FACE RECOGNITION SYSTEM USING PYTHON

1. Introduction

Face Recognition is one of the most important applications of Computer Vision and Artificial Intelligence. It is widely used in security systems, attendance systems, surveillance cameras, and smart devices. The purpose of this project is to design and implement a real-time face recognition system using Python that can detect human faces from a live camera feed and recognize them by comparing them with a database of known faces.

The project combines traditional computer vision techniques with deep learning-based feature extraction to achieve accurate and efficient face recognition in real time.

2. Project Objectives

The main objectives of this project are:

- Understanding the fundamentals of face detection and face recognition.
- Building a database of known faces using images.
- Extracting facial features using pre-trained deep learning models.
- Recognizing faces in real time using a webcam.
- Displaying the recognized person's name on the video stream.
- Applying computer vision and artificial intelligence concepts in practice.

3. System Architecture Overview

The face recognition system is divided into two main phases:

1. Enrollment Phase (Building the face database)

2. Recognition Phase (Real-time face recognition)

Each phase follows a clear processing pipeline as explained below.

4. Enrollment Phase (Database Creation Workflow)

This phase is responsible for building the database of known faces before starting real-time recognition.

Images Folder

↓

Face Detection

↓

128-D Face Embedding

↓

Store (Database)

In this phase, the system loads images from a specified folder. For each image, face detection is performed to locate the face. Once detected, a 128-dimensional face embedding is extracted using a pre-trained Convolutional Neural Network (CNN). This embedding represents the unique features of the face. The extracted embeddings are then stored in memory along with the corresponding person names to form the face database.

5. Recognition Phase (Real-Time Workflow)

After building the database, the system starts real-time face recognition using a webcam.

Camera Frame

↓

Face Detection

↓

Face Embedding

↓

Compare Distances

↓

Recognized Name

The camera continuously captures video frames. Each frame is processed to detect faces. For every detected face, a face embedding is extracted and compared with all stored embeddings in the database. The system calculates the distance between embeddings and selects the closest match. If the distance is below a certain threshold, the face is recognized as a known person; otherwise, it is labeled as “Unknown”.

6. Detailed Processing Pipeline

The complete processing pipeline of the system can be summarized as follows:

Camera

↓

Read Frame

↓

Detect Faces

↓

Extract Embeddings

↓

Compare with Database

↓

Draw Name & Bounding Box

↓

Display Frame

Each step is executed sequentially to ensure accurate and real-time face recognition.

7. Face Detection Technique (HOG)

The system uses Histogram of Oriented Gradients (HOG) for face detection. HOG works by analyzing the direction and intensity of image gradients and edge structures. It is efficient and

suitable for real-time applications on CPU-based systems. Although it is less accurate than CNN-based detectors, it provides good performance with lower computational cost.

8. Face Encoding and Recognition

Face encoding is performed using a deep learning-based model trained on a large dataset of human faces. Each face is represented by a 128-dimensional numerical vector known as a face embedding. Face recognition is achieved by comparing these vectors using Euclidean distance. The face with the minimum distance is selected as the best match.

9. Tools and Technologies Used

- Programming Language: Python
- Image Processing: OpenCV
- Face Detection and Recognition: face_recognition library
- Numerical Computations: NumPy
- Detection Method: HOG
- Feature Extraction: CNN-based Face Embedding

10. Results and Discussion

The implemented system successfully detects and recognizes faces in real time under suitable lighting conditions. Known faces are accurately identified, and unknown faces are correctly labeled. The use of frame resizing improves system performance and ensures smooth video processing. Some images failed during encoding due to poor quality or unclear facial features, highlighting the importance of high-quality datasets.

11. Limitations

- Reduced accuracy in low-light conditions.
- HOG-based detection may fail in complex backgrounds.

- Dependence on the quality of images in the database.
- No face alignment or anti-spoofing mechanisms are implemented.

12. Future Improvements

- Use CNN-based face detection for higher accuracy.
- Add face alignment techniques.
- Store face embeddings in a persistent database.
- Implement attendance or access control systems.
- Improve recognition accuracy using multiple images per person.

13. Conclusion

This project demonstrates a complete real-time face recognition system using Python and computer vision techniques. By combining traditional face detection methods with deep learning-based feature extraction, the system achieves reliable and efficient performance. The project provides a strong foundation for understanding real-world applications of artificial intelligence and computer vision.

14. References

- OpenCV Documentation
- face_recognition Library Documentation
- dlib Library
- Python Official Documentation