to automate face recognition, and OpenCV and face-recognition libraries in Python are widely used for this purpose.

This project presents a face recognition system that detects and recognizes faces in pre-recorded video files using OpenCV and the face-recognition library. The system extracts facial features and compares them with stored encodings of known faces to recognize individuals. The system displays the recognized person's name in real-time on the video stream.

II. Methodology

A. Data Collection

This step involves collecting a set of clear, well-lit images of known individuals and storing them in a folder named "faces". Each image should contain only one face per image. This dataset of known individuals is used later for comparison with the detected faces in the video frames.

B. Face Encoding

The next step is to extract facial features from the images using the face-recognition library. This is achieved by using the "face encodings" function which computes a 128-dimensional vector for each face. These vectors represent the unique features of each face, such as the distance between the eyes, the shape of the nose, etc. The resulting encodings are then saved in a " encodings " folder as CSV files. Each CSV file contains the face encodings for one known individual.

C. Face Recognition

In this step, face recognition is performed on a pre-recorded video using the OpenCV library. The video is read frame by frame, and each frame is converted to grayscale. The Haar cascade classifier is used to detect faces in the grayscale frame. This classifier is a pre-trained algorithm that identifies faces in an image based on their unique features, such as the contrast between the eyes and the skin. The detected faces are then cropped, resized, and passed to the face-recognition library to compute their encodings.

D. Comparison and Labeling

Once the face encodings have been computed for the detected faces in the video frames, they are compared with the encodings of known faces stored in the "encodings" folder. A match is found if the distance between the two encodings is less than a threshold value. If a match is found, the corresponding name of the known individual is retrieved from the "known names" list, and the name is displayed on the video frame along with a rectangle around the detected face. If no match is found, the label "Unknown" is displayed.

E. Display

Finally, the resulting video stream is displayed in real-time with the recognized names displayed on the frames. This step involves using OpenCV to display the video frames and overlaying the detected names and rectangles on top of them. The resulting video can be saved or streamed live.

III. Experimental Setup

A. Hardware

- Computer with a minimum of 4 GB RAM and a modern processor (i5 or higher).
- Webcam or video file for input.

B. Software

- Visual Studio Code (or any other code editor that supports Python).
- Python 3.6 or higher installed.
- Required Python packages (face-recognition, OpenCV, NumPy).

- A dataset of known faces in a ”faces” folder and their encodings in an ”encodings” folder.
- A pre-recorded video for testing the face recognition system.

C. Procedure

- Launch Visual Studio Code and open the face recognition project.
- Ensure that the required Python packages are installed.
- Collect a dataset of known individuals and store them in a ”faces” folder.
- Use the face-recognition library to extract facial features from the images and store them as encodings in a ”encodings” folder.
- Record a pre-recorded video or use a live webcam stream as input for the face recognition system.
- Launch the Python script and let it perform face recognition on the video stream.
- Evaluate the performance of the face recognition system by checking if the recognized names match the known names of the individuals in the video.
- Repeat the experiment with different videos or live streams to test the robustness of the system.

D. Metrics

1) Accuracy:
The percentage of correctly recognized faces out of the total number of faces in the video.

2) False positives:
The number of faces that were incorrectly recognized as known individual.

3) False negatives:
The number of faces that were not recognized as a known individual.

4) Processing time:
The time it takes for the system to recognize a face and display the result on the video stream.

IV. Results

In our experiments, we utilized various resources, including videos from Google, to evaluate the performance of our face recognition system. Our results show that the system was able to successfully detect and recognize faces that were completely visible in the video. However, when the face was at an angle of more than 45 degrees in any direction, the system was unable to detect it. This suggests that the performance of the system is limited to frontal views of the face and may need further improvements to handle more complex and varied scenarios. Nonetheless, the current system is a promising step towards developing a robust and accurate face recognition system.

V. Limitation

- The system requires a large dataset of known individuals. The accuracy of the system is directly proportional to the quality and size of the dataset. Therefore, if the system is used in a different environment or with a different set of individuals, the system may need to be retrained with a new dataset.
- The system is not effective when the face of an individual is partially or completely obscured. The system relies on the detection and recognition of facial features, and if those features are not visible, the system may not be able to recognize the individual.

VI. References

The following resources were used as guidance during the development of this project:

- Computer Vision - Object Detection in Python
- Face Recognition with Python 3.10 Tutorial(Webcam)