Facial Recognition-Based Cabin Access Verification System

Internship Project at SoulVibe.Tech

Boddana Tirumala Rao
B.Tech Computer Science Engineering
Intern, SoulVibe.Tech
btirumalarao27@gmail.com

Contents

1 Introduction **2**

2 System Architecture and Components **3**
2.1 Core Components 3
2.2 Folder Structure 3

3 Key Features and Functionality **4**
3.1 Face Detection & Recognition 4
3.2 Access Validation 4
3.3 Add New User Functionality 4
3.4 Access Logging 4
3.5 User Database Management 4

4 System Flow (Conceptual) **5**
4.1 Access Control Flow 5
4.2 User Registration Flow 5

5 Challenges and Learnings **6**
5.1 Challenges 6
5.2 Learnings 6

6 Conclusion **7**

1 Introduction

In today's rapidly advancing technological landscape, the need for secure, efficient, and user-friendly access control systems has become more important than ever. Traditional methods such as keycards, PIN codes, or physical locks are either prone to security breaches or lack the convenience required in dynamic environments. As a result, biometric systems, particularly facial recognition, have emerged as a leading solution due to their accuracy, non-intrusive nature, and ease of deployment.

This project aims to build a real-time, facial recognition-based cabin access verification system that ensures only authorized personnel can access restricted areas. Unlike cloud-based systems that depend on internet connectivity and raise privacy concerns, our solution is designed to function completely offline. It leverages a local database to store authorized user information, enabling fast and secure access control without external dependencies.

The system is built using Python and utilizes state-of-the-art libraries like `face_recognition` and OpenCV to perform tasks such as face detection, feature encoding, and real-time recognition. Its modular architecture not only supports core functionalities like access verification and logging but also includes the capability to register new users on the fly. Each access attempt is logged with a timestamp, enabling administrators to track user activity for security audits or attendance monitoring.

This project was undertaken as an internship at SoulVibe.Tech with the goal of applying theoretical knowledge to a real-world security application. Through this internship, the project not only showcases the practical implementation of computer vision techniques but also emphasizes software design principles, system integration, and user interaction within a constrained and realistic deployment environment.

2 System Architecture and Components

The application adopts a modular design approach. Python, due to its extensive library support and readability, is the primary programming language. Key libraries are used to implement face detection, recognition, image processing, and data management.

2.1 Core Components

- **face_recognition:** Provides simple yet powerful APIs for face detection and encoding comparison using deep learning models.
- **OpenCV (cv2):** Handles webcam input, frame capture, image conversion, and user interface rendering (boxes, text).
- **NumPy:** Facilitates efficient numerical operations, especially when working with face encodings as vectors.
- **Pandas:** Supports reading and writing user and log data in structured tabular formats.
- **CSV, OS:** Used for file system traversal, directory creation, and manipulation of CSV files.
- **Datetime, Time:** Responsible for timestamping events, measuring processing time, and managing delays in the UI.

2.2 Folder Structure

```
facial_access_system/
     train/                   # Stores authorized user face images
     access_log.csv           # Logs all access attempts
     authorized_users.csv     # Registered users (Name, Encoding, Timestamp)
     add_user.py              # Script to register new users
     access_control.py        # Real-time access control script
     report.pdf               # This report
     presentation.pptx        # Project slides
     README.md                # Project instructions
```

Each component is modular, making the system maintainable and extensible.

3 Key Features and Functionality

3.1 Face Detection & Recognition

- Real-time video frames are captured and resized to optimize performance.
- Frames are converted to RGB, and faces are detected using HOG + CNN models.
- Each face is encoded as a 128-dimensional vector using the `face_recognition` library.
- Detected faces are compared against known encodings to check for matches.

3.2 Access Validation

- The `authorized_users.csv` file stores names and corresponding face encodings.
- A matching face results in "Access Granted" with a green box overlay; otherwise, "Access Denied" with a red box.
- The match threshold is adjustable to balance false accept and reject rates.

3.3 Add New User Functionality

- Pressing 'A' during execution pauses recognition and starts the user registration routine.
- User is prompted for a name; a face snapshot is taken, encoded, and added to the database.
- System validates that only one face is present in the frame. It shows warnings if multiple or no faces are detected.
- Images are saved in the `train/` folder for audit purposes.

3.4 Access Logging

- Each access attempt—granted or denied—is logged into `access_log.csv`.
- Fields include: Name (or "Unknown"), Timestamp, and Access Status.
- This file can be used for auditing and generating attendance or security reports.

3.5 User Database Management

- At launch, the system reads the CSV and loads all user encodings into memory.
- When a new user is registered, both the CSV and in-memory list are updated in real time.
- Allows dynamic, live database updates without restarting the application.

4 System Flow (Conceptual)

4.1 Access Control Flow

1. System initializes and loads existing user encodings.
2. Webcam feed starts, capturing frames in real-time.
3. Frames are resized and converted to RGB format.
4. System detects faces and computes encodings.
5. Detected encodings are compared against stored encodings.
6. Access decision is made; bounding boxes and status labels are drawn.
7. Attempt is logged in CSV file.
8. System listens for:
   - ‘Q‘: Quit the application
   - ‘A‘: Start the user registration module
9. The process loops for continuous access control.

4.2 User Registration Flow

1. Admin presses ‘A‘ to register a new user.
2. Current webcam session is paused, and the user is prompted for their name.
3. A single face image is captured.
4. If the face detection finds exactly one face:
   - Encoding is created.
   - Name, encoding, and timestamp are saved to `authorized_users.csv`.
   - Image is saved to the `train/` directory.
5. Control is returned to the main access loop.

5 Challenges and Learnings

5.1 Challenges

- **Real-time Speed vs Accuracy:** Higher-resolution frames improve recognition accuracy but reduce speed. This was mitigated by resizing frames and limiting recognition frequency.
- **Encoding Storage in CSV:** Directly saving NumPy arrays in CSV is not straightforward. We used string conversion and evaluated them back to arrays during loading.
- **Webcam Conflicts:** Handling camera initialization and release during transitions between modules required careful resource management.
- **Multiple Faces:** Preventing registration errors when multiple faces were present required robust detection logic and user warnings.

5.2 Learnings

- Practical integration of facial recognition models with real-time video processing.
- Use of persistent data structures for maintaining biometric user databases.
- Importance of error handling and user feedback in live systems.
- Hands-on experience with modular design and scalable architecture for computer vision systems.

6 Conclusion

The Facial Recognition-Based Cabin Access Verification System marks a significant step toward intelligent and secure workspace access management. By leveraging advanced computer vision techniques and a fully offline architecture, the system offers a practical and scalable solution to modern access control challenges. Its modularity, real-time capabilities, and built-in user registration and logging features make it suitable not only for cabins but also for broader applications such as smart offices, labs, and secure zones.

Throughout the development process, this project emphasized not just technical implementation but also thoughtful user experience design, robust data handling, and real-world usability. It successfully demonstrated how Python and its ecosystem of libraries can be used to integrate deep learning-based face recognition into an end-to-end application that balances both performance and usability.

This project also provided valuable hands-on experience in system design, debugging, real-time video processing, and biometric data management. By engaging with real-time hardware (webcam), persistent data storage (CSV files), and user feedback mechanisms, the system highlights the importance of harmonizing software and hardware layers.

Future work may include enhancing the system with multi-factor authentication (e.g., facial recognition + RFID), introducing liveness detection to prevent spoofing attacks, enabling mobile notifications for unauthorized access, and integrating cloud synchronization for remote administration. Additionally, the user interface could be further refined with a graphical dashboard for easier monitoring and management.

In summary, the project lays a strong foundation for secure, real-time biometric access systems and serves as a powerful learning experience in applied machine learning, computer vision, and software engineering principles.