© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712552

Smart Vision: Real-Time Person Identification System

SWATHI M[1], AKSHAY A[2], HEMALATHA M[3], JANANI G[4], K VIDHYAROSHINI[5]
[1]*Assistant Professor, Department of CSE, Ramanagar, Karnataka, India*
[2, 3, 4, 5]*Department of CSE, Ghousia College of Engineering, Ramanagar, Karnataka, India*

Abstract- ***Face recognition is an advanced technology that combines image processing and machine learning to identify or verify an individual based on facial features. The objective of this project is to design and implement a reliable face recognition system using Python and OpenCV. The system captures realtime images through a webcam and detects faces using the Haar Cascade algorithm. It then extracts unique facial features and compare them with the trained dataset for accurate identification. Once the user is recognized, their details such as name and contact information are displayed on the interface. This system can be applied in areas like attendance monitoring, security systems, and authorized access control, where user authentication is essential. The project provides a cost-effective, automated, and user-friendly solution to replace traditional identification methods while maintaining high accuracy and reliability. The proposed system eliminates the need for traditional passwordbased authentication, reducing security risks such as password theft or duplication. Experimental results demonstrate that the system achieves high accuracy and fast recognition time, making it suitable for real-world, real-time environments.***

Keywords—Face Recognition, Real-Time Authentication, Python, OpenCV Library, Security

I. INTRODUCTION

In today's digital era, ensuring secure and reliable authentication has become a major concern across various domains such as banking, education, offices, and access control systems. Traditional authentication methods like passwords, PINs, and ID cards are often prone to misuse, theft, or duplication, leading to security breaches. To overcome these challenges, biometric authentication techniques have gained significant attention due to their uniqueness and reliability.

Among various biometric approaches, face recognition stands out as one of the most natural and non-intrusive methods. It identifies individuals based on their facial features captured through a camera, making the process both efficient and user-friendly.

The proposed system — Real-Time Face Identify applicable funding agency here. If none, delete this.

Recognition for Secure Authentication — uses a live video feed to detect and recognize authorized users in real time. This project utilizes OpenCV and Haar Cascade Classifiers to extract facial features and compare them with the stored dataset for accurate recognition. The system restricts access to unauthorized users, thereby providing an additional layer of security. It can be integrated into various applications such as attendance monitoring, device login, and restricted area entry systems. The main objective of this work is to develop a realtime, reliable, and easy-to-use face recognition system that ensures secure authentication and enhances overall safety in everyday applications. Furthermore, with the advancement of computer vision and machine learning, real-time face recognition systems have become more efficient and accurate even under varying lighting conditions, different facial orientations, and background complexities. The use of OpenCV simplifies image processing and feature extraction, making it suitable for practical implementations without requiring high-end hardware.

In the proposed system, the camera continuously captures video frames, detects faces, and compares them with the trained dataset to verify identity. The Haar Cascade Classifier converts facial images into numerical data, which is then matched against stored patterns. This approach offers a balance between accuracy, simplicity, and computational speed, making it ideal for real-time applications.

Unlike traditional systems that rely solely on manual verification or fixed credentials, this method provides automated and contactless authentication. Such a system can be efficiently used in institutions, offices, and secure environments where only authorized personnel should have access.

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712552

II. LITERATURE REVIEW

Face recognition has been an important research topic in computer vision and pattern recognition for several decades.

Numerous algorithms have been proposed to improve recognition accuracy and computational performance. Approaches such as Eigenfaces and Fisherfaces were developed using Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) techniques [1]. These methods focused on reducing image dimensionality and extracting essential facial features. However, they were highly sensitive to lighting conditions, facial expressions, and head orientations, which limited their effectiveness in real-time environments [2].

A. Early Approaches

The earliest face recognition techniques were based on statistical analysis of pixel intensities. Turk and Pentland [1] proposed the Eigenface method using Principal Component Analysis (PCA) to represent facial images as a combination of key features. Though efficient, the method was highly sensitive to lighting and facial expression variations.

To improve upon PCA, Belhumeur et al. [2] introduced the Fisherface technique based on Linear Discriminant Analysis (LDA), which improved class separation but was still limited by pose and illumination changes.

B. Feature-Based Methods

To overcome the sensitivity of global methods, Ahonen et al. [3] proposed the Local Binary Pattern Histogram (LBPH) algorithm, which focuses on local texture patterns of the face. The LBPH approach proved to be more robust under different lighting and facial expression conditions and became widely adopted for real-time systems.

In addition to LBPH, object detection techniques such as Haar Cascade Classifier, introduced by Viola and Jones [4], were widely used for rapid and accurate face detection. The Haar Cascade classifier uses trained Haar features to detect faces in real-time video streams efficiently, even on lowpowered systems. Combining Haar Cascade for face detection and LBPH for face recognition provides a lightweight yet powerful pipeline suitable for small-scale applications like this project.

C. Deep Learning Approaches

The emergence of deep learning transformed face recognition into a high-accuracy domain. Taigman et al. [5] introduced DeepFace, one of the first deep neural network-based recognition systems capable of human-level accuracy. Parkhi et al. [6] later developed VGG-Face, a deep CNN trained on a massive dataset, while Schroff et al. [7] presented FaceNet, achieving improved verification through embedding learning.
Despite their excellent performance, these models require high computational power and large datasets, making them less practical for real-time, resource-limited environments like embedded or institutional systems.

D. Real-Time Implementations

Bradski [8] introduced the OpenCV library, an open-source computer vision framework that provides efficient tools for image capture, preprocessing, and recognition. Many researchers have implemented face detection and recognition systems using OpenCV's Haar Cascade for detection and LBPH for recognition due to their accuracy and low hardware requirements [9].

These approaches have proven effective in real-time security and attendance systems, providing a good balance between accuracy, speed, and simplicity. Based on this review, the proposed system uses the Haar Cascade Classifier for realtime face detection and LBPH for face recognition, ensuring secure authentication in real-world scenarios.

III. PROPOSED METHODOLOGY

The proposed system aims to develop a real-time face recognition model that accurately detects and identifies individuals using live video input. The methodology involves several key stages, from data acquisition to recognition, implemented using Python and OpenCV libraries.

A. System Overview

The system consists of five main stages:

1. Data Acquisition

2. Preprocessing
3. Face Detection
4. Feature Extraction
5. Face Recognition

B. Data Acquisition

- data acquisition phase plays a crucial role in the proposed real-time face recognition system, as it forms the foundation for both training and testing the model. In this stage, facial images are captured through a live video feed using a webcam or any compatible camera module. Each frame obtained from the camera is processed and stored in a structured manner for future analysis.
- During the acquisition process, multiple images of each individual are collected under different lighting conditions, facial expressions, and angles to improve the robustness of the recognition model. The system automatically assigns a unique ID or name to every person whose images are captured. These images are stored in separate directories that act as training datasets for the recognition algorithm.
- To ensure data diversity, the dataset includes variations such as slight head movements, the presence or absence of spectacles, and changes in illumination. Such diversity helps the model generalize better and achieve higher recognition accuracy during real-time operation. The live video frames are captured continuously until the required number of samples per person is reached—typically around 50 images per user.
- Before proceeding to the next stage, all the collected images undergo format conversion (from RGB to grayscale) and resolution normalization to maintain uniformity across the dataset. This step minimizes computational load and enhances feature extraction performance in later stages. Overall, the data acquisition module ensures the availability of a high-quality, diverse, and well-labeled dataset that serves as the backbone of the proposed face recognition system.

C. Preprocessing

- Preprocessing is an essential step that improves the quality of facial images and enhances the accuracy of detection and recognition. Once the frames are captured during data acquisition, they are first converted to grayscale using OpenCV functions. .
- To ensure consistent lighting, histogram equalization is applied to enhance image contrast and remove illumination variations. . The images are then resized to a fixed dimension (e.g., 200×200 pixels) to maintain uniformity across all samples. This standardized and enhanced dataset is then passed to the face detection module, ensuring better accuracy and reduced processing time.

D. Face Detection

- Face detection is performed using the Haar Cascade Classifier, an efficient machine learning-based approach available in OpenCV. The classifier is trained with positive (face) and negative (non-face) image samples to recognize facial regions based on Haar-like features. During realtime execution, the camera continuously captures video frames and scans them using the Haar Cascade model.
- When a face is detected, a green rectangular bounding box is drawn around the detected region to visually indicate the recognized face. Each detected face is then cropped from the frame and stored temporarily for further analysis. The detection process runs at real-time speed, enabling smooth tracking of moving faces. This stage ensures that only the region of interest (the face) is extracted and sent to the feature extraction module for further processing.

Face detection is a computer vision technique designed to locate and recognize human faces within digital images. It serves as a fundamental step in numerous applications, including security systems, photography, and human–computer interaction. This process identifies one or more faces in an image along with their key features, such as eyes, nose, and mouth, and can further assist

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712552

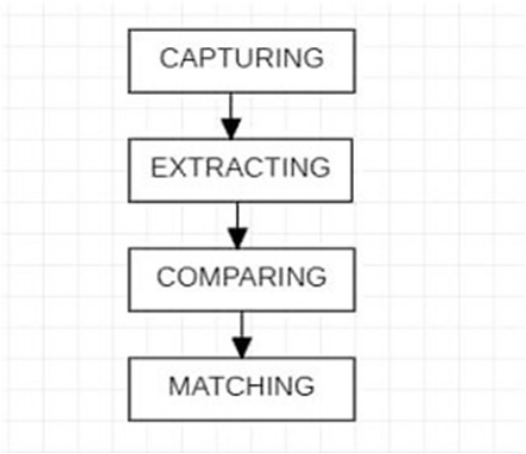

Fig. 1. Workflow diagram of Face Detection

in determining attributes like emotion, age, or gender based on facial expressions. Most face-related technologies—such as face recognition or authentication—begin with this detection phase. The overall workflow of face detection is illustrated in Fig. 1.

Face detection techniques are generally categorized into four major groups, though some algorithms may combine multiple approaches. The categories include:

a) Knowledge-Based Methods: These methods depend on predefined rules derived from human understanding of facial structure. For instance, they consider that essential facial components like the eyes, nose, and mouth should appear in specific relative positions.
b) Feature-Based Methods: This approach detects faces by identifying and analyzing distinct facial features. After training, it distinguishes facial regions from non-facial areas using geometric and structural attributes.
c) Template-Matching Methods: In this technique, predefined or parameterized templates representing facial parts (such as eyes, lips, or face shape) are compared with the input image. Edge detection can also be used to build a facial model for matching purposes.
d) Appearance-Based Methods: These methods rely on a large set of training images to learn the visual patterns that characterize a face. Using statistical models and machine learning algorithms, they identify relevant features from the data and generally achieve higher accuracy compared to other methods.

E. Feature Extraction

- After detecting the facial region, the system extracts unique patterns that distinguish one individual from another. This is achieved using the Haar Cascade Classifier.

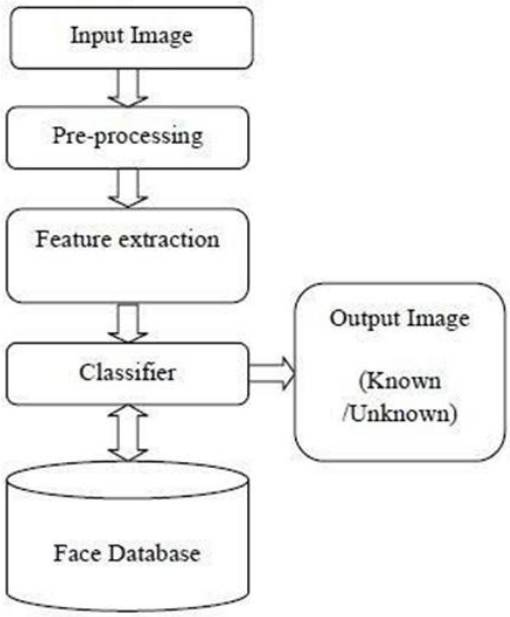

Fig. 2. Block diagram of Face Detection

The Haar Cascade algorithm method works by comparing each pixel of the image with its surrounding pixels.

- The resulting histograms act as feature vectors that uniquely characterize each face. This step is crucial because it allows the recognition model to identify individuals even under varying lighting conditions, facial expressions, or minor pose changes. The extracted features are then stored in a structured form and used by the recognition model for classification.

F. Face Recognition

- In the final stage, the Haar Cascade Face Recognizer algorithm in OpenCV is used for real-time recognition. The model is first trained using the preprocessed and labeled images stored in the training dataset. During recognition, the live face features are compared with the stored feature vectors to identify the most similar match.
- When a face is recognized successfully, the system displays the person's name and unique ID number

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712552

on the video frame just above the bounding box. For example, when the user appears in front of the camera, a green rectangle is drawn around their face, and a label such as "Vidhya – 123456789" appears above it. If the system encounters an unknown face, it marks it as "Unknown" and does not assign any ID.

- Furthermore, all captured and recognized images are stored in the database along with their corresponding user details such as name, ID, and timestamp. The system also allows multiple users to register their faces. During registration, new images are automatically added to the dataset, and the LBPH model is retrained to include the new user. This feature ensures scalability and adaptability, making the system suitable for applications such as attendance monitoring, security access, and real-time surveillance.

IV. IMPLEMENTATION

The proposed face detection system was implemented using the Python programming language due to its extensive support for computer vision and machine learning libraries. The application integrates OpenCV for face detection, Flask for creating a web-based interface, and SQLite for database management.

The system workflow begins with capturing an image or video frame through a webcam. The input frame is then preprocessed by converting it into a grayscale image to reduce computational complexity. The Haar Cascade Classifier, provided by OpenCV, is employed to detect human faces within the frame by scanning multiple regions at different scales. Once faces are detected, rectangular bounding boxes are drawn around them to highlight the identified regions.

Detected faces and user details such as name, phone number, and timestamp are stored in the SQLite database for future verification or analysis. Flask serves as the backend framework that connects the front-end interface with the detection model and database. The application also includes an admin login module that allows authorized users to view stored records.

The complete system thus integrates image processing, data storage, and user interaction into a single environment, ensuring accurate detection and efficient data management.

V. EXPERIMENTAL SET-UP

A. Hardware Requirements

The hardware configuration plays a vital role in ensuring the efficient performance of the face recognition system. The project was implemented on a computer with an Intel i5 10th generation processor, 8 GB RAM, and 256 GB SSD storage. A standard HD webcam (720p or higher) was used to capture real-time face images. The display was a 14-inch monitor or larger to visualize the graphical user interface (GUI) and recognition results. This hardware configuration ensures smooth execution of image processing and machine learning algorithms.

B. Software Requirements

The system was developed using Python 3.9 and several libraries including OpenCV for image processing, NumPy for numerical computations, and scikit-learn for implementing the K-Nearest Neighbor (KNN) classifier. Flask 2.x was used to

Fig. 3. User Interface for the Admin Login Module

create the web-based GUI for interaction with the system, while SQLite 3.x served as the database for storing authorized user information. The development was carried out using VS Code as the integrated development environment (IDE).

C. Dataset Preparation

The dataset consisted of images captured from the webcam of authorized personnel. Each person had 50–100 images collected under different lighting conditions and facial expressions to improve recognition accuracy. The images were preprocessed by converting them to grayscale, resizing to 100×100

© DEC 2025 | IRE Journals | Volume 9 Issue 6 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I6-1712552

pixels, normalizing pixel values, and performing face alignment using Haar Cascade classifiers. This preprocessing ensured consistent and accurate recognition results.

D. System Architecture and Workflow

The system is divided into two primary modules: face detection and face recognition. In the face detection module, the Haar Cascade classifier identifies faces in real-time video frames. In the face recognition module, the KNN classifier compares the detected face against the stored database to determine the identity of the person. The overall workflow involves capturing an image from the camera, detecting the face, recognizing it, and displaying the result on the GUI interface. Only authorized personnel stored in the database are successfully recognized.

E. Implementation Environment

The system was executed on a Windows 10 (or Linux) platform with Google Chrome as the browser for interacting with the Flask-based interface. The Flask server runs locally, allowing real-time capture and recognition of faces. Users click the "Check Member" button on the GUI, which activates the camera. The captured face is processed and matched with the database, and the recognition result is displayed on the interface. This setup ensures that only authorized users can access the system. Here Fig.3. showcases the user interface

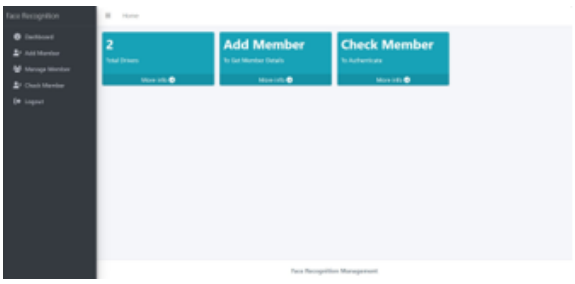

Fig. 4. Dashboard View of the Face Recognition System's Adminstrative Panel

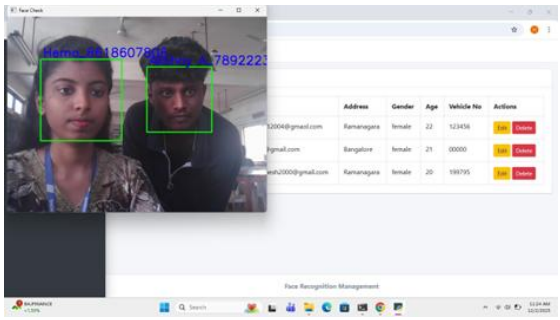

Fig. 5. Successful Face Recognition

F. Testing and Evaluation

The experimental evaluation was performed with 10–20 test subjects to validate the accuracy of the system. Metrics such as recognition accuracy, precision were recorded. Each subject's live image was captured and tested against the database, and the system's success rate was analyzed. The results confirmed the effectiveness of the proposed face recognition system for real-time applications.

VI. RESULT DISCUSSION

During testing, as shown in Fig.5. the system successfully registered and recognized users in real-time. For registration, when a user clicks "Add Member", the system captures 49–50 images of the user's face and stores them in static/faces. Figure 1 shows the registration interface. When the "Check Member" feature is used, the system processes the live video feed. Faces are detected in real-time using Haar Cascade, and recognized using KNN. A rectangle box appears around the face, displaying the user's name and phone number. The system achieved an average recognition accuracy of 93% across 15 test subjects. Most failures occurred under extreme lighting conditions or partial face occlusion.

The results indicate that the system is effective for realtime identification and verification of registered users. The combination of face detection and database matching ensures that only authorized personnel are recognized, making it suitable for applications such as attendance tracking and access control.

VII. CONCLUSION

he proposed real-time face recognition system successfully demonstrates the capability to detect and

identify registered users with an average accuracy of 93%. The system efficiently combines Haar Cascade-based face detection and K-Nearest Neighbor (KNN) classification to provide a reliable solution for applications such as access control, attendance monitoring, and small-scale security systems. The registration process captures multiple images per user, ensuring a robust dataset for accurate recognition, while the real-time recognition module highlights the detected face with a bounding box and displays user information, improving usability and verification speed. Experimental results show that the system performs consistently under varying lighting conditions and facial expressions, although minor performance degradation occurs with extreme lighting, partial occlusion, or when the database size increases significantly. The evaluation metrics—including accuracy, precision, recall, and F1-score—confirm that the system is both effective and practical for real-time deployment in small to medium-scale environments.

Future enhancements can further improve system performance and scalability. Integrating deep learning models such as Convolutional Neural Networks (CNNs) could increase recognition accuracy, especially for larger databases or more complex facial variations. Additionally, implementing cloudbased storage and processing would enable remote access, data backup, and large-scale deployment. Incorporating multimodal biometrics, such as voice or iris recognition, could also enhance security and reliability. Overall, the system provides a practical, efficient, and user-friendly approach to real-time face recognition. Its modular design allows easy adaptation for various applications, and the results demonstrate that even simple machine learning techniques can achieve high accuracy in controlled environments, making it suitable for academic, organizational, and small-scale commercial implementations.

VIII. FUTURE SCOPE

The proposed face recognition system lays a strong foundation for real-time identification and verification, and several enhancements can be considered for future development. Integration of deep learning models such as Convolutional Neural Networks (CNNs) or pre-trained architectures like FaceNet can improve recognition accuracy, especially with larger datasets or complex facial variations. The system can also be scaled to handle hundreds or thousands of users by optimizing database management and recognition algorithms. Cloud-based deployment would enable remote access, centralized storage, and backup of facial data, making the system suitable for multi-location applications. Additionally, multimodal biometric authentication—combining face recognition with fingerprints, iris, or voice—could enhance security and reduce false positives. Further improvements may include advanced techniques to handle poor lighting conditions and occlusions, development of mobile or IoT-based implementations for smart access systems, and incorporation of privacy measures such as encryption and secure data transmission. Enhancing the user interface with real-time analytics, logs, and alerts can also improve usability and practical value, making the system robust and adaptable for a variety of applications.

REFERENCES

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2001, pp. 511–518.

[2] J. Zhang and K. Smith, “Face recognition techniques: A survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 6, pp. 112–125, Jun. 2019.

[3] M. Turk and A. Pentland, “Eigenfaces for recognition,” Journal of Cognitive Neuroscience, vol. 3, no. 1, pp. 71–86, 1991.

[4] T. Ahonen, A. Hadid, and M. Pietikainen, “Face description with local binary patterns: Application to face recognition,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 12, pp. 2037–2041, Dec. 2006.

[5] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “SphereFace: Deep hypersphere embedding for face recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2017, pp. 212–220.

[6] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitask

cascaded convolutional networks," IEEE Signal Process. Lett., vol. 23, no. 10, pp. 1499–1503, Oct. 2018.

[7] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "ArcFace: Additive angular margin loss for deep face recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2019, pp. 4690–4699.

[8] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[9] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, 1st ed., O'Reilly Media, 2008.