

Face Recognition Using Haar Cascade

Preeti Kudambal¹, D. Suprithreddy², Revansiddapa .T³, Srinivas P⁴, Vishwa J⁵

¹Professor, Department Of Artificial Intelligence And Data Science, Faculty Of Engineering And Technology, Sharnbasva University, Kalaburagi, India.

^{2,3,4,5}Student, Department Of Artificial Intelligence And Data Science, Faculty Of Engineering And Technology, Sharnbasva University, Kalaburagi, India.

ABSTRACT

The Face Recognition System using Haar Cascade and LBPH Algorithm is an intelligent computer vision-based application designed to detect and recognize human faces through a webcam. In the present digital era, face recognition technology plays an important role in security, authentication, surveillance, attendance management, and identity verification. Traditional identification methods such as ID cards, passwords, and manual attendance registers are time-consuming and may lead to errors, proxy attendance, or unauthorized access. To overcome these limitations, the proposed system provides an automated and contactless method for identifying registered persons using facial features.

The system is developed using Python, OpenCV, Flask, Haar Cascade Classifier, and LBPH Face Recognizer. Haar Cascade is used to detect the human face from the live camera frame, while the LBPH algorithm is used to recognize the detected face by comparing it with trained face data. The system first collects face samples of registered users through the webcam and stores them in separate dataset folders. After collecting sufficient face images, the model is trained and saved for future recognition. During recognition, the webcam captures the live face, detects the face region, compares it with the trained model, and displays the recognized person's name along with the confidence value.

The project also includes a simple web-based interface developed using Flask, which allows users to collect face data, train the model, start face recognition, stop recognition, view registered persons, and check recognition logs. The system maintains a record of recognized persons with time and confidence details. The proposed solution reduces manual effort, improves identification accuracy, and provides a low-cost face recognition system suitable for academic institutions, offices, laboratories, and small organizations. Overall, this project demonstrates the practical use of computer vision and machine learning techniques for real-time face detection and recognition.

Keywords: Face Recognition, HAAR, Face Data

INTRODUCTION

In today's rapidly developing technological world, computer vision has become one of the most important fields in computer science. Computer vision allows machines to see, process, and understand visual information from images and videos. One of the most useful applications of computer vision is face recognition. Face recognition is a biometric technology used to identify or verify a person based on facial features. It is widely used in security systems, attendance management, mobile authentication, surveillance, access control, and identity verification.

In traditional identification systems, a person is verified using ID cards, passwords, PIN numbers, signatures, or manual records. These methods are simple but have many limitations. ID cards can be lost, passwords can be forgotten or shared, and manual attendance systems may lead to errors or proxy attendance. In many institutions and organizations, maintaining manual records consumes more time and effort. Therefore, there is a need for an automated system that can identify a person accurately and quickly without physical contact.

The proposed project titled "**Face Recognition Using Haar Cascade**" is developed to detect and recognize human faces using Python and OpenCV. The system uses a webcam to capture live video input. The captured frames are processed using image processing techniques. Haar Cascade Classifier is used to detect the face from the image or video frame. After detecting the face region, the system uses the LBPH Face Recognition Algorithm to recognize the registered person from the trained dataset.

Haar Cascade is a machine learning-based object detection method. It is commonly used for face detection

because it is fast, simple, and suitable for real-time applications. It detects faces by identifying important facial features such as eyes, nose, mouth, cheeks, and face structure. OpenCV provides pre-trained Haar Cascade XML files that can be directly used for detecting human faces. In this project, the Haar Cascade frontal face classifier is used to detect faces from webcam frames.

The LBPH algorithm, which stands for Local Binary Pattern Histogram, is used for recognizing the face after detection. LBPH works by analyzing the texture patterns of the face image and comparing them with the trained face data. It is suitable for small-scale

face recognition systems because it works effectively with limited datasets and does not require high-end hardware. In this project, the face images of registered persons are collected and stored in dataset folders. These images are then trained using LBPH, and the trained model is saved for recognition.

The proposed system is developed using Python as the main programming language because Python provides simple syntax and strong library support for computer vision applications. OpenCV is used for image processing, face detection, model training, and face recognition. Flask is used to develop the web-based interface of the system. The interface allows the user to perform different operations such as collecting face data, training the model, starting recognition, stopping recognition, viewing registered persons, and checking recognition logs.

The working of the system begins with face data collection. The user enters the name of the person, and the webcam captures multiple face images. These images are stored in a dataset folder with the person's name. After collecting the dataset, the training module processes the face images and creates a trained model file. During recognition, the system captures the live face from the webcam, detects the face using Haar Cascade, compares the detected face with the trained model, and displays the recognized person's name.

The system is useful because it reduces manual work and provides automatic face-based identification. It can be used as a basic model for attendance systems, security applications, and access control systems. Since the project uses a normal webcam and open-source tools, it is cost-effective and easy to implement. The system also helps students understand important concepts such as image processing, face detection, face recognition, dataset creation, model training, and real-time webcam-based applications.

In conclusion, the Face Recognition System using Haar Cascade and LBPH Algorithm provides a simple, efficient, and practical solution for real-time face detection and recognition. The project demonstrates how computer vision can be used to develop automated identification systems. It also creates a strong foundation for future improvements such as automatic attendance marking, database integration, login authentication, report generation, and advanced deep learning-based face recognition.

1.1 PROBLEM STATEMENT

In many institutions and organizations, identification and attendance systems are still maintained manually. Manual systems require more time and effort, and they may lead to errors, duplicate entries, and proxy attendance. In such systems, it becomes difficult to verify whether the correct person is present or not. ID card-based systems are also not completely secure because ID cards can be lost, exchanged, or misused by others.

Password-based authentication systems are commonly used, but they also have limitations. Users may forget their passwords, share them with others, or use weak passwords. Fingerprint-based systems provide better security, but they require physical contact with the scanner and may not work properly if the finger is wet, dusty, or injured. Therefore, a contactless and automated identification system is required.

Face recognition provides a better solution because the face is a natural and unique identity feature of a person. A webcam can capture the face without physical contact, and the system can automatically identify the person by comparing the face with stored face data. However, developing such a system requires proper face detection, dataset creation, model training, and real-time recognition.

The main problem addressed in this project is to design and develop a simple face recognition system that can detect human faces using Haar Cascade and recognize registered persons using the LBPH face recognition algorithm. The system should allow face data collection, model training, real-time recognition, and recognition log maintenance through a user-friendly web interface.

Therefore, the problem can be stated as:

“To design and develop a real-time face recognition system using Haar Cascade and LBPH algorithm that can detect faces from webcam input, train collected face data, recognize registered persons, and display

recognition results automatically.”

1.2 OBJECTIVES

The main objectives of the Face Recognition System using Haar Cascade are as follows:

- To develop a real-time face recognition system using Python and OpenCV.
- To detect human faces from live webcam video using Haar Cascade Classifier.
- To collect face images of registered persons and store them in separate dataset folders.
- To train the collected face images using the LBPH face recognition algorithm.
- To recognize registered persons by comparing live webcam input with the trained model.
- To display the recognized person's name along with confidence value on the output screen.
- To design a simple and user-friendly web interface using Flask.
- To maintain recognition logs for tracking recognized persons.
- To reduce manual effort in identity verification and attendance-related applications.
- To create a basic framework for future enhancement such as attendance marking, database integration, report generation, and advanced face recognition techniques.

CHAPTER 2

LITERATURE SURVEY

2.5.1 Face Detection Using Haar Cascade

Viola and Jones (2001) proposed a study titled “Rapid Object Detection using a Boosted Cascade of Simple Features.” [1] The authors introduced a fast object detection framework based on Haar-like features, integral images, AdaBoost learning, and cascade classifiers. The study demonstrated that human faces can be detected efficiently from images using a trained classifier. This method became one of the most widely used techniques for real-time face detection applications.

Viola and Jones (2004) further improved the real-time face detection framework by presenting a robust face detection approach. [2] The research focused on detecting frontal faces with high speed and accuracy. The study showed that a cascade classifier can reject non-face regions quickly and process only face-like regions in detail. This concept is highly useful in webcam-based face detection systems because it reduces processing time.

2.5.2 Face Recognition Using LBPH Algorithm

Ahonen, Hadid, and Pietikainen (2006) proposed a face recognition method based on Local Binary Patterns. [5] The study introduced the use of local texture patterns for representing facial images. In this method, the face image is divided into small regions, and local binary pattern histograms are extracted from each region. These histograms are then combined to form a feature representation of the face.

Ojala, Pietikainen, and Maenpaa (2002) presented Local Binary Pattern as a texture classification method. [6] The authors explained how local binary patterns can describe texture information by comparing each pixel with its neighboring pixels. This method became popular in computer vision because it is simple, efficient, and useful for texture-based image analysis.

2.5.3 OpenCV-Based Computer Vision Applications

Bradski (2000) introduced OpenCV as an open-source computer vision library. [9] OpenCV provides several tools and functions for image processing, video processing, object detection, and machine learning. It supports programming languages such as Python, C++, and Java. Because of its wide range of features, OpenCV is commonly used in academic projects, research work, and real-time computer vision applications.

OpenCV provides built-in support for webcam access, image conversion, face detection, object tracking, and face recognition. [10] In the proposed system, OpenCV is used to capture video frames from the webcam, convert frames into grayscale, detect faces using Haar Cascade, train face data using LBPH, and recognize faces in real time.

2.5.4 Web-Based Face Recognition System

Flask is a lightweight Python web framework used for developing web applications. [11] It provides routing, request handling, template rendering, and communication between frontend and backend components. Flask is suitable for small and medium-level applications because it is simple, flexible, and easy to integrate with Python libraries.

In face recognition applications, a web interface improves user interaction. Instead of running commands manually, users can perform operations through buttons and forms. The user can collect face data, train the model, start recognition, stop recognition, view registered persons, and check recognition logs through the interface.

SYSTEM DESIGN

3.1 METHODOLOGY

The proposed Face Recognition System using Haar Cascade and LBPH Algorithm follows a modular and real-time computer vision-based methodology. The system is designed to detect human faces from webcam input, collect face samples, train the face recognition model, and recognize registered persons automatically. The methodology is divided into different phases, where each phase contributes to the complete working of the system.

The main objective of the methodology is to provide a simple, accurate, and user-friendly face recognition system using Python, OpenCV, Flask, Haar Cascade Classifier, and LBPH Face Recognizer. Haar Cascade is used for detecting the face from the video frame, while the LBPH algorithm is used for recognizing the person by comparing the detected face with the trained dataset.

• System Architecture and Design

The system is developed using a web-based architecture where the user interacts with the application through a Flask web interface. The frontend of the system is designed using HTML, CSS, and Bootstrap, while the backend logic is implemented using Python and Flask.

The backend is responsible for handling user requests such as collecting face data, training the model, starting face recognition, stopping recognition, and displaying recognition logs. OpenCV is used for accessing the webcam, processing video frames, detecting faces, and recognizing faces. The overall system is designed in such a way that the user can operate the face recognition system easily without directly using command-line instructions.

The system architecture includes the following main components:

- User Interface
- Flask Backend
- Webcam Input
- Haar Cascade Face Detection Module
- Dataset Storage
- LBPH Model Training Module
- Face Recognition Module
- Recognition Log Module

3.2 SYSTEM ARCHITECTURE

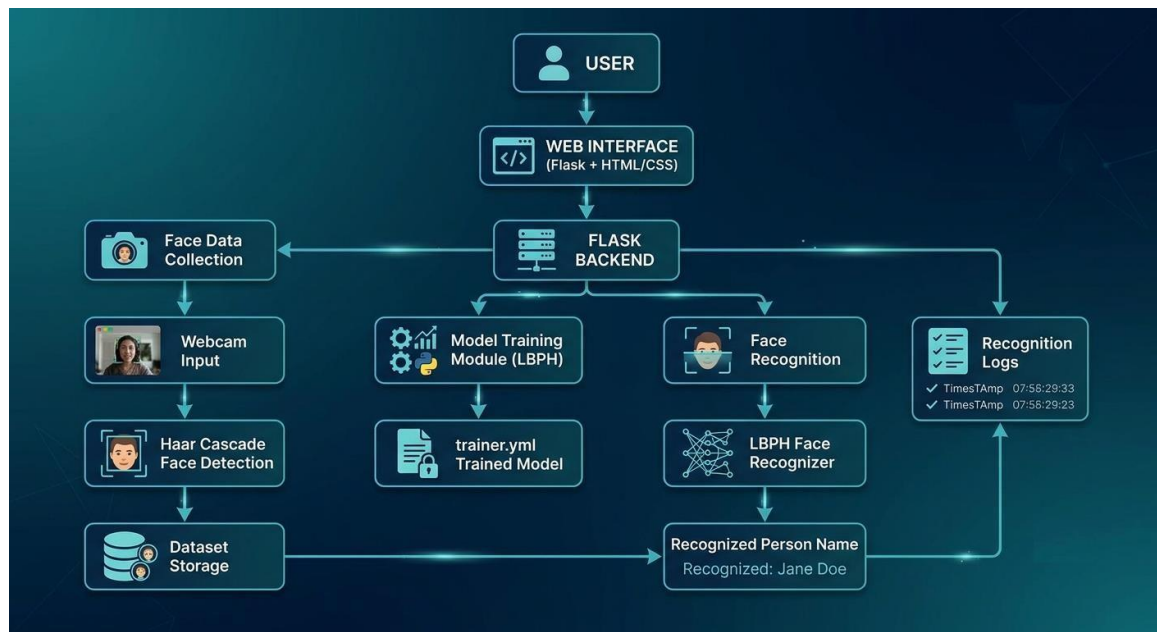


Fig 3.2.1 Schematic Architecture Diagram

The architecture of the Face Recognition System using Haar Cascade and LBPH Algorithm consists of multiple interconnected modules that work together to perform real-time face detection and recognition. The system begins with the user interface, where the user can access different options such as collect data, train model, start recognition, stop recognition, view registered users, and check recognition logs.

The frontend layer is developed using HTML, CSS, and Bootstrap. This layer provides a simple and user-friendly interface for interacting with the system. The user does not need to directly run Python files manually. Instead, all major operations are performed using buttons and forms available on the web interface.

All requests from the frontend are sent to the Flask backend. The Flask backend acts as the central processing unit of the system. It receives user requests, calls the required Python functions, manages dataset operations, starts webcam processing, trains the model, and controls the recognition process.

From the Flask backend, the workflow is divided into different functional modules:

3.3 DATA FLOW DIAGRAM

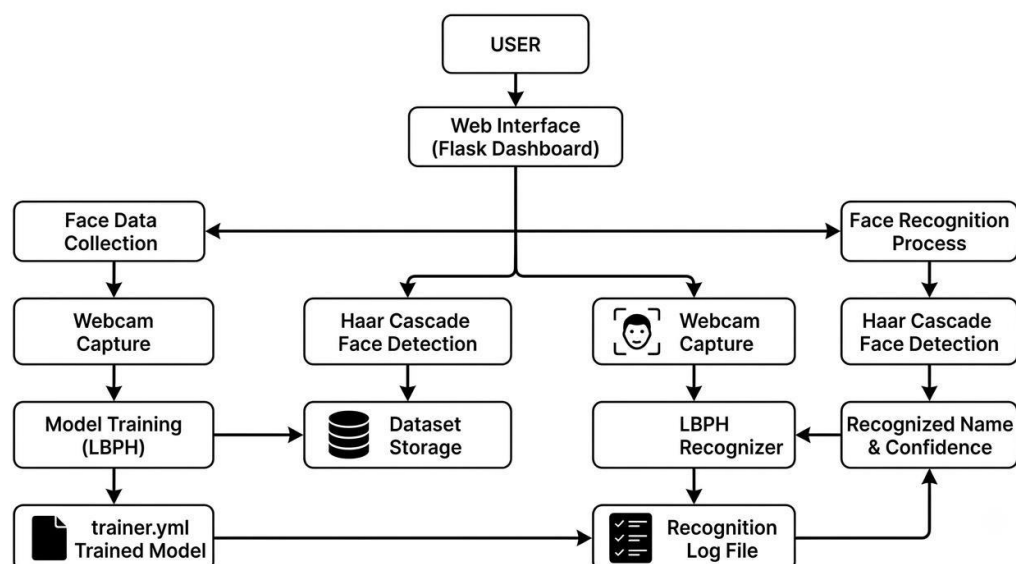


Fig 3.3.1 Data Flow Diagram of Face Recognition System

The Data Flow Diagram includes the following main elements:

1. User
2. Web Interface
3. Flask Backend
4. Webcam Input
5. Haar Cascade Face Detection
6. Dataset Storage
7. LBPH Model Training
8. Trained Model
9. Face Recognition
10. Recognition Logs

IMPLEMENTATION

The implementation of the Face Recognition System using Haar Cascade and LBPH Algorithm was carried out using a modular computer vision and web-based architecture. The system integrates Python programming, OpenCV image processing, Haar Cascade face detection, LBPH face recognition, dataset storage, model training, and Flask-based web interface. The project was developed to provide automatic face detection and recognition through webcam input.

The implementation process begins with the development of a user-friendly web interface. Through this interface, the user can perform important operations such as collecting face data, training the model, starting face recognition, stopping the recognition process, viewing registered persons, and checking recognition logs. The backend of the system is developed using Python and Flask. OpenCV is used to capture video frames from the webcam and perform image processing operations.

The main function of the system is divided into three important phases: face data collection, model training, and face recognition. During face data collection, the system captures multiple face images of a person using the webcam. Haar Cascade Classifier detects the face region from each video frame, and the detected face images are stored in the dataset folder. Each registered person has a separate folder containing their face samples.

After collecting the dataset, the training module is executed. The LBPH Face Recognizer reads the collected face images, assigns labels to each person, and trains the model. After training is completed, the trained model is saved as a model file. The label mapping is also stored separately so that the predicted label can be matched with the correct person's name during recognition.

During the recognition phase, the webcam captures live video frames. Haar Cascade detects the face from the frame, and the trained LBPH model compares the detected face with the stored trained data. If a matching face is found, the system displays the recognized person's name along with the confidence value. The system also stores recognition details in the recognition log file.

The project contains important files such as the main Flask application file, dataset folder, Haar Cascade XML file, trained model file, label mapping file, template files, and recognition log file. These files work together to complete the face recognition process.

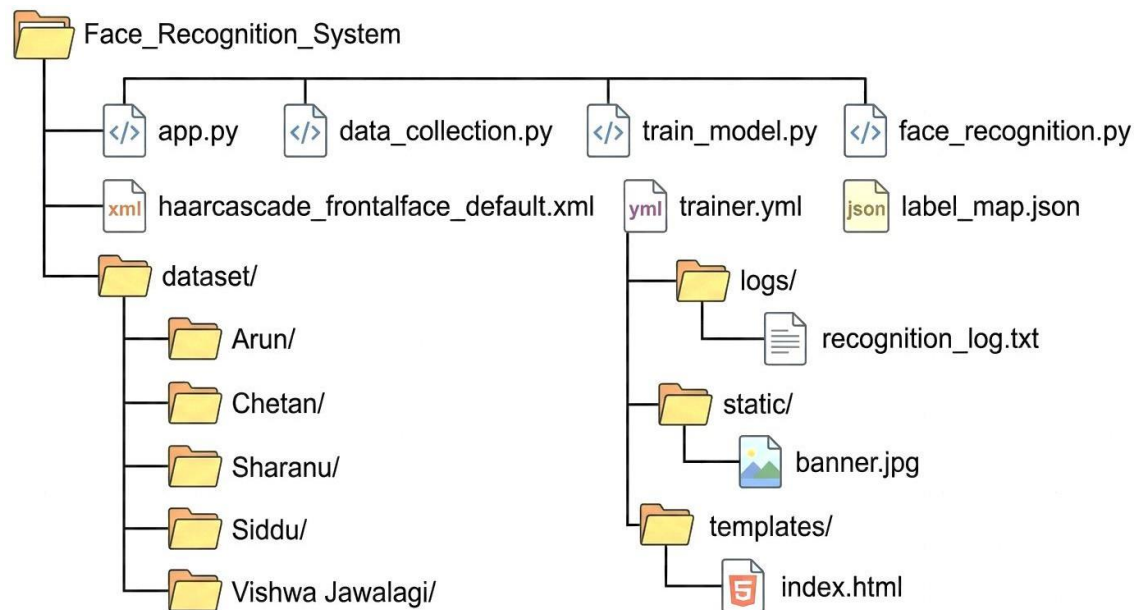


Fig 4.1 Project Folder Structure of Face Recognition System

The above figure shows the project folder structure of the face recognition system. It contains the main application file, dataset folder, Haar Cascade classifier file, trained model, label mapping file, templates, static files, and recognition log folder.

4.1 TOOLS AND TECHNOLOGIES USED

1. Python
2. OpenCV
3. Haar Cascade Classifier
4. LBPH Face Recognizer
5. Flask
6. HTML
7. CSS
8. Bootstrap
9. Visual Studio Code
10. Webcam
11. Dataset Folder
12. Trained Model File

4.2 MODULES DEVELOPED

The Face Recognition System is developed using a modular approach. Each module performs a specific function, and all modules work together to complete the face recognition process. The main modules developed in this project are explained below.

User Interface Module

Face Data Collection Module

Face Detection Module

Dataset Management Module

Model Training Module

Face Recognition Module

Recognition Log Module

Start and Stop Recognition Module

Label Mapping Module

File Management Module

RESULT AND DISCUSSION

The Face Recognition System using Haar Cascade and LBPH Algorithm was successfully implemented and tested using webcam input. The system demonstrated effective face detection, face data collection, model training, real-time face recognition, and recognition log maintenance. The project was developed using Python, OpenCV, Flask, Haar Cascade Classifier, and LBPH Face Recognizer.

The main aim of the system was to detect and recognize registered persons automatically through live webcam video. The system first collects face images of registered persons and stores them in the dataset folder. After data collection, the model is trained using the LBPH algorithm. During recognition, the webcam captures the live face, Haar Cascade detects the face region, and the trained LBPH model recognizes the person.

The system provides a simple web-based dashboard where the user can perform different operations such as collecting face data, training the model, starting recognition, stopping recognition, viewing registered persons, and checking recognition logs. This makes the system easy to use and suitable for academic demonstration.

CONCLUSION

The Face Recognition System using Haar Cascade and LBPH Algorithm successfully addresses the need for an automated and contactless identification system. The project demonstrates how computer vision techniques can be used to detect and recognize human faces through webcam input. The system was developed using Python, OpenCV, Flask, Haar Cascade Classifier, and LBPH Face Recognizer.

The main objective of the project was to design a system that can collect face data, train the recognition model, and recognize registered persons in real time. Haar Cascade Classifier was used for detecting the human face from live video frames, while the LBPH algorithm was used for recognizing the detected face by comparing it with the trained dataset. The system successfully performed face detection, dataset creation, model training, face recognition, and recognition log maintenance.

The project provides a simple web-based interface that allows users to operate the system easily. Through the interface, users can collect face images, train the model, start recognition, stop recognition, view registered persons, and check recognition logs. This makes the system user-friendly and suitable for academic demonstration as well as small-scale practical applications.

The system reduces manual effort in identity verification and provides a faster method for recognizing

registered persons. It can be used as a foundation for applications such as attendance management, security monitoring, access control, and visitor tracking. Since the project uses open-source tools and a normal webcam, it is cost-effective and easy to implement.

The performance of the system depends on factors such as lighting conditions, camera quality, face position, and dataset quality. The system works better when the face is clearly visible and properly trained. Although the current system is suitable for small-scale applications, it can be improved in the future by adding database integration, automatic attendance marking, advanced security features, cloud storage, and deep learning-based face recognition methods.

REFERENCES

- [1] Viola, P., & Jones, M., "Rapid Object Detection using a Boosted Cascade of Simple Features," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001.
- [2] Viola, P., & Jones, M., "Robust Real-Time Face Detection," International Journal of Computer Vision, 2004.
- [3] Lienhart, R., & Maydt, J., "An Extended Set of Haar-like Features for Rapid Object Detection," Proceedings of the International Conference on Image Processing, 2002.
- [4] Ahonen, T., Hadid, A., & Pietikainen, M., "Face Description with Local Binary Patterns: Application to Face Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006.
- [5] Ojala, T., Pietikainen, M., & Maenpaa, T., "Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.
- [6] Bradski, G., "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
- [7] OpenCV Documentation, "Haar Cascade Face Detection and Face Recognition Modules," OpenCV Library.
- [8] Python Software Foundation, "Python Programming Language Documentation," Python Software Foundation.
- [9] Flask Documentation, "Flask Web Framework for Python," Pallets Projects.
- [10] Gonzalez, R. C., & Woods, R. E., Digital Image Processing, Pearson Education.
- [11] Forsyth, D. A., & Ponce, J., Computer Vision: A Modern Approach, Pearson Education.
- [12] Jain, A. K., Ross, A., & Prabhakar, S., "An Introduction to Biometric Recognition," IEEE Transactions on Circuits and Systems for Video Technology, 2004.
- [13] Turk, M., & Pentland, A., "Eigenfaces for Recognition," Journal of Cognitive Neuroscience, 1991.
- [14] Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J., "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.
- [15] Kumar, A., & Kaur, A., "Face Recognition Techniques: A Review," International Journal of Computer Applications.