Face Recognition System Using Python and ML

Name: Tanishak Jain

Roll No: 2102161520059

Name : Yogesh Murugan

Roll No: 2102161520063

Name : Shriram Kumar

Roll No: 2102161520056

Guide Name : Mr. Ajay Pratap

Abstract:

Face recognition is a smart technology that identifies people by looking at their faces. In this project, we built a system that uses tools like Python, OpenCV, and machine learning to
recognize people's faces in real time. The system can be used in places like schools, offices, and public areas to track attendance or improve security—without needing any physical contact.

1. Introduction:

Face recognition is becoming more common in daily life. It is used in mobile phones, offices, airports, and even in police work. Unlike fingerprints or eye scans, face recognition doesn't need to touch anything—just a camera is enough. The process works in four steps :

1. Find the face
2. Pick out important features (like eyes or nose)
3. Compare it with stored faces
4. Decide who it is

2. How It Works:

2.1 Face Detection Methods

There are two main ways to find and understand a face:

- Geometric method: Looks at where features like eyes, nose, and mouth are placed on the face.
- Photometric method: Looks at the face as a collection of pixels and tries to recognize it from patterns.

Some tools used include:

- Haar Cascades (Viola-Jones): Used to quickly find faces.
- HOG (Histogram of Oriented Gradients): Finds shapes and outlines in images.
- SIFT: Detects special points in images to tell them apart.

2.2 Using Python and Machine Learning

We use Python programming and a library called OpenCV to work with face images. The LBPH method helps the system recognize faces by learning patterns. It can then match new faces with ones it has seen before.

3. Methodology:

This section explains the step-by-step process:

1. Data Collection: We collect images of people we want the system to recognize and store them in a folder. Each image should clearly show the person's face.
2. Face Detection: The system looks at an image or video and finds where the face is. This is called detection. OpenCV or face recognition is used for this step.
3. Face Encoding: Each detected face is converted into a special code made of 128 numbers. This code represents the features of the face.
4. Face Matching: When a new face appears, the system compares its code with the saved ones. If it finds a match, it shows the person's name; otherwise, it shows "Unknown."
5. Output: The system displays the result, often by drawing a box around the face with the person's name.

4. Implementation :

Step-by-Step Implementation:

Step 1: Import Libraries

Step 2: Load Known Faces-Create a folder with known faces and load them using Python. Step 3: Capture Real-Time Video-Use OpenCV to access the webcam and detect faces.
Step 4: Compare Faces-Compare the detected face with known faces using face encodings.
Step 5: Display Results-Show the name of the recognized person on the video.

5. Main Parts of the System:

1. Face Detection – Finds faces using the webcam.
2. Preprocessing – Makes all face images the same size and color format.
3. Dataset Creation – Takes 100 pictures of ea ch person.
4. Training – Learns from these pictures to recognize people later.
5. Recognition – Matches live camera input to known faces.
6. Database Management – Saves face data and keeps it organized.
7. User Interface – Shows results and messages to users (like "Face Found" or "Unknown").

6. Tools and Technologies Used:

Programming Language: Python Editor: PyCharm Libraries: OpenCV,
NumPy,
Pandas Algorithm: LBPH (used to recognize faces) Hardware: Intel i5,
8GB RAM, 256GB SSD Software: Python 3.7+, OpenCV, Windows 11

7. How We Built It:

7.1Collecting Data

The system uses a webcam to take 100 photos of each person. These are saved in a folder and processed to make them ready for training.

7.2Training the System

The saved images are used to train the system. It creates a "faceprint"—a unique digital version of a face.

7.3Recognizing Faces

When the webcam sees a face again, it checks if it matches any of the trained faces. If it matches well, it says

"Known"; otherwise, it says "Unknown".

8. Where It Can Be Used:

- Attendance systems (schools and offices)
- Door access control
- Police investigations and surveillance
- Hospitals (to identify patients)
- Smart cities (e.g., traffic control and safety)

9. Challenges:

- Poor lighting or camera angle can cause problems
- Privacy concerns (storing people's face data safely)
- People might try to trick the system using photos or videos
- Needs to work fast and accurately in real-time situations

10. Conclusion:

Face recognition technology has improved a lot over the past 20 years. Today, it can be used for things like secure transactions, surveillance, and controlling access to buildings. These systems usually work well in controlled settings where conditions are ideal.

In the future, face recognition will be used more often in smart environments, where machines act like helpful assistants. For this to work smoothly, the technology must be able to recognize people naturally—just like humans do—without needing special actions or setups. It should also understand when it makes sense to recognize someone, just like a person would.

Although the technology is close to reaching this level, more research is still needed to make it work well in all kinds of situations, especially when using different types of input (like video or images from different angles).

Face recognition could also make things like taking student attendance faster and as accurate as other methods like fingerprint or card scanners. Today, there are many tools—like Microsoft's Face API and OpenCV—that make face detection and recognition easier and more reliable. While some tools have limits (like how often you can use them), combining different tools can help improve results and reduce those limitations

11. References:

1. https://github.com/ageitgey/face_recognition
2. https://opencv.org/
3. https://pypi.org/project/face-recognition/
4. https://arxiv.org/abs/1503.03832