Raspberry Pi-based Face Recognition Door Lock System

by

SEIFELDIN SHERIF ELNOZAHY

1191102388

Session 2024/2025

The project report is prepared for

Faculty of Engineering

Multimedia University

in partial fulfilment for

Bachelor of Engineering (Hons) Electronics

majoring in Computer

FACULTY OF ENGINEERING

MULTIMEDIA UNIVERSITY

February 2025

DECLARATION

I hereby declare that this work has been done by myself and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

I also declare that pursuant to the provisions of the Copyright Act 1987, I have not engaged in any unauthorised act of copying or reproducing or attempt to copy / reproduce or cause to copy / reproduce or permit the copying / reproducing or the sharing and / or downloading of any copyrighted material or an attempt to do so whether by use of the University's facilities or outside networks / facilities whether in hard copy or soft copy format, of any material protected under the provisions of sections 3 and 7 of the Act whether for payment or otherwise save as specifically provided for therein. This shall include but not be limited to any lecture notes, course packs, thesis, text books, exam questions, any works of authorship fixed in any tangible medium of expression whether provided by the University or otherwise.

I hereby further declare that in the event of any infringement of the provisions of the Act whether knowingly or unknowingly the University shall not be liable for the same in any manner whatsoever and undertakes to indemnify and keep indemnified the University against all such claims and actions.

Signature: SEIF
Name: SEIFELDIN SHERIF ELNOZAHY
Student ID: 1191102388
Date: 21-FEB-2025

ACKNOWLEDGEMENT

I would like to extend my heartfelt gratitude to my supervisor, Dr. Chinnaiyan Senthilpari, for his invaluable guidance and unwavering support throughout this project. His expertise and insightful advice have been pivotal in shaping both the direction and execution of my work. This project could not have been completed without his consistent mentorship and constructive feedback. I am also deeply thankful to my moderator, Dr. Lee Chu Liang, for his thoughtful insights and suggestions, which have significantly enhanced and refined this project.

My sincere appreciation goes to the faculty and staff of the Faculty of Engineering for equipping me with a strong foundation in engineering principles. Their dedication and encouragement have been instrumental in my academic growth.

A special note of thanks to my parents, whose belief in my abilities has been a driving force throughout this journey, and to my siblings for their unwavering support and motivation. I am equally grateful to my friends, whose encouragement and companionship have provided both strength and inspiration. Their presence has made this challenging journey more manageable and rewarding.

Finally, I am grateful to everyone who has contributed to this project, whether through technical assistance or moral support. Your contributions have been invaluable in ensuring its successful completion. Thank you all.

ABSTRACT

In the rapidly advancing domain of security technology, access control systems play a pivotal role in safeguarding residential and commercial environments. This project focuses on designing and implementing a Raspberry Pi-based face recognition door lock system, integrating artificial intelligence and computer vision to deliver a robust, efficient, and user-friendly solution. The system leverages facial recognition as its primary authentication mechanism, utilising the Raspberry Pi as the central processing unit. Key components include a camera module for real-time image capture, a relay module for controlling a solenoid lock, and software tools like OpenCV for image processing. The design incorporates advanced features such as adaptive learning, enabling the system to enhance recognition accuracy for authorised users over time, and emotion detection using the DeepFace library to interpret user emotional states. Additionally, the system adapts to challenging lighting conditions and varying distances while providing real-time notifications for remote monitoring. Testing revealed that the system achieves an average face detection time of 564.43 ms and an average emotion detection time of 327.63 ms, ensuring real-time performance. Significant findings include successfully implementing adaptive face recognition, ensuring the system evolves with usage, and seamlessly integrating multiple enhancements like real-time alerts and emotion detection. Testing demonstrated high accuracy in face recognition, even under varying environmental conditions. The modular design approach allowed efficient hardware-software integration and scalability for diverse applications. In conclusion, this project successfully developed an intelligent face recognition door lock system, combining the accessibility of Raspberry Pi hardware with the versatility of open-source software libraries. By addressing the limitations of traditional access control methods, the system provides a practical, scalable, and affordable solution, underscoring the potential of biometric technologies in modern security systems.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

FYP	Final Year Project
AI	Artificial Intelligence
CNN	Convolutional Neural Network
CSI	Camera Serial Interface
DC	Direct Current
GPIO	General Purpose Input/Output
HOG	Histogram of Oriented Gradients
IoT	Internet of Things
PCA	Principal Component Analysis
TTS	Text-to-Speech
USB	Universal Serial Bus
VNC	Virtual Network Computing
CLAHE	Contrast Limited Adaptive Histogram Equalization
TCP	Transmission Control Protocol
GPU	Graphics Processing Unit
IP	Internet Protocol

CHAPTER 1 INTRODUCTION

1.1 Overview

In today's fast-paced and security-conscious world, traditional door-locking mechanisms are increasingly being complemented or replaced by advanced technologies that provide enhanced safety and convenience. The reliance on physical keys often presents challenges such as misplacement, duplication risks, and unauthorised access, which can compromise security. Similarly, traditional keypad-based systems are prone to issues like forgotten passcodes and tampering. As society gravitates toward smarter and more personalised solutions, facial recognition technology has emerged as a robust and sophisticated method for secure access control. This innovation leverages the unique biological features of individuals to offer a highly reliable and user-friendly alternative to conventional locks.

Facial recognition technology has seen remarkable growth in recent years, driven by advancements in artificial intelligence and the increasing demand for smart security systems. Global markets have embraced this technology, with applications spanning diverse sectors, including banking for secure transactions, healthcare for patient identification, and public safety for surveillance and crime prevention. Facial recognition has emerged as a cornerstone for intelligent access control in the smart home ecosystem, enabling seamless user experiences and enhancing security. As smart cities and IoT devices become more prevalent, the integration of facial recognition is expected to grow exponentially, making it a critical component of modern security solutions.

Facial recognition systems integrate computer vision and artificial intelligence (AI) to provide intelligent, automated access control. Computer vision is pivotal in enabling machines to interpret and process visual data, such as detecting and identifying faces in real-time. Computer vision algorithms form the foundation of facial recognition systems by extracting key facial features and comparing them to stored profiles. Meanwhile, AI facilitates learning and decision-making within these

systems, allowing them to improve their accuracy over time, adapt to environmental changes, and even respond to users dynamically. Together, these technologies create an access control solution that is secure, adaptive, and intuitive.

This project addresses real-world challenges by designing a Raspberry Pi-based face recognition door lock system. By combining affordability, scalability, and advanced technological capabilities, the project seeks to deliver a solution that is both accessible and robust. The system operates using the Raspberry Pi as its core processing unit, and a camera module for real-time face detection and recognition. A solenoid lock mechanism ensures secure physical access, and software-based optimisations enhance the entire setup. The project incorporates advanced features to address potential limitations and extend the system's usability in diverse scenarios.

One of the standout features of the system is adaptive face recognition, which allows the system to learn and refine its understanding of authorised faces over time. This capability eliminates the need to maintain large, static datasets, making the system efficient and scalable. Adaptive recognition ensures that the system evolves alongside its users, capturing variations in appearance, such as aging or slight changes in facial features. Continuous learning makes the system highly accurate and relevant without requiring manual updates.

Adaptive face recognition works by continuously updating the dataset of facial features to improve accuracy and relevance. The system captures new variations of authorised users' faces, such as changes in appearance due to aging, hairstyles, or accessories, and integrates them into the existing dataset. This dynamic approach eliminates the need for manual updates or retraining of the recognition model. Additionally, advanced preprocessing techniques, such as normalisation and feature extraction, enable the system to perform consistently under varying environmental conditions, including lighting changes, distance variations, and background noise. The system ensures high reliability and scalability by leveraging real-time data and intelligent algorithms.

Another key feature is emotion detection, which adds a layer of interaction by recognising user's emotional states. By using advanced AI models to detect emotions such as happiness or anger, the system creates a more personalised user experience. This functionality not only enhances the adaptability but also demonstrates its ability to extend beyond security into intelligent interaction, making it suitable for various applications beyond home use, such as workplaces or public buildings.

The system is designed to perform reliably under varying distances and scaling, ensuring accurate face recognition regardless of how far a user stands from the camera. This capability is essential for real-world use, where users may approach the system from different angles or distances. Coupled with lighting condition adaptation, the system can handle challenging scenarios such as low-light conditions, bright sunlight, or shadows. Advanced image preprocessing techniques ensure consistent performance across diverse environmental settings, maintaining accuracy and reliability.

Another critical feature is the ability to manage multiple authorised faces without requiring retraining of the recognition model. This allows the system to handle multiple users effortlessly, making it ideal for shared environments such as homes, offices, or schools. The system ensures flexibility and scalability by supporting multiple profiles while maintaining its simplicity and efficiency.

Lighting condition adaptation is a critical enhancement designed to ensure the system's accuracy under varying illumination levels. Using advanced preprocessing techniques, the system normalises image brightness and contrast, reduces shadows, and enhances features in low-light or overly bright environments. These adjustments allow the system to detect and recognise faces reliably, whether used in dim hallways, brightly lit rooms, or outdoor settings with inconsistent lighting. This feature ensures that the system remains functional and dependable across diverse scenarios, addressing a common challenge many vision-based systems face.

The system also incorporates real-time notifications, offering seamless communication with connected devices like laptops without requiring a dedicated

app. This feature enables the Raspberry Pi to send notifications to a laptop when a recognised or unrecognised face is detected. The laptop delivers voice feedback using text-to-speech for authorised faces, indicating access has been granted. Conversely, for unknown faces, a notification is sent to alert the user, along with audible feedback signalling the detection of an unrecognised individual. This approach leverages network connectivity to extend the system's functionality, ensuring users remain informed of access events even remotely. By eliminating the need for app installation, the feature enhances convenience while maintaining simplicity and efficiency in operation.

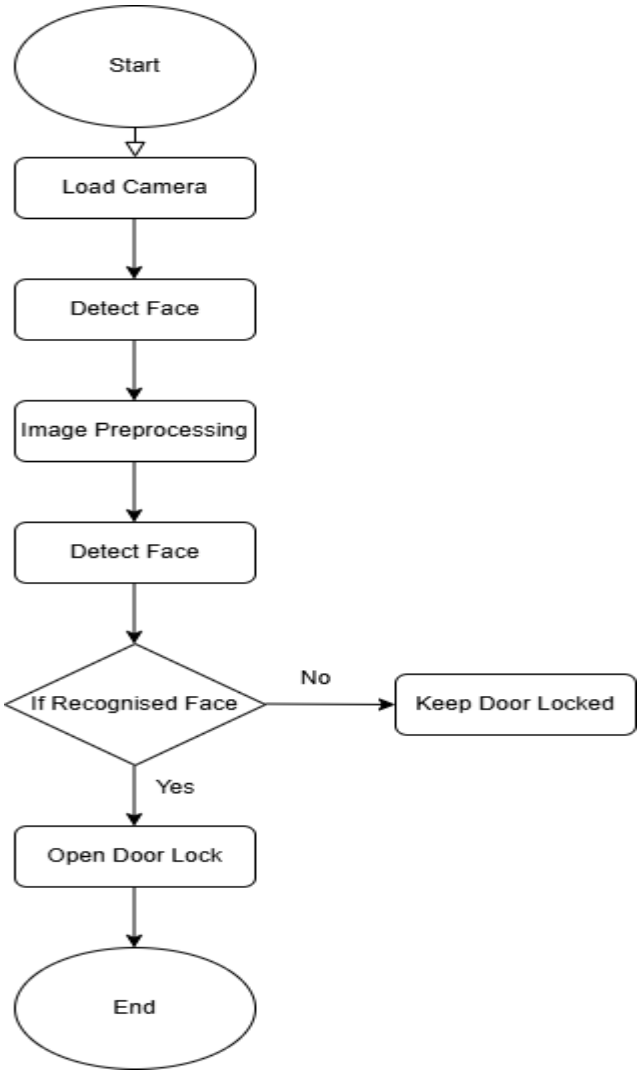

Figure 1.1: Flowchart of the system

The flowchart of the system's operation in figure 1.1 illustrates its step-by-step functionality, starting camera initialization to facial recognition and final decision-making regarding door lock status. The system first captures real-time video input and preprocesses the images to detect faces. Once a face is detected, it compares the features with stored profiles to determine if the individual is authorised. If recognised, the solenoid lock is triggered to grant access, otherwise, the door remains securely locked. This logical and efficient flow ensures a seamless and secure experience for users.

Table 1.1: Comparison between traditional locks and the facial recognition system

Feature	Traditional Locks	Facial Recognition Systems
Security Level	Relies on physical keys, which can be stolen, lost, or duplicated.	Offers higher security through biometric authentication, reducing risk of duplication or theft.
Convenience	Requires carrying keys or remembering combinations.	Hands-free operation; users only need their face for access.
Access Management	Difficult to grant or revoke access (e.g., requires physical keys).	Easy to manage; access can be granted or revoked digitally.
Authentication Time	Relatively slow, especially if keys are misplaced or combinations forgotten.	Fast and seamless; it typically takes a few seconds for recognition.
Cost	Generally lower initial cost, but may require ongoing maintenance (e.g., replacing locks or keys).	Higher initial cost due to technology, but minimal maintenance required.
Scalability	Limited; difficult to manage multiple users efficiently.	Highly scalable; supports multiple authorised users without additional hardware.
Vulnerability	Susceptible to lock picking, key duplication, and physical tampering.	Vulnerable to spoofing attempts but can include anti-spoofing mechanisms for added security.
Environmental Adaptability	No impact from lighting or weather conditions.	May face challenges with lighting, distance, or environmental conditions but can be mitigated with advanced software.

Durability	Mechanical parts can wear out over time.	Electronic components are durable but may require occasional updates or recalibration.
Integration with Other Systems	Limited; typically, standalone systems.	Easily integrates with smart home systems and IoT devices for enhanced functionality.
Notification Capability	None; does not inform users of unauthorised access attempts.	Can send real-time notifications for access attempts or failures.
User Identification	No user-specific identification	Identifies specific users and can log access for each individual.

Finally, table 1.1 shows a comparison between traditional locks and the facial recognition system highlights the advantages of the proposed solution. While traditional locks rely on physical components like keys or pin codes, facial recognition eliminates these vulnerabilities using biometrics. It offers greater security, ease of use, and adaptability, making it a superior choice for modern access control needs.

By implementing these features, the project demonstrates the practical applications of facial recognition and highlights the potential for further innovation. Integrating enhancements like real-time notifications and lighting condition adaptation ensures its robustness across a wide range of scenarios. Furthermore, the system's modularity allows for future upgrades, such as adding a mobile application for remote monitoring or expanding its use cases beyond door locks to other security-sensitive environments.

The system offers tailored benefits to specific user groups, making it a versatile solution for various applications. It provides homeowners a secure, keyless entry mechanism that eliminates the risk of lost or duplicated keys. Office managers can easily manage access for multiple employees, granting or revoking permissions digitally without requiring physical changes to locks. In educational institutions, the system simplifies access control for staff and students while maintaining a high level

of security. Its ability to log access events also provides valuable insights, making it ideal for shared environments where accountability and efficiency are paramount.

In conclusion, the facial recognition door lock system powered by a Raspberry Pi is a major advancement in access control technology. By combining computer vision, artificial intelligence, and cutting-edge features, the project offers a safe, effective, and intuitive solution to conventional security problems. The addition of advanced enhancements and adaptive learning capabilities ensures that the system meets current security needs and is well-equipped to evolve alongside emerging requirements. This work demonstrates how accessible tools and technologies can be combined to create impactful, real-world applications, paving the way for future advancements in smart security solutions.

1.2 Problem Statements

In today's interconnected and fast-paced world, ensuring secure and efficient access control has become more critical than ever. Traditional locking mechanisms, the cornerstone of security for decades, face several inherent challenges that compromise their effectiveness in modern contexts. One of the primary concerns is the risk of unauthorized access. Despite their ubiquity, physical keys are prone to duplication, theft, and tampering. Lock-picking techniques further exacerbate this vulnerability, exposing homes, workplaces, and other secure facility to potential breaches. The consequences of such compromises can range from theft and property damage to severe privacy violations, making it imperative to seek more robust solutions.

The inconvenience associated with physical keys also contributes to their inefficiency. Misplacing or forgetting keys is a common problem that disrupts daily routines and introduces unnecessary delays. Managing multiple sets of keys becomes an operational burden for organisations and shared spaces, often leading to confusion or mismanagement. While addressing some of these issues, traditional keypad-based systems bring their own set of challenges, such as forgotten passcodes or susceptibility to tampering and hacking.

Another significant limitation of traditional systems lies in their inability to adapt to dynamic environments. Granting or revoking access permissions requires manual intervention, such as rekeying locks or redistributing keys. This process is not only time-intensive but also impractical in contexts where access needs frequently change, such as in offices with high employee turnover, educational institutions, or multi-tenant buildings. Furthermore, traditional systems offer no real-time monitoring or alerts mechanism, leaving users unaware of unauthorized access attempts until it is too late. This lack of visibility creates a critical gap in the security framework.

While biometric solutions, such as facial recognition systems, have emerged as a modern alternative to address these challenges, they have limitations. Environmental factors, such as varying lighting conditions, distances, and facial obstructions, can significantly impact the accuracy of these systems. Inconsistent lighting, for example, can obscure facial features, leading to false negatives or recognition failures. These vulnerabilities undermine the reliability and trustworthiness of biometric authentication, limiting its adoption in high-security scenarios.

Scalability presents another challenge for both traditional and biometric systems. In large-scale deployments, such as office buildings, campuses, or multi-location setups, updating access permissions or managing multiple users becomes increasingly complex. Existing solutions often lack the flexibility to handle these demands efficiently, requiring significant time and effort to implement changes.

This project aims to address these challenges by developing a Raspberry Pi-based facial recognition door lock system that combines affordability, scalability, and advanced technology. By leveraging the processing power of the Raspberry Pi, the system integrates real-time facial recognition with adaptive learning capabilities, enabling it to refine its accuracy over time. The project incorporates robust preprocessing techniques to mitigate environmental factors, such as poor lighting or variable distances, ensuring consistent performance across diverse scenarios.

In addition to its technical capabilities, the system is designed to offer significant operational advantages. Real-time notifications provide immediate alerts for both

authorised and unauthorised access attempts, ensuring users remain informed and in control. The system's ability to manage multiple users without retraining further simplifies access management, making it ideal for shared environments such as offices, schools, and multi-tenant buildings. This project delivers a comprehensive, efficient, and reliable solution for modern access management needs by addressing the limitations of traditional locks and existing biometric systems.

The proposed system also establishes the groundwork for upcoming developments in access control technology. Potential enhancements include integrating the system with smart home ecosystems; and enabling seamless interaction with devices such as security cameras, lighting systems, and voice assistants. Additionally, mobile app integration could provide users with remote monitoring and control capabilities, further enhancing convenience and usability. These forward-looking features ensure the system remains relevant and adaptable to emerging requirements, making it a robust solution for current and future security challenges.

1.3 Project Scope

This project aims to develop a Raspberry Pi-based facial recognition door lock system that provides a scalable, secure, and efficient access management solution. Several activities were undertaken to achieve this objective, beginning with the hardware implementation. The setup involves integrating a Raspberry Pi, camera module, relay module, and solenoid lock to create a functional prototype capable of real-time operation. Complementing the hardware, the software development phase introduced advanced features such as adaptive face recognition, emotion detection, and preprocessing techniques for lighting condition adaptation. These features ensure that the system operates reliably in diverse scenarios while maintaining high accuracy and efficiency.

Extensive testing and validation were conducted to evaluate the system's performance under various conditions, including changes in lighting, user distance, and facial orientations. This rigorous testing process verified the system's ability to

process frames in real time, detect faces accurately, and provide immediate feedback. Additionally, the project incorporated system optimisation techniques, such as frame skipping and GPU acceleration, to enhance responsiveness and reduce latency.

However, the scope of this project does not include certain activities. For instance, while the system is designed for compatibility with smart home ecosystems, integration with platforms like Alexa or Google Home is outside the current implementation. Similarly, developing a dedicated mobile application for remote management and monitoring has not been undertaken, although it remains a potential future enhancement. Furthermore, large-scale deployment testing, such as managing multiple doors or user groups across a campus or office building, is beyond the immediate scope of this project.

Despite its robust functionality, the system is subject to certain limitations. The reliance on the Raspberry Pi's computational capabilities imposes constraints on processing power, particularly when handling complex algorithms or multiple simultaneous tasks. While preprocessing techniques effectively address environmental challenges like varying lighting and distances, extreme conditions—such as direct sunlight or heavy shadows—may still impact accuracy. Lastly, scalability remains limited to single-door use, with minimal testing in multi-door or multi-user environments.

In conclusion, the project successfully demonstrates the potential of facial recognition technology in access control, despite its defined scope and inherent limitations. The foundational work established here provides significant opportunities for future enhancements, including advanced anti-spoofing techniques, mobile app integration, and broader scalability for large-scale deployments. These developments will further cement the system's value as a reliable and modern solution for access management.

1.4 Report Outline

This report is organized to give readers a thorough overview of the project, guiding them logically and effectively through its development, methods, and outcomes. The initial sections introduce the project, outlining its motivation, objectives, and the challenges it seeks to address. A detailed explanation of the project's scope and limitations is also provided, setting the context for the work undertaken.

Following this, the report delves into an exploration of existing technologies and related works. By prior research and implementations, this section highlights the gaps and opportunities that the project aims to address, establishing the foundation for its contributions to the access control field.

The implementation section presents a thorough description of the hardware and software components of the system. It covers the design and integration of key components, such as the Raspberry Pi, camera module, relay, and solenoid lock, and explains how they work together to achieve the desired functionality. The discussion also includes an overview of the software algorithms, preprocessing techniques, and optimisations implemented to ensure accuracy and reliability in real-world scenarios.

The results section focuses on the project's outcomes, presenting performance metrics and insights from rigorous testing under diverse conditions. These results are analysed and discussed, providing an evaluation of the system's strengths and limitations, supported by quantitative data and observations.

Finally, the report concludes with a summary of the project's achievements, emphasising its contributions to the field of modern security systems. It also identifies areas for future enhancement, offering further research and development directions to ensure the system remains adaptable and impactful in addressing emerging security challenges.

This organisation ensures a seamless flow of ideas, allowing readers to gain a deep understanding of the project and its implications, from its conception to its realisation and beyond.

1.5 Conclusion

Chapter 1 has outlined the motivation, objectives, and scope of the Raspberry Pi-based face recognition door lock system. The chapter emphasized the limitations of traditional access control methods and introduced the innovative use of facial recognition technology to overcome these challenges. By leveraging artificial intelligence and computer vision, the project aims to deliver a robust, scalable, and user-friendly solution.

The key contributions of this project include the integration of advanced features such as adaptive face recognition, emotion detection, and real-time notifications. These features enhance both the security and convenience of access control systems, making them suitable for a variety of environments, from homes to workplaces.

In the subsequent chapters, the theoretical foundations and implementation details will be explored in depth, beginning with a critical review of relevant literature to provide the necessary context for the project's design and execution.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

Facial recognition technology has emerged as a transformative tool in modern security systems, addressing critical concerns in both industrial and domestic settings. The ability to recognize people by their distinctive facial features offers a non-intrusive, user-friendly, and reliable alternative to traditional access control methods, such as physical keys or PIN codes. As society increasingly embraces automation and smart technologies, facial recognition systems have widespread applications in diverse fields, including home security, public safety, banking, and healthcare. The growing reliance on this technology underscores the need for comprehensive research to enhance its robustness, scalability, and adaptability.

The focus of this review is to critically evaluate the existing body of research on facial recognition systems, particularly in the context of low-cost implementations using Raspberry Pi. This review examines various academic and industrial works spanning various approaches, such as machine learning models, edge computing, and preprocessing techniques. By exploring these methodologies, the review provides a foundation for understanding the challenges and opportunities in developing an effective facial recognition-based door lock system.

Specifically, this review investigates solutions proposed in the past decade that address environmental adaptability, computational efficiency, and integration with hardware components like solenoid locks. Furthermore, it highlights industrial applications, such as security systems employing advanced facial recognition techniques, and identifies their capabilities and limitations. Products and systems designed for smart home environments are examined, focusing on their alignment with industrial needs and the practical challenges they aim to solve.

The primary scope of this review includes recent advancements in facial recognition algorithms, including Convolutional Neural Networks (CNN), Histograms of

Oriented Gradients (HOG), and Principal Component Analysis (PCA). Additionally, it explores the integration of facial recognition with edge computing and Internet of Things (IoT) devices, which are essential for attaining scalable and reasonably priced applications. By critically analysing these approaches, the review identifies gaps in existing literature, such as the need for enhanced lighting adaptability, real-time notifications, and adaptive learning capabilities.

This review also evaluates the challenges faced by current systems, such as their vulnerability to environmental changes, computational overhead, and limited scalability. The findings establish a framework for developing the proposed Raspberry Pi-based facial recognition door lock system, which aims to address these shortcomings. The discussion ultimately positions the project within the broader landscape of academic and industrial innovations, underscoring its significance in advancing secure, adaptable, and efficient access control solutions.

2.2 Literature Review Section

2.2.1 A Face Recognition Method in Machine Learning for Enhancing Security in Smart Home

The paper titled "A Face Recognition Method in Machine Learning for Enhancing Security in Smart Home" [1] looks into how advanced algorithms for machine learning can be used for facial recognition in home security. To guarantee excellent face identification and recognition accuracy, the authors emphasize the usage of Convolutional Neural Networks (CNN), such as Mobilenet and Alexnet. This work shows how deep learning models may offer reliable biometric verification in smart home settings, which is a major breakthrough in the industry.

A major strength of this paper lies in its methodology, which leverages CNNs to extract hierarchical facial features, enabling accurate recognition even under challenging conditions. The authors discuss how the network's layered architecture contributes to identifying unique patterns in facial images, enhancing the system's

reliability. Furthermore, integrating automated processes reduces the need for manual intervention, making it suitable for modern security applications.

Despite its strengths, the paper identifies several limitations that restrict its practicality. First, the computational complexity of CNNs requires substantial processing power, making it less feasible for real-time applications on resource-constrained devices like Raspberry Pi. Second, privacy concerns emerge as a critical issue, particularly regarding storing and handling sensitive biometric data. The authors suggest potential solutions, to mitigate these risks, such as on-device processing and data encryption.

The study aligns closely with existing trends in AI-based security systems, offering a comprehensive framework for implementing facial recognition in smart homes. However, it primarily focuses on controlled environments with stable lighting and consistent user behaviour, leaving room for improvement in adaptability and scalability. The authors acknowledge the need for further research to address these gaps, particularly in scenarios involving varying lighting conditions, facial orientations, and dynamic user interactions.

This paper provides a valuable foundation for the proposed project, which aims to overcome the limitations highlighted in [1] The project balances computational efficiency with accuracy by adopting a lightweight recognition architecture optimized optimised for Raspberry Pi. Additionally, the integration of preprocessing techniques, such as lighting normalisation and gamma correction, addresses the environmental adaptability concerns raised in [1]. The project also incorporates adaptive learning capabilities, enabling the system to evolve without manual updates, enhancing its usability and long-term effectiveness.

In conclusion, while the work in [1] demonstrates the potential of CNN-based facial recognition for enhancing home security, its applicability is limited by high computational demands and privacy risks. The proposed project builds on these findings by delivering a scalable, efficient, and adaptable solution, addressing the gaps identified in the literature.

2.2.2 Raspberry Pi-powered Door Lock with Facial Recognition

The paper titled "Raspberry Pi-powered Door Lock with Facial Recognition" [2] explores implementing a cost-effective facial recognition system designed for access control. Utilising the Haar Cascade classifier for face detection, the system integrates seamlessly with a Raspberry Pi and a solenoid lock, presenting a viable solution for home security. The authors emphasize the design's affordability and simplicity, making it accessible to users with minimal technical expertise. This approach highlights the potential of using low-cost hardware and lightweight algorithms for real-world applications.

A notable strength of this paper is its focus on practicality. As a detection algorithm, Haar Cascade is computationally lightweight and well-suited for resource-constrained devices like the Raspberry Pi. The integration of hardware components, such as the solenoid lock, demonstrates a comprehensive understanding of the needs of a security system, ensuring that both software and hardware aspects work cohesively. Additionally, the paper provides a detailed account of the system's deployment, making it a valuable reference for researchers and practitioners aiming to replicate similar setups.

However, the paper also identifies significant limitations. The Haar Cascade classifier, while efficient, struggles with low-light conditions, varying facial orientations, and detecting faces at greater distances. These constraints limit the system's reliability in diverse environments, which is critical for real-world use. Moreover, the static nature of the face dataset requires manual updates to include new users, reducing the system's scalability and adaptability. The lack of advanced preprocessing techniques further impacts its performance under challenging conditions.

The proposed project addresses these limitations by incorporating adaptive recognition and advanced preprocessing techniques, such as lighting normalisation and gamma correction, to enhance reliability under varying environmental conditions. Additionally, the use of real-time learning allows the system to dynamically adapt to

new users without requiring manual updates, significantly improving scalability. The integration of emotion detection and real-time notifications extends the system's functionality beyond basic access control, demonstrating its potential for broader applications.

In conclusion, while [2] provides a foundational approach to developing a low-cost facial recognition-based door lock system, it falls short in handling real-world challenges such as environmental adaptability and scalability. The current project builds on these findings by introducing innovative features and optimizations that enhance performance and usability, ensuring a more robust and versatile solution for modern access control systems.

2.2.3 Face Recognition Door Lock System Using Raspberry Pi

The paper titled "Face Recognition Door Lock System Using Raspberry Pi," published in the *International Journal of Research in Advent Technology (IJRASET)* [3], demonstrates an access control system based on facial recognition that uses the Raspberry Pi as the central processor unit and Haar Cascade classifiers. The authors emphasise the system's simplicity and effectiveness, aiming to provide a secure and user-friendly method for managing access in domestic environments. The study incorporates OpenCV for image processing and highlights the use of a solenoid lock for door control.

One of the key strengths of this work is its focus on the practical implementation of facial recognition within a cost-effective framework. By leveraging the Raspberry Pi and the Haar Cascade classifier, the authors demonstrate how low-budget hardware and lightweight algorithms can be combined to create a functional security system. The integration of OpenCV enables efficient image processing, while the solenoid lock ensures secure physical access.

However, the study identifies several limitations that impact the system's real-world applicability. The reliance on Haar Cascade classifiers restricts the system's accuracy,

particularly in scenarios involving poor lighting or unusual facial angles. Moreover, the paper highlights a dependency on stable internet connectivity for certain features, which limits the system's usability in environments with intermittent network access. Another significant drawback is the system's inability to adapt dynamically to new users, requiring manual updates to the face database for scalability.

The proposed project builds on the foundation laid by [3] by addressing these critical limitations. Adaptive recognition techniques are introduced to enhance the system's ability to dynamically learn and update user profiles, eliminating the need for manual interventions. Advanced preprocessing methods, including lighting normalisation and gamma correction, are employed to improve performance under varying environmental conditions. Additionally, including real-time notifications and emotion detection extends the system's functionality, making it suitable for a broader range of applications.

In conclusion, the paper [3] provides valuable insights into developing a low-cost, facial recognition-based security system but falls short in addressing challenges related to environmental adaptability and scalability. The current project enhances these aspects by integrating adaptive features, robust preprocessing techniques, and additional functionalities, offering a more comprehensive and reliable solution for modern access control systems.

2.2.4 Home Security System Based on Facial Recognition

The paper titled "Home Security System Based on Facial Recognition" [4] explores the application of facial recognition technology using the Histograms of Oriented Gradients (HOG) algorithm. The study integrates this algorithm with Internet of Things (IoT) capabilities to provide a comprehensive security solution that includes remote monitoring and access control. The authors emphasise the benefits of using HOG, a lightweight and efficient method for face detection, and its compatibility with resource-constrained hardware like Raspberry Pi.

A key strength of this paper is its focus on IoT integration, which enhances the system's functionality by enabling real-time alerts and remote-control capabilities. The use of HOG for face detection ensures computational efficiency, making the system suitable for low-power devices. Additionally, the paper highlights the simplicity of the setup, which is designed to be user-friendly and cost-effective.

Despite its strengths, the study identifies several limitations that affect the system's performance and reliability. While efficient, reliance on HOG presents challenges in scenarios with poor lighting or complex environmental conditions. The system's recognition accuracy decreases significantly in low-light environments or when faces are partially obscured. Furthermore, the authors note that the system's face dataset is static, requiring manual updates to add or remove users, which limits scalability and adaptability in dynamic settings.

The proposed project builds on the findings in [4] by addressing these challenges through advanced preprocessing techniques, such as lighting normalisation and gamma correction, to enhance recognition accuracy under varying environmental conditions. Adaptive learning capabilities are introduced to allow the system to dynamically update its database of authorised users, eliminating the need for manual intervention. Moreover, the project simplifies IoT integration by enabling real-time notifications without requiring dedicated apps, ensuring seamless communication with connected devices.

In conclusion, while [4] provides a solid foundation for the use of facial recognition and IoT in home security systems, it falls short in handling real-world challenges such as environmental adaptability and dynamic user management. The current project overcomes these limitations by incorporating robust preprocessing techniques, adaptive learning, and enhanced IoT integration, offering a more reliable and versatile solution for modern security needs.

2.2.5 Face Recognition Door Lock System Using Raspberry Pi (PCA)

The paper titled "Face Recognition Door Lock System Using Raspberry Pi," presented at the *Asia-Pacific Conference on Smart Home Technology* [5], explores the use of Principal Component Analysis (PCA) for facial recognition. PCA is employed to simplify the dataset by reducing its dimensionality, which significantly reduces computational overhead. The system integrates Raspberry Pi as the primary processing unit and includes a solenoid lock for physical access control, aiming to deliver a cost-effective and lightweight security solution.

The paper's key strength lies in its use of PCA, which enables efficient handling of facial data by extracting the most significant features from images. This approach minimises processing time and resource requirements, making it suitable for low-power devices like Raspberry Pi. The authors also highlight the affordability and simplicity of the system, making it accessible to users with limited technical expertise. The detailed implementation methodology also provides a clear framework for developing similar systems.

However, the study identifies notable limitations restricting its effectiveness in real-world scenarios. While computationally efficient, PCA tends to struggle with recognition accuracy in challenging environments, such as those with poor lighting, varying facial angles, or distant faces. The system's reliance on a static face database limits its scalability, as adding new users requires manual updates. The authors acknowledge these challenges but do not propose concrete solutions, leaving a gap in the system's adaptability and usability.

The proposed project addresses these limitations by integrating advanced preprocessing techniques, such as lighting normalisation and gamma correction, to enhance the system's performance under diverse environmental conditions. Adaptive recognition capabilities are incorporated to allow the system to dynamically update its user database, eliminating the need for manual interventions. Moreover, including real-time notifications and emotion detection extends the system's functionality, making it suitable for a wider range of applications beyond basic access control.

In conclusion, while [5] provides a valuable framework for implementing cost-effective facial recognition systems, its reliance on PCA limits its adaptability and accuracy in complex environments. The current project builds on these findings by introducing innovative features and enhancements that address the identified gaps, resulting in a more robust and scalable solution for modern access control needs.

2.3 Comparison of Results

The reviewed papers collectively provide a comprehensive overview of facial recognition systems, particularly those implemented on low-cost hardware platforms like Raspberry Pi. Each paper approaches the topic from a unique perspective, contributing valuable insights into algorithm efficiency, adaptability, and practical implementation challenges. Table 2.1 summarises the findings of the literature review for the five key papers focused on.

Table 2.1: Literature Review Table of Findings

Paper Title	Year	Idea	Technology	Application	Advantage	Findings
A Face Recognition Method in Machine Learning for Enhancing Security in Smart Home	2022	Use of CNNs (e.g., Mobilenet, Alexnet) for facial recognition in smart homes	CNN-based Machine Learning	Smart home security	High accuracy and automation	Effective for controlled environments, but high computational demand limits real-time use.
Raspberry Pi-powered Door Lock with Facial Recognition	2020	Low-cost facial recognition system using Haar	Haar Cascade, OpenCV, Raspberry Pi	Home security	Affordable and simple design	Struggles with low-light conditions and scaling; limited adaptability to diverse scenarios.

		Cascade classifier				
Face Recognition Door Lock System Using Raspberry Pi	2024	Implementation of Haar Cascade with Raspberry Pi for door access control	Haar Cascade, Raspberry Pi	Home and office access control	Lightweight and cost-effective	Reliance on internet and static face dataset restricts scalability and adaptability.
Home Security System Based on Facial Recognition	2022	IoT-enabled home security system utilising HOG algorithm for face recognition	HOG, IoT, Raspberry Pi	Home security	IoT integration for remote monitoring	Poor performance under low-light conditions and limited scalability due to static datasets.
Face Recognition Door Lock System Using Raspberry Pi (PCA)	2022	Use of PCA for dimensionality reduction in facial recognition	PCA, Raspberry Pi	Low-cost facial recognition	Computational efficiency for small datasets	Inaccurate under low-light and complex environments; lacks adaptability and dynamic updates.

The paper "A Face Recognition Method in Machine Learning for Enhancing Security in Smart Home" highlights the strengths of CNNs in achieving high accuracy in controlled environments. However, it notes the computational complexity of such models, making them impractical for resource-constrained devices like Raspberry Pi.

"Raspberry Pi-powered Door Lock with Facial Recognition" and "Face Recognition Door Lock System Using Raspberry Pi" focus on cost-effective solutions using Haar Cascade. These papers demonstrate the feasibility of low-cost implementations but

fail to address critical challenges such as adaptability to environmental changes and dynamic user management.

The study "Home Security System Based on Facial Recognition" integrates IoT capabilities with the HOG algorithm, offering remote monitoring features. While this approach adds value through IoT integration, the system struggles with low-light performance and static datasets, limiting its reliability and scalability.

Finally, "Face Recognition Door Lock System Using Raspberry Pi (PCA)" emphasises computational efficiency through dimensionality reduction techniques. While PCA proves effective for managing small datasets, its accuracy diminishes in complex real-world environments. The lack of adaptability to new users further restricts its practicality.

The proposed project builds on the findings of these papers by addressing their limitations. It introduces adaptive recognition for dynamic user management, advanced preprocessing techniques to improve performance under varying environmental conditions, and real-time notifications without requiring app dependency. These enhancements create a scalable, reliable, and versatile solution, bridging the gaps identified in the literature.

2.4 Identifying Research Gaps and Future Steps

The literature review highlights several key research gaps in existing facial recognition systems, particularly those designed for low-cost Raspberry Pi implementations. A significant challenge identified is the difficulty of achieving consistent recognition performance under varying environmental conditions, such as poor lighting and diverse facial orientations. While advanced algorithms like CNNs offer high accuracy, their computational demands limit their applicability to resource-constrained devices. Conversely, lightweight approaches, like Haar Cascade and PCA, struggle with adaptability and scalability in real-world scenarios.

Another gap is the lack of systems capable of dynamically managing user databases. Most existing solutions rely on static datasets that require manual updates to add or remove users. This limitation reduces their scalability and usability in dynamic environments, such as homes or workplaces with frequent changes in access requirements. Furthermore, the absence of advanced preprocessing techniques, such as lighting normalisation and gamma correction, further restricts the reliability of these systems.

The literature also points to the underutilisation of real-time notifications and emotion detection in current systems. While IoT integration has been explored, few implementations effectively leverage these capabilities to enhance user interaction and security monitoring. The inability to add new faces in real-time, as observed in many systems, is another area that requires attention.

This project aims to address these gaps through several innovative steps. First, adaptive recognition techniques will enable the system to learn and update its database dynamically, ensuring scalability and adaptability. Second, advanced preprocessing methods, including lighting normalisation and gamma correction, will be integrated to enhance recognition accuracy under challenging conditions. Third, the system will incorporate a feature that allows unknown faces to be authorised directly by pressing a button, simplifying the user management process. Lastly, real-time notifications and emotion detection will be implemented to extend the system's functionality, providing users with a more interactive and secure experience.

Future steps include exploring energy-efficient hardware configurations and advanced anti-spoofing mechanisms to enhance the system's robustness. Additionally, integrating mobile applications for remote monitoring and control will be considered, aligning the project with the growing demand for IoT-enabled smart home solutions. These advancements will ensure that the system remains at the forefront of facial recognition technology, addressing current challenges and emerging requirements.

2.5 Conclusion

The Literature Review chapter provided an in-depth analysis of existing research and technologies relevant to facial recognition systems. By examining key studies, it highlighted the evolution of access control solutions, emphasizing the strengths and weaknesses of various approaches. The review critically evaluated methodologies such as HOG-based face detection, emotion detection with AI, and notification systems, identifying gaps like scalability, lighting adaptation, and real-time performance optimization.

These gaps form the basis of this project's innovations, including adaptive learning capabilities, robust preprocessing techniques, and real-time notifications. Furthermore, the literature review established the foundation for the system's design, ensuring the integration of proven concepts while addressing identified shortcomings.

This chapter sets the stage for the next phase of the report, which delves into the design and implementation of the Raspberry Pi-based face recognition door lock system, translating theoretical insights into practical applications.

CHAPTER 3 DETAILS OF THE DESIGN

3.1 Introduction

The implementation of the Raspberry Pi-based facial recognition door lock system involved an intricate combination of hardware assembly, software programming, and rigorous testing to achieve a robust and efficient solution for modern access control challenges. This chapter outlines the comprehensive steps undertaken to realize the project, detailing the integration of hardware components, the development of adaptive facial recognition software, and the testing process to validate system performance under diverse scenarios.

The primary goals of the implementation process were to create a system that is not only secure and reliable but also scalable and user-friendly. This required careful attention to both hardware and software integration, as well as addressing key challenges such as computational limitations of the Raspberry Pi and variability in environmental conditions.

The project's hardware implementation centres around the Raspberry Pi as the primary processing unit, interfaced with a camera module for real-time video capture, a relay module, and a solenoid lock for physical access control. Each hardware component was carefully selected for its compatibility, cost-effectiveness, and suitability for low-power applications. The assembly required precise GPIO connections, ensuring seamless communication between the Raspberry Pi and the peripheral devices.

On the software side, the system leverages Python-based libraries, including OpenCV and DeepFace, for facial recognition and emotion detection. The software design incorporates advanced features such as adaptive learning to dynamically update the authorized face database, preprocessing techniques to handle varying environmental conditions, and real-time notifications to alert users of access events.

The interaction between the hardware and software components was achieved through efficient coding practices, ensuring smooth and responsive system behaviour.

The testing and validation process played a crucial role in ensuring the reliability and scalability of the system. Comprehensive tests were conducted to evaluate the system's performance across different lighting conditions, distances, and facial orientations. Numerical results were collected and analysed to measure detection accuracy, processing times, and emotion detection rates. The findings from these tests informed iterative refinements to enhance the overall functionality and robustness of the system.

In summary, this chapter provides a detailed account of the hardware setup, software design, and testing methodologies employed to implement the Raspberry Pi-based facial recognition door lock system. The following sections delve deeper into each aspect, offering insights into the technical decisions, challenges encountered, and innovative solutions developed throughout the project.

3.2 Hardware Implementation

The hardware implementation of the Raspberry Pi-based facial recognition door lock system plays a critical role in ensuring seamless functionality and reliability. This section provides a detailed account of the hardware setup, focusing on the components used, their specifications, roles, and how they were interconnected to achieve the desired functionality.

The hardware design revolves around the Raspberry Pi Model 3B, which serves as the central processing unit, interfaced with a camera module for real-time video input and a relay module connected to a solenoid lock for physical access control. Supporting components such as the DC barrel jack, microSD card, jumper wires, and power supplies ensure stable connections and reliable operation. Each component was selected for its compatibility with the system requirements and its ability to deliver cost-effective and scalable solutions.

The following subsections detail each component's description, role in the circuit, and specific connections, providing a comprehensive understanding of the hardware implementation. A high-quality circuit diagram illustrates the connections, and a photograph of the assembled hardware setup demonstrates the practical realization of the design.

3.2.1 Raspberry Pi

The Raspberry Pi Model 3B serves as the core processing unit for the project, handling all computational tasks, including facial recognition, emotion detection, and communication with peripheral devices. It features:

- **Specifications**:
    - Processor: Quad-core Cortex-A53 (ARMv8) 64-bit SoC
    - RAM: 1GB
    - GPIO Pins: 40 pins for interfacing with external components
    - Networking: Built-in Wi-Fi and Bluetooth
- **Role in the Circuit**:
    - Executes the facial recognition algorithm and controls the relay module to activate the solenoid lock.
    - Communicates with the camera module to capture video frames.
    - Sends notifications to the connected laptop.

- **Connections**:
    - GPIO 17: Connected to the IN pin of the relay module for lock control.
    - 5V and GND: Provide power to the relay module.
    - CSI Port: Connects to the camera module.

As shown in Figure 3.2.1, the Raspberry Pi 3B is a compact, low-power computing board equipped with multiple USB ports, an HDMI output, and GPIO pins for hardware interfacing, making it ideal for embedded applications such as this system

Figure 3.2.1 :The Raspberry Pi Model 3B

3.2.2 Raspberry Pi Camera Module v2

The camera module captures live video feeds, which are processed by the Raspberry Pi for facial recognition. It is lightweight and optimized for compatibility with the Raspberry Pi.

- **Specifications**:
    - Resolution: 8MP (Model v2)
    - Interface: CSI (Camera Serial Interface)
    - Frame Rate: Up to 30fps
- **Role in the Circuit**:
    - Provides real-time video input for facial recognition and emotion detection.
- **Connections**:
    - CSI Connector: Connected to the Raspberry Pi's CSI port.

The Raspberry Pi Camera Module v2, shown in Figure 3.2.2, captures real-time video for face detection and recognition, ensuring efficient image acquisition.

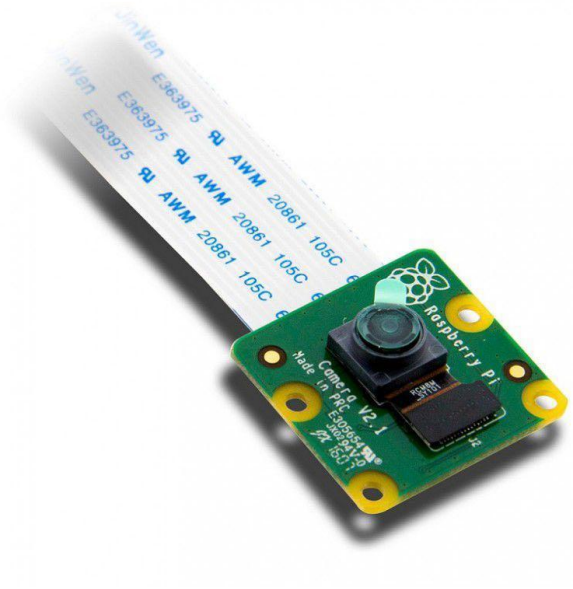

Figure 3.2.2: The Raspberry Pi Camera Module

3.2.3 Relay Module

The relay module acts as an intermediary between the Raspberry Pi and the solenoid lock, enabling the system to control the high-voltage lock circuit safely.

- **Specifications**:
    - Type: Double-channel relay module
    - Operating Voltage: 5V
    - Control Signal: TTL logic (3.3V/5V compatible)
- **Role in the Circuit**:
    - Isolates the low-voltage Raspberry Pi GPIO from the high-voltage lock circuit.

- Switches the solenoid lock on or off based on the Raspberry Pi's signal.

- **Connections**:
  - VCC: Connected to the 5V pin of the Raspberry Pi.
  - GND: Connected to the GND pin of the Raspberry Pi.
  - IN: Connected to GPIO 17 of the Raspberry Pi (single-channel usage).
  - NO and COM: Connected to the solenoid lock and the power supply.

The system uses a double-channel relay module (Figure 3.2.3) to control the solenoid lock, enabling secure door access.

Figure 3.2.3: The double-channel relay module

3.2.4 Solenoid Lock

The solenoid lock provides the physical locking and unlocking mechanism for the door. Controlled by the relay module, it ensures secure access.

- **Specifications**:
    - Operating Voltage: 12V DC
    - Current: 500mA
    - Type: Electromagnetic lock
- **Role in the Circuit**:
    - Locks or unlocks the door based on the signal received from the relay module.
- **Connections**:
    - Positive Terminal: Connected to the NO terminal of the relay module.
    - Negative Terminal: Connected to the negative terminal of the 12V power supply.

A solenoid lock (Figure 3.2.4) is used to physically secure and release the door upon successful authentication.

Figure 3.2.4: The solenoid lock

3.2.5 Power Supplies

Two separate power supplies are used to ensure reliable operation:

- **5V Power Supply**:
    - Powers the Raspberry Pi and connected peripherals like the relay module.
- **12V Power Supply**:
    - Powers the solenoid lock to ensure sufficient electromagnetic force for operation.
- **Role in the Circuit**:
    - Provide stable power for both low-voltage (Raspberry Pi and relay module) and high-voltage (solenoid lock) components.

The Raspberry Pi power adapter (Figure 3.2.5) ensures a stable power supply for uninterrupted system operation.

Figure 3.2.5: Power supply used for the Raspberry Pi

A 12V power adapter (Figure 3.2.6) is used to power the solenoid lock, ensuring reliable operation.

Figure 3.2.6: Power supply used for the solenoid lock

3.2.6 DC Barrel Jack and Jumper Wires

The DC barrel jack ensures a secure connection between the 12V power supply and the solenoid lock. Jumper wires facilitate connections between the Raspberry Pi GPIO pins and other components.

- **Specifications**:
    - Barrel Jack: Standard 5.5mm outer diameter, 2.1mm inner diameter.
    - Jumper Wires: Male-to-Male and Male-to-Female connectors.
- **Role in the Circuit**:
    - DC Barrel Jack: Connects the solenoid lock to the power supply securely.
    - Jumper Wires: Enable flexible and reliable connections between components.

A DC power jack adapter (Figure 3.2.7) is used to connect the solenoid lock's power supply securely.

Figure 3.2.7: DC barrel jack

Jumper wires (Figure 3.2.8) are used to establish electrical connections between the Raspberry Pi and peripheral components.

Figure 3.2.8: Jumper wires

3.2.7 MicroSD Card

The microSD card is essential for storing the Raspberry Pi operating system (e.g., Raspbian) and the project scripts.

- **Specifications**:
    - Storage: 32GB
    - Compatibility: Supports Raspberry Pi bootloader
- **Role in the Circuit**:
    - Houses the operating system and project files necessary for system operation.
- **Connections**:
    - Inserted into the Raspberry Pi's microSD card slot.

A 16GB microSD card (Figure 3.2.9) is used to store the Raspberry Pi operating system and project files.

Figure 3.2.9: The microSD card

3.2.8 Hardware Components and Their Roles

To summarize the hardware setup, Table 3.2.1 lists all the components used in the system, along with their specifications and roles:

Table 3.2.1: Hardware Components, Specifications, and Roles

Component	Specifications	Role in Circuit
Raspberry Pi Model 3B	Quad-core Cortex-A53, 1GB RAM	Central processing unit
Camera Module	8MP resolution, CSI interface	Captures real-time video
Relay Module	Double-channel, 5V control signal	Controls the solenoid lock (one channel used)
Solenoid Lock	12V DC, 500mA	Provides secure physical locking mechanism
5V Power Supply	5V DC	Powers Raspberry Pi and peripherals
12V Power Supply	12V DC	Powers the solenoid lock
DC Barrel Jack	5.5mm outer, 2.1mm inner diameter	Connects 12V power supply to solenoid lock
Jumper Wires	Male-to-Male, Male-to-Female	Connects GPIO pins to other components

MicroSD Card	16GB-64GB, compatible with Pi OS	Stores the OS and project files

The table provides a concise yet comprehensive view of the hardware components utilized in this project, detailing their essential contributions to the overall system functionality.

3.2.9 Circuit Diagram

To illustrate the interconnections of the hardware components, the following simplified circuit diagram (Figure 3.2.10) has been created. This diagram provides a clear representation of how the Raspberry Pi Model 3B interfaces with the relay module, solenoid lock, power supplies, and the camera module.

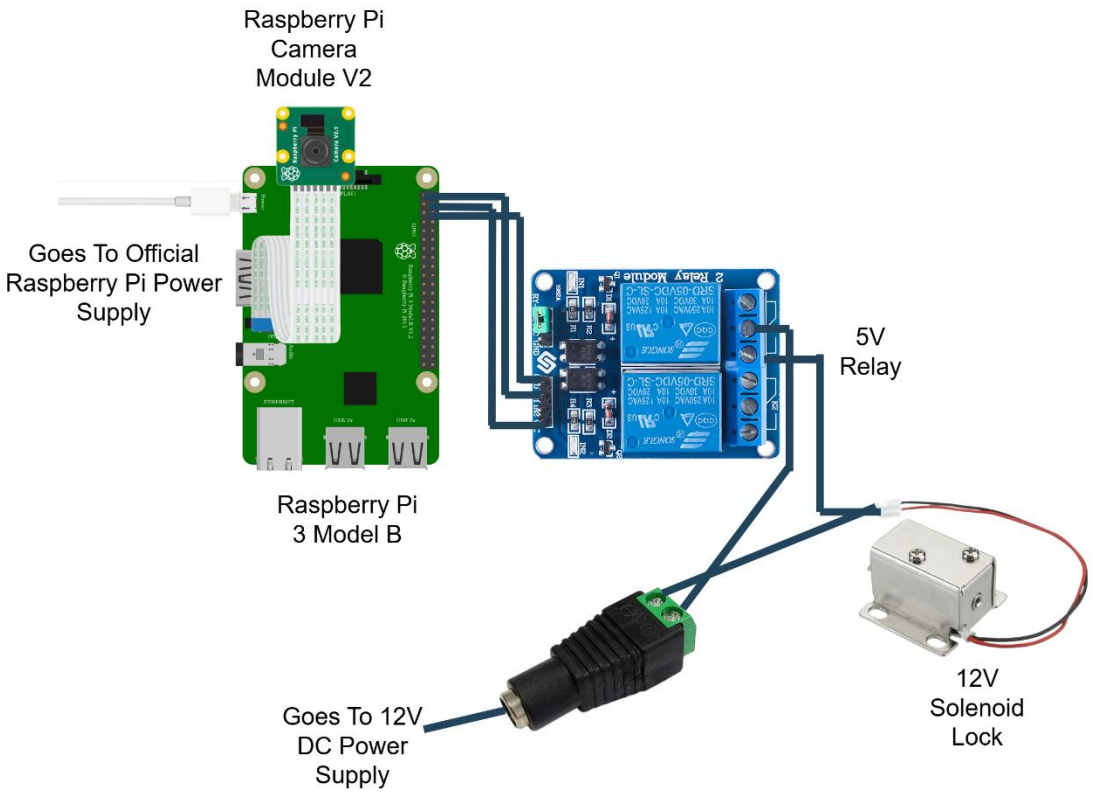

Figure 3.2.10: Circuit diagram illustrating the hardware connections for the face recognition door lock system.

This diagram visually explains the flow of power and control signals within the system, offering clarity on the role and connectivity of each component. It simplifies the understanding of how the Raspberry Pi interacts with the solenoid lock through the relay module while maintaining separate power supplies for different components.

3.2.10 Hardware Assembly

In this section, we outline the step-by-step process of assembling the hardware components of the Raspberry Pi-based face recognition door lock system. The assembly integrates the Raspberry Pi, camera module, relay module, solenoid lock, and other necessary components into a functional setup. The objective is to ensure secure connections, operational reliability, and an organized layout for ease of maintenance.

Step-by-Step Assembly Process

1. **Prepare the Raspberry Pi:**
    - Insert the preloaded microSD card containing the operating system and software into the slot on the Raspberry Pi Model 3B.
    - Connect the Raspberry Pi Camera Module to the CSI (Camera Serial Interface) port. Ensure the metal contacts on the ribbon cable face the CSI port for proper connectivity.
    - Secure the Raspberry Pi on a stable platform or within a protective case to prevent damage during the assembly process.
2. **Connect the Relay Module:**
    - Use jumper wires to connect the relay module to the Raspberry Pi:

        - Connect the VCC pin on the relay to the 5V GPIO pin of the Raspberry Pi.

        - Connect the GND pin on the relay to a GND GPIO pin on the Raspberry Pi.

        - Connect the IN1 pin on the relay to GPIO 17 on the Raspberry Pi.

    - Position the relay module close to the Raspberry Pi for tidy and secure wiring.

3. **Wire the Solenoid Lock:**

    - Connect the positive terminal of the 12V Solenoid Lock to the NO (Normally Open) terminal of the relay.

    - Connect the COM (Common) terminal on the relay to the positive terminal of the 12V DC Power Supply.

    - Attach the negative terminal of the solenoid lock to the negative terminal of the 12V DC power supply.

4. **Integrate the DC Barrel Jack:**

    - Use a DC barrel jack adapter to connect the 12V DC power supply to the solenoid lock and relay module. This ensures a secure and stable power connection.

5. **Establish Stable Connections:**

    - Double-check all jumper wire connections to ensure they are secure and properly aligned with the circuit diagram.

- Inspect the relay connections, solenoid lock, and GPIO pins on the Raspberry Pi to verify there are no loose wires or misconfigurations.

6. **Power Up the Raspberry Pi:**

   - Use the official Raspberry Pi power adapter to power up the Raspberry Pi through its micro-USB port.

   - Confirm that the 12V DC power supply is securely connected to the solenoid lock and relay module.

7. **Test the Circuit:**

   - Run basic GPIO scripts on the Raspberry Pi to test the relay and solenoid lock functionality.

   - Ensure the relay triggers correctly and the solenoid lock responds as intended.

The fully assembled hardware setup, as shown in Figure 3.2.11, integrates the Raspberry Pi, relay module, solenoid lock, and camera module for real-time face recognition and access control.

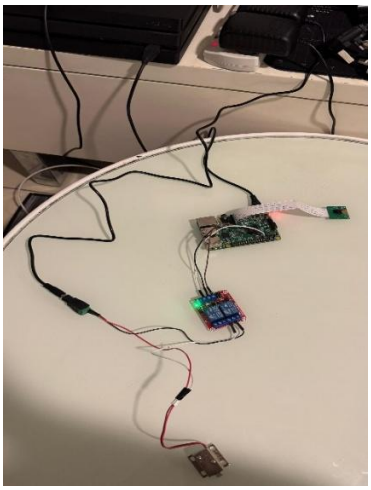

Figure 3.2.11: Photograph of the fully assembled hardware setup

The hardware assembly process for the Raspberry Pi-based face recognition door lock system is critical to its overall performance. By following the step-by-step instructions provided above, the components are securely integrated, ensuring smooth operation and reliability. The final assembled setup is both compact and efficient, designed for easy maintenance and future upgrades.

The hardware implementation of the Raspberry Pi-based face recognition door lock system forms the backbone of the project, integrating multiple components into a cohesive and functional setup. Each component, from the Raspberry Pi Model 3B to the solenoid lock and relay module, plays a vital role in ensuring secure and reliable access control. The carefully designed circuit connections, coupled with the logical assembly process, provide a robust foundation for the system. The inclusion of power supply mechanisms and adaptable configurations further enhances the system's efficiency and scalability. This meticulously constructed hardware platform sets the stage for seamless integration with the software and testing phases, demonstrating the project's commitment to precision and practicality.

3.3 Software Implementation

The software implementation of the Raspberry Pi-based face recognition door lock system involves an intricate combination of algorithms and processes designed for efficiency, accuracy, and adaptability. The development began with a simulation phase, where the feasibility of the face recognition algorithm was tested using a laptop and a pre-trained CNN model. This simulation provided valuable insights into the algorithm's performance, forming the foundation for the subsequent hardware integration.

Building on this groundwork, the system was implemented on the Raspberry Pi to achieve real-time face detection, recognition, and dynamic adaptability. The system detects faces, recognizes authorized individuals, adapts to changing conditions, and

notifies remote devices in real-time. The following subsections outline the implementation of key features, including the facial recognition algorithm, the notification system, emotion detection, and the dynamic addition of unknown faces to the authorized list.

Python was used to create the software components, utilizing libraries like DeepFace, cv2, face_recognition, and others. Reliable performance under a range of environmental situations is ensured by advanced preprocessing techniques including gamma correction and lighting normalization. These components collectively create a robust and efficient system capable of handling diverse use cases, such as recognizing faces at different distances, adapting to dynamic lighting conditions, and processing real-time notifications on remote devices.

The project utilizes several essential Python libraries for face recognition, GPIO control, and notification handling, as shown in Figure 3.3.1

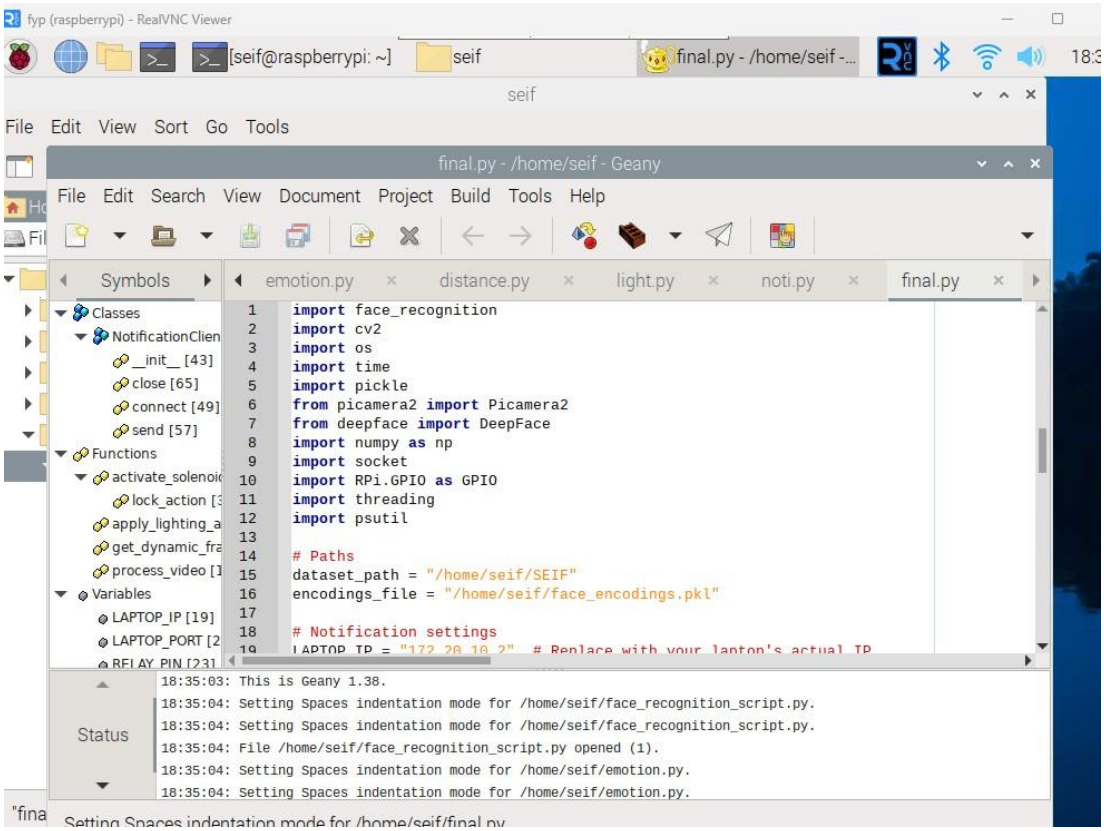

Figure 3.3.1: Libraries used in the project's script

3.3.1 Simulation and Preliminary Testing

As part of FYP1, a simulation of the face recognition system was conducted to test the feasibility of the algorithm and evaluate its real-time processing capabilities. The simulation also aimed to simulate the functionality of the door lock/unlock mechanism based on facial recognition results.

System Setup

The simulation was conducted on a laptop environment using the following components:

1. **Haar Cascade Classifier**: Used for face detection within webcam video streams.

2. **Pre-Trained CNN Model**: Deployed to classify detected faces into two categories: authorized (e.g., “SEIF”) and unauthorized (“NOT SEIF”).

3. **Laptop Webcam**: Served as the video input device for real-time frame processing.

4. **OpenCV and TensorFlow**: Core libraries for video processing and model execution.

The simulation environment for testing face recognition is shown in Figure 3.3.2.

```python
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau, TensorBoard
from tensorflow.keras.regularizers import l2

# Define paths to your data directories
train_data_dir = r'C:\Users\sseno\OneDrive\Documents\FYP\Dataset'
validation_data_dir = r'C:\Users\sseno\OneDrive\Documents\FYP\Datatest'

# Define image dimensions and other constants
img_width, img_height = 224, 224
batch_size = 16
epochs = 30

# Data Augmentation
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.3,
    zoom_range=0.3,
    rotation_range=40,
    width_shift_range=0.3,
    height_shift_range=0.3,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2],
    channel_shift_range=0.2,
    fill_mode='nearest'
)

test_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='binary'
```

Figure 3.3.2: Simulation Setup for Face Recognition

Algorithm and Process Overview

The simulation workflow followed these steps:

1. **Face Detection**:
    - The Haar Cascade Classifier was used to detect faces in the video stream.
    - Detected faces were cropped and preprocessed for further recognition.

2. **Face Recognition**:

    - A pre-trained CNN model classified faces into two categories:

        - **SEIF** (authorized).

        - **Not SEIF** (unauthorized).

    - The input face images were resized to 224x224 pixels, normalized, and expanded to match the model's input requirements.

3. **Simulated Door Lock/Unlock Mechanism**:

    - If the face was recognized as **SEIF**, the system simulated unlocking the door by displaying the message: "Door is unlocking...".

    - For unrecognized faces, the system simulated locking the door with: "Door is locking...".

Code Implementation

```
# Detect faces using Haar Cascade
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.05, minNeighbors=5, minSize=(30,

for (x, y, w, h) in faces:
    face_img = frame[y:y+h, x:x+w]

    # Preprocess image for model input
    resized_img = cv2.resize(face_img, (224, 224))
    normalized_img = resized_img.astype('float32') / 255.0
    preprocessed_img = np.expand_dims(normalized_img, axis=0)

    # Make prediction
    prediction_value = model.predict(preprocessed_img)[0][0]

    if prediction_value > 0.4:
        label = "SEIF"
        simulate_door_unlock()
    else:
        label = "Not SEIF"
        simulate_door_lock()

    # Annotate frame
    cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
    cv2.putText(frame, label, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 2)
```

Figure 3.3.3: A key snippet of the simulation code

During the simulation, the system successfully detects and recognizes faces in real-time, as illustrated in Figure 3.3.4 and Figure 3.3.5.

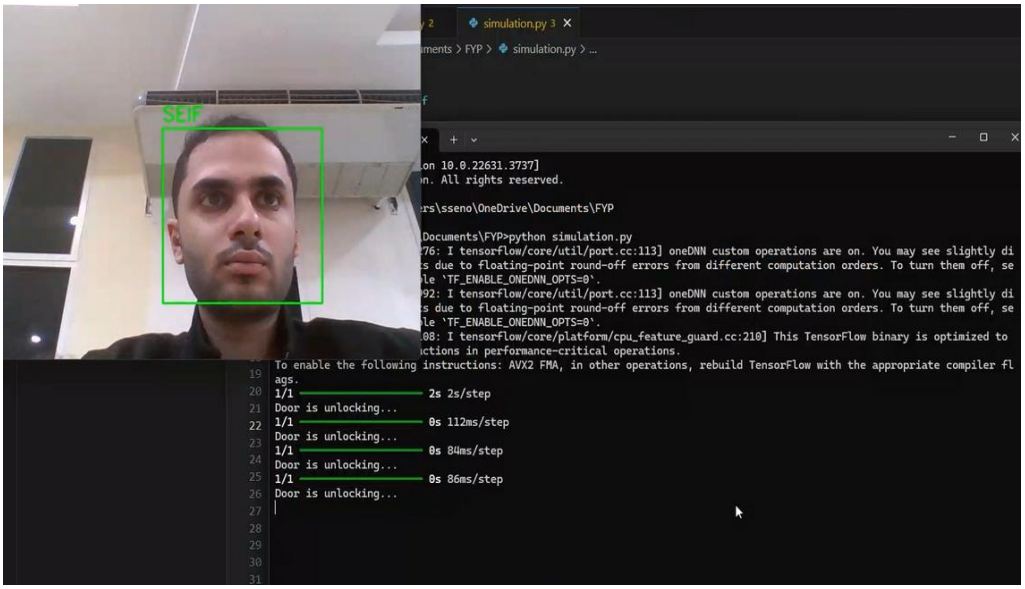

Figure 3.3.4: Face Detection and Recognition During Simulation

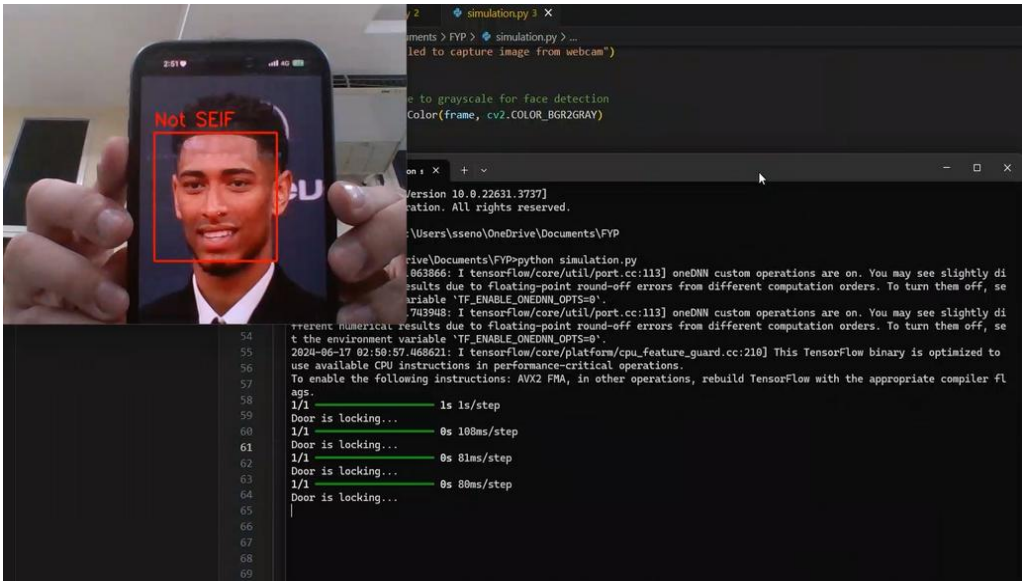

Figure 3.3.5: Face Detection and Recognition During Simulation

The simulation successfully demonstrated the feasibility of the face recognition algorithm and its potential integration with a door lock mechanism. Insights gained from this phase laid the foundation for the hardware implementation and enhancements in FYP2.

3.3.2 Facial Recognition Algorithm

Face recognition systems deployed on edge devices, such as the Raspberry Pi, must balance computational efficiency with recognition accuracy. Traditional deep learning models, such as CNN-based approaches, offer high accuracy but are computationally expensive, making them impractical for real-time processing on low-power devices. In contrast, lightweight algorithms such as the Histogram of Oriented Gradients (HOG) have demonstrated their effectiveness in edge computing scenarios, enabling efficient face detection without requiring GPU acceleration [6].

The Histogram of Oriented Gradients (HOG) algorithm has been widely used for efficient face detection [7]. This method provides real-time performance while maintaining accuracy, making it suitable for resource-constrained environments. HOG-based detection extracts key facial features using gradient orientation patterns, ensuring robustness in different environmental conditions. Additionally, its computational simplicity allows it to outperform more complex deep learning-based approaches in terms of processing speed on embedded hardware.

System Setup for Facial Recognition

Before delving into the algorithm details, it is essential to understand how the system is prepared to execute facial recognition effectively:

1. **Virtual Environment**: The facial recognition script is executed within a Python virtual environment on the Raspberry Pi. This ensures that all dependencies, such as OpenCV, DeepFace, and face_recognition libraries, are correctly managed and isolated from the base system. During the system

setup, the script initializes the camera module and loads necessary libraries. The Raspberry Pi terminal logs provide a detailed summary of these initialization steps. Figure 3.3.6 shows the Raspberry Pi terminal output during the initialization of the camera module, confirming successful detection and configuration.

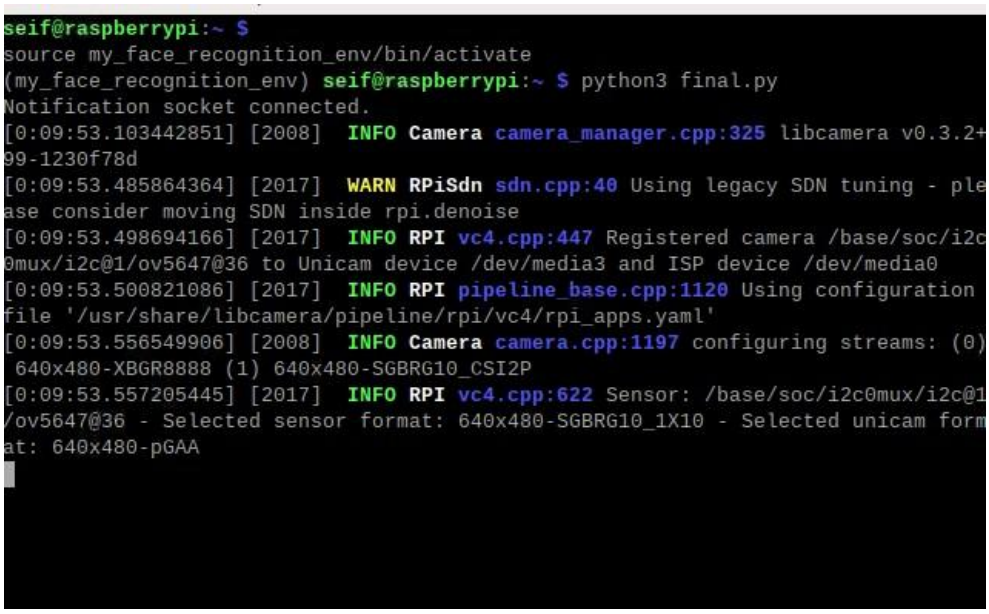

Figure 3.3.6: Raspberry Pi Terminal Output Showing Camera Initialization

Figure 3.3.7 shows the activation of the virtual environment in the Raspberry Pi terminal, ensuring isolated dependency management for the face recognition system.

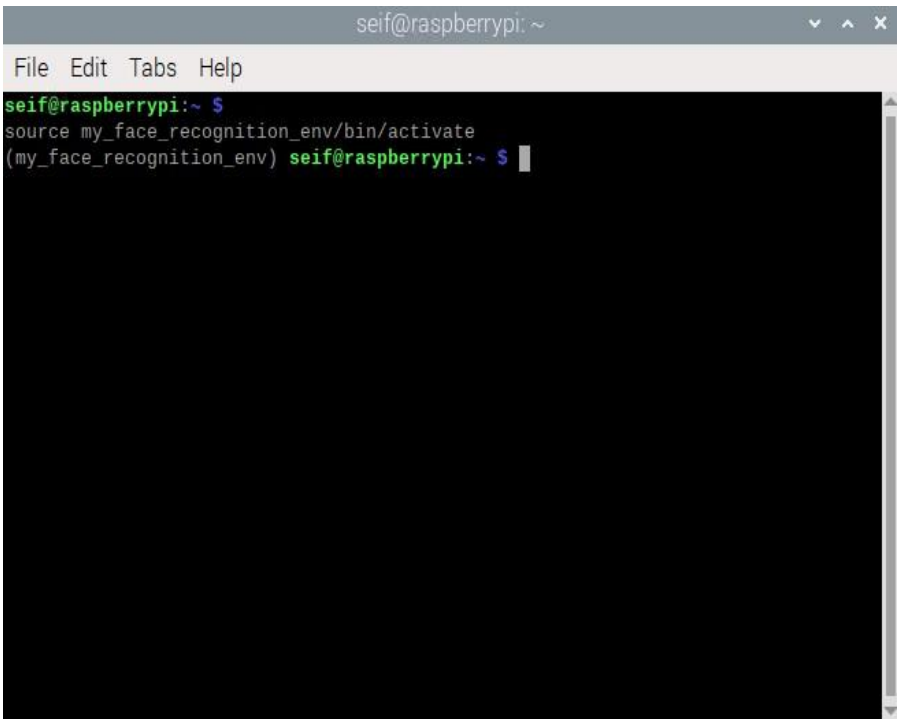

Figure 3.3.7: Virtual environment activated in the terminal

2. **Dataset of Authorized Faces**: A preloaded dataset of photos of authorized individuals is stored on the Raspberry Pi. This dataset is processed during initialization to generate 128-dimensional encodings, which are stored in the encodings file. These encodings are later used for face matching during runtime. The dataset folder, as shown in Figure 3.3.8, contains multiple images used for training and recognizing authorized faces.

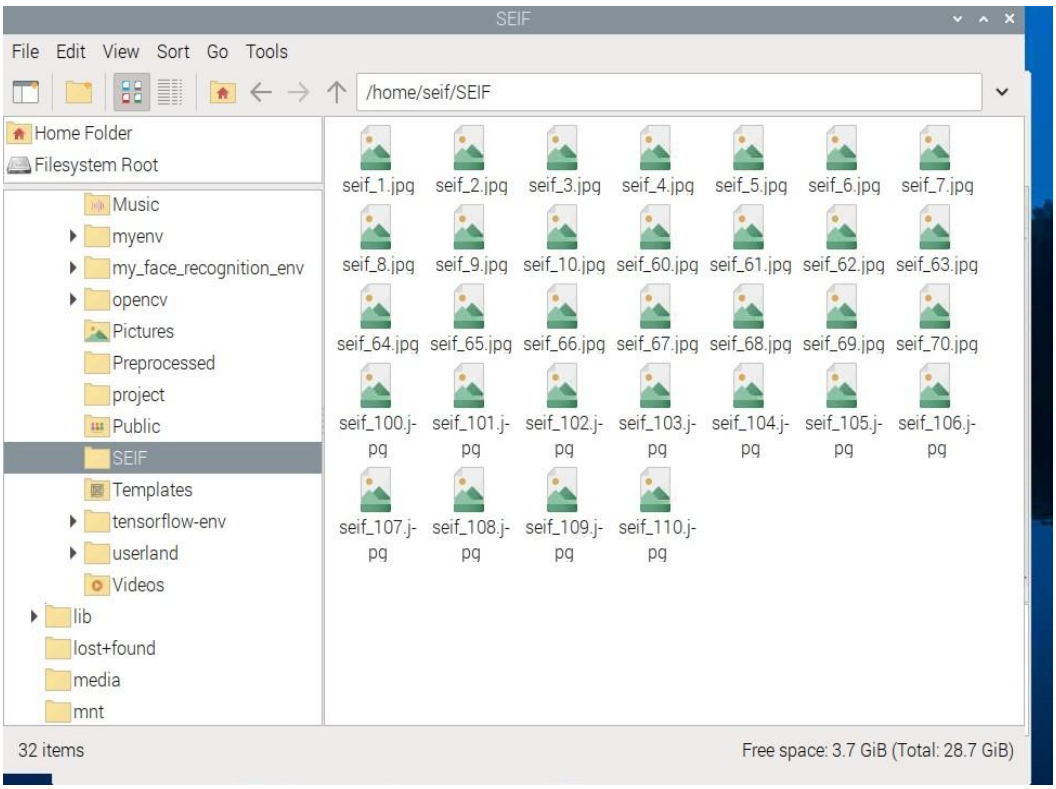

Figure 3.3.8: Dataset folder with my photos

3. **RealVNC Viewer**: RealVNC Viewer was utilized during development and testing to provide remote access to the Raspberry Pi. This tool allowed effective debugging, script execution monitoring, and system setup adjustments. The Raspberry Pi desktop environment accessed via RealVNC is shown in Figure 3.3.9, allowing remote control and monitoring of the system.

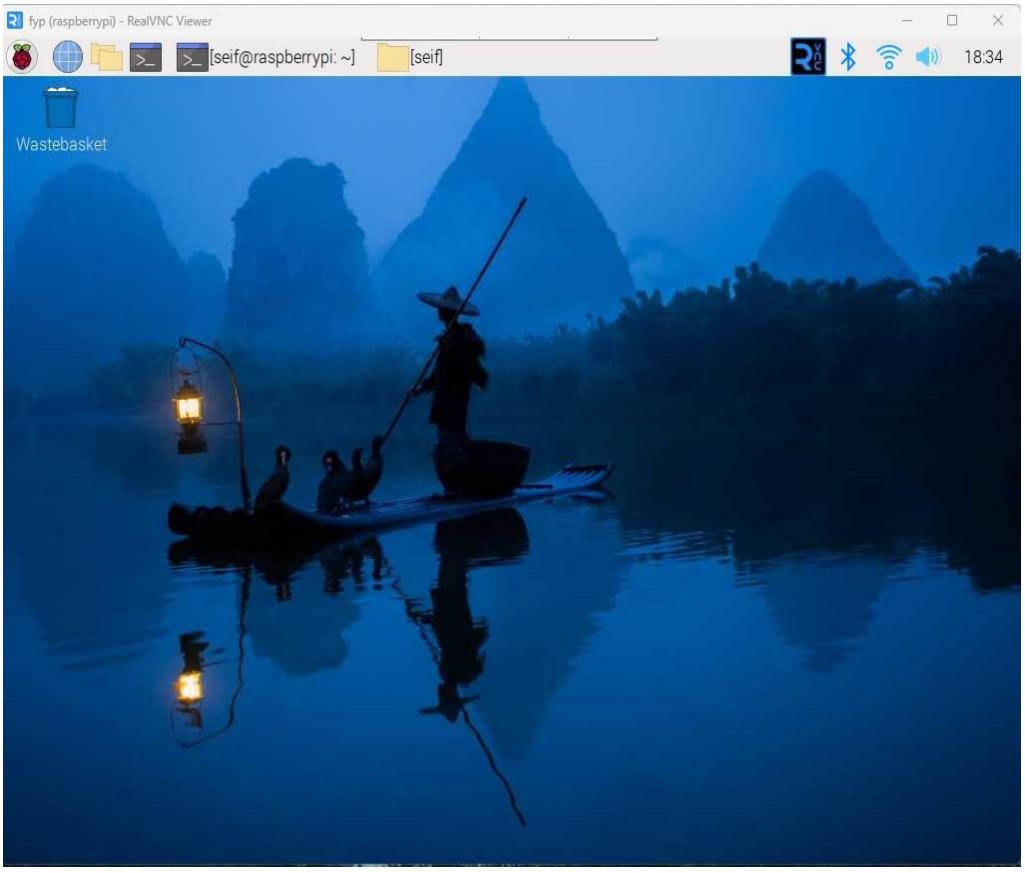

Figure 3.3.9: Raspberry Pi desktop in RealVNC

Process Overview

The facial recognition process comprises three main stages:

1. **Face Detection**:
    - The HOG model detects faces by identifying key facial landmarks, such as eyes, nose, and mouth.
    - This lightweight algorithm ensures real-time performance, even on the Raspberry Pi 3B.

Upon initializing the video stream, the system begins face detection. Figure 3.3.10 illustrates an initial video frame where no faces are detected, ensuring the background is clear before recognition starts.

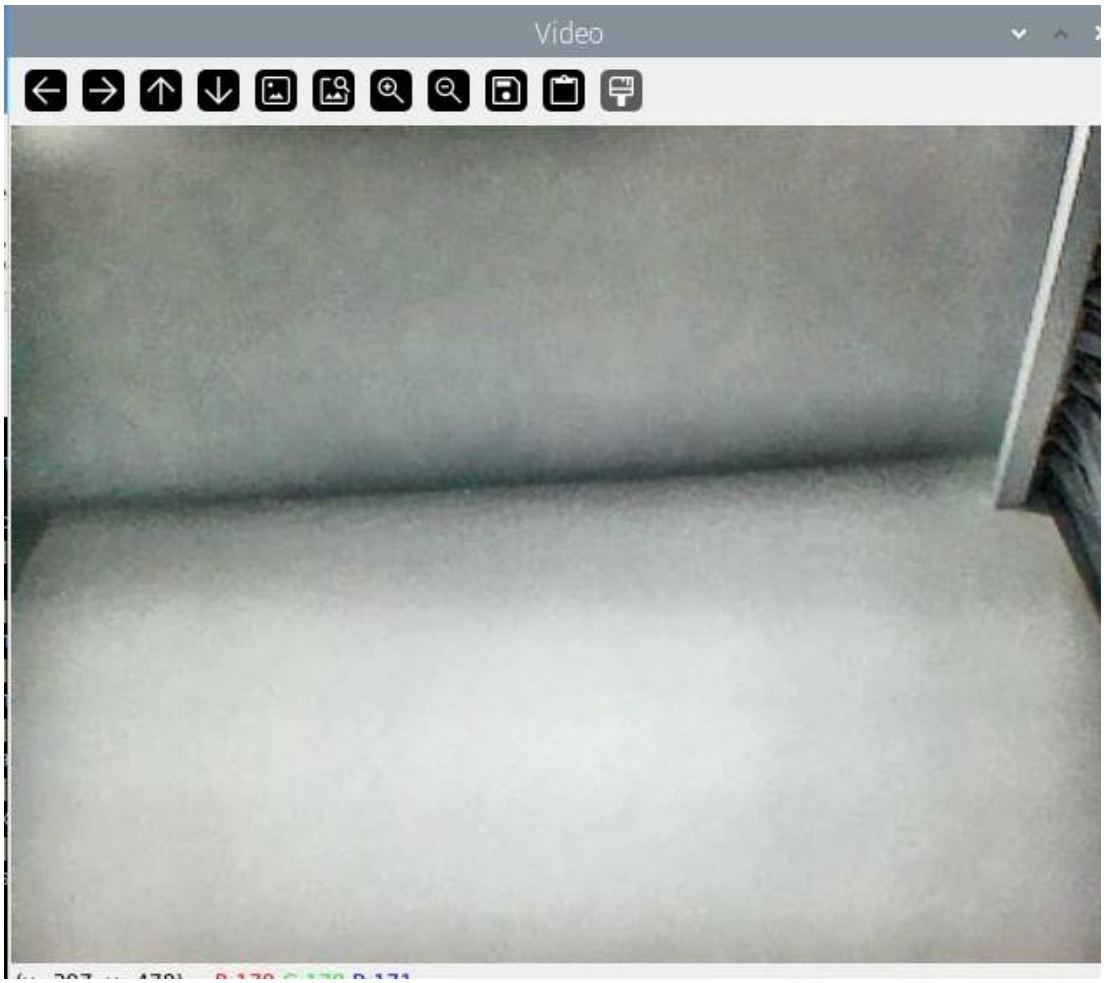

Figure 3.3.10: Initial Video Frame with No Face Detected

2. **Face Encoding**:

    - Once detected, each face is encoded into a 128-dimensional vector that captures unique facial features. These encodings serve as a fingerprint for face recognition.

- "The script retrieves pre-computed encodings from the encodings_file, which contains 128-dimensional vectors for all authorized individuals. These encodings were pre-calculated using a dataset of photos stored on the Raspberry Pi."

3. **Face Matching**:

   - The system compares the newly detected face encoding with stored encodings of authorized individuals using Euclidean distance.

   - If the distance is below a pre-defined threshold (0.4), the face is recognized as a match.

As the system processes video frames, it detects and recognizes faces in real time. Figure 3.3.11 demonstrates the real-time face recognition process, displaying the detected face along with the identified emotion and access status.

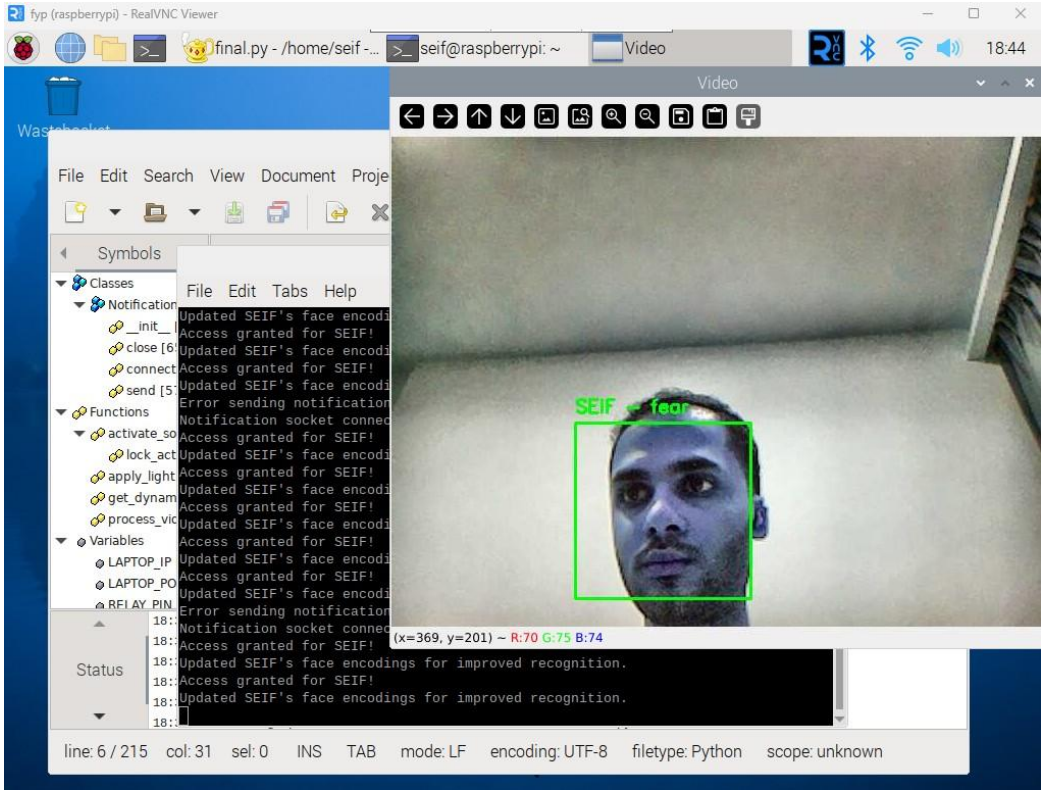

Figure 3.3.11: Face Detection and Recognition in Real Time

To provide a structured overview of the facial recognition process, Table 3.3.1 summarizes the key steps involved in the algorithm. This table outlines the sequential operations, starting from capturing the frame to displaying the recognition results and triggering corresponding actions.

Table 3.3.1: Overview of the Facial Recognition Algorithm

Step	Description
Capture Frame	Captures a video frame using the Raspberry Pi camera.
Enhance Lighting	Applies CLAHE and gamma correction to normalize brightness.
Detect Faces	Identifies facial regions using the HOG-based detector.
Encode Faces	Generates a 128-dimensional encoding for detected faces.
Compare with Encodings	Matches face encodings with the stored dataset using Euclidean distance.
Output Result	Displays recognition results and triggers actions (e.g., notifications).

Lighting Normalization and Gamma Correction

Lighting inconsistencies, such as low-light conditions or bright backgrounds, can negatively impact face detection and recognition accuracy. To address this, the system implements the following enhancements**:**

1. **Contrast Limited Adaptive Histogram Equalization (CLAHE):**

    - CLAHE enhances image contrast by redistributing the intensity values across the image, ensuring uniform brightness.

    - This is particularly useful in dim or uneven lighting environments.

2. **Gamma Correction:**

    - Gamma correction adjusts the brightness dynamically, enhancing darker regions of the frame to reveal facial features.

    - The gamma correction formula is:

$$I_{corrected} = 255X\left(\frac{I_{original}}{255}\right)^{\gamma}$$

Where:

- $I_{original}$ is the original pixel intensity.
- γ is the gamma adjustment factor (e.g., γ = 1.2).
- $I_{corrected}$ is the brightness-adjusted intensity.

Lighting conditions significantly impact face detection accuracy. Figure 3.3.12 illustrates how the implemented lighting normalization technique enhances visibility and ensures consistent facial recognition even in low-light environments.

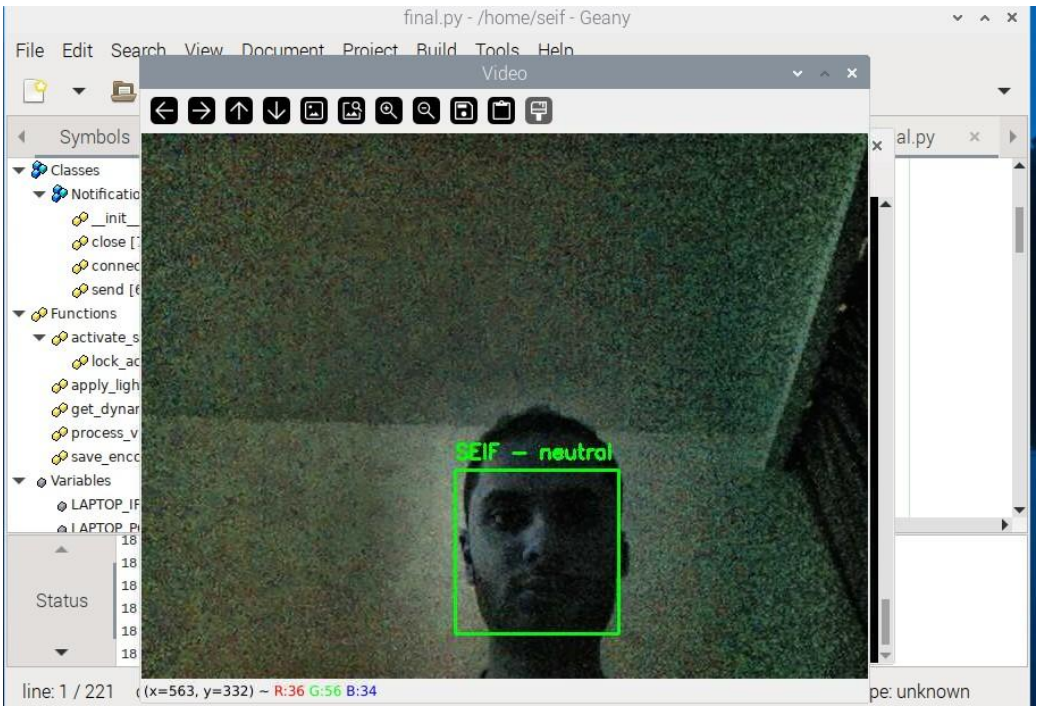

Figure 3.3.12: Effect of Lighting Normalization on Face Detection

Integration into the System

The algorithm is integrated into the system via a Python script that processes each video frame captured by the Raspberry Pi Camera Module V2. It dynamically adjusts the brightness using the apply_lighting_adaptation function, which incorporates CLAHE and gamma correction. Detected faces are resized to reduce computational load, and their locations are scaled back for precise visualization.

To ensure real-time performance, the system processes resized frames at 25% of their original size during detection. Detected face locations are scaled back to original dimensions for precise matching. Additionally, the system employs dynamic frame skipping, adjusting the number of frames processed based on CPU usage to avoid overloading the Raspberry Pi.

Code Implementation

Below in figure 3.3.13 is the key snippet of the facial recognition algorithm from the project's Python script:

```
# Enhance lighting conditions
frame = apply_lighting_adaptation(frame)
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# Resize frame for faster processing
small_frame = cv2.resize(rgb_frame, (0, 0), fx=0.25, fy=0.25)
face_locations = face_recognition.face_locations(small_frame, model="hog")
face_locations = [(top * 4, right * 4, bottom * 4, left * 4) for (top, right, bottom, left

# Encode detected faces
face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

# Compare with known encodings
face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
if face_distances.size > 0 and face_distances.min() < 0.4:
    name = known_face_names[face_distances.argmin()]
else:
    name = "Unknown"
```

Figure 3.3.13: Implementation of Lighting Adaptation and Face Detection

The facial recognition algorithm incorporates several optimizations to enhance accuracy and efficiency. Table 3.3.2 summarizes the key features, including face detection, encoding, matching, and real-time performance enhancements.

Table 3.3.2: Key Features of the Facial Recognition Algorithm

Feature	Description
Face Detection	HOG algorithm detects faces in real-time with computational efficiency.
Face Encoding	128-dimensional vectors ensure unique representation of facial features.
Face Matching	Euclidean distance calculates similarity between face encodings.
Lighting	CLAHE improves brightness uniformity for better detection in

Normalization	varying lighting.
Gamma Correction	Dynamically adjusts frame brightness for low-light accuracy.
Real-Time Optimization	Frame resizing and adaptive skipping ensure real-time performance.

3.3.3 Adaptive Learning and Unknown Face Addition

The ability to adapt dynamically to new inputs is one of the most significant enhancements of the Raspberry Pi-based face recognition system. Unlike traditional face recognition systems that rely on static datasets, this system incorporates adaptive learning, dynamically updating stored face encodings over time [8]. This feature enhances recognition accuracy by learning from repeated interactions and allows the system to evolve alongside the user.

A key limitation of conventional face recognition systems is their reliance on manually updated databases, which must be retrained whenever a new user is added. To overcome this challenge, this project integrates adaptive learning, enabling the system to update face encodings dynamically and recognize new users without requiring full retraining. Recent research has shown that incremental learning techniques significantly improve recognition accuracy by continuously refining stored facial encodings based on new observations [9].

This section outlines the design, implementation, and significance of these features in detail, highlighting how adaptive learning enhances usability, scalability, and long-term reliability.

1. Adaptive Learning: Improving Recognition Over Time

Adaptive learning allows the system to refine its knowledge of authorized individuals with every interaction. This feature addresses common challenges in face recognition, such as slight variations in appearance due to:

- Lighting conditions: Changes in illumination may alter facial features.
- Aging effects: Subtle changes in facial contours over time.
- Dynamic factors: Accessories like glasses or hats.

Each time a known face is detected, the system updates its stored encoding to reflect the most recent observation. This iterative learning process ensures the stored data evolves alongside the user, minimizing false negatives and improving recognition accuracy.

How It Works:

1. During each recognition cycle, the system compares the detected face's encoding with stored encodings.
2. If a match is found, the corresponding stored encoding is updated with the newly detected encoding.
3. This ensures that minor variations in the face are captured over time, enhancing the reliability of recognition.

Code Snippet:

```
if best_match_distance < 0.4:
    name = known_face_names[best_match_index]
    known_face_encodings[best_match_index] = face_encoding  # Update encoding
    print(f"Updated {name}'s face encodings for improved recognition.")
```

Figure 3.3.14: Code snippet of Adaptive learning

The system enhances recognition accuracy by continuously updating stored face encodings for known users. This ensures improved recognition over time, as shown in Figure 3.3.15.

```
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
```

Figure 3.3.15: Updating Encodings for Known Faces

Advantages:

- **Consistency**: Maintains recognition accuracy over time without requiring manual updates.
- **Adaptability**: Handles natural changes in a user's appearance.
- **Efficiency**: Reduces the need for extensive retraining.

2. Unknown Face Detection and Dynamic Addition

The system identifies faces that do not match stored encodings as "Unknown." This triggers a prompt to the user, allowing them to dynamically add new individuals to the list of authorized faces. This feature ensures scalability and eliminates the need for preloading datasets for every user.

Step-by-Step Process:

1. Prompting for Unknown Faces

When an unrecognized face is detected, the system displays a real-time prompt on the Raspberry Pi terminal, requesting the user to decide whether to authorize the face. The prompt includes the following options:

- Press 'a': To authorize the detected face and add it to the dataset of authorized individuals.

- Press 'q': To ignore the detected face and quit the operation.

This functionality is implemented in the Python script, where the system identifies faces as “Unknown” when no match is found in the encodings_file. Upon receiving user input to authorize the face, the system saves the corresponding 128-dimensional face encoding along with the name provided by the user. This interactive feature facilitates adaptive learning, as shown in Figure 3.3.16.

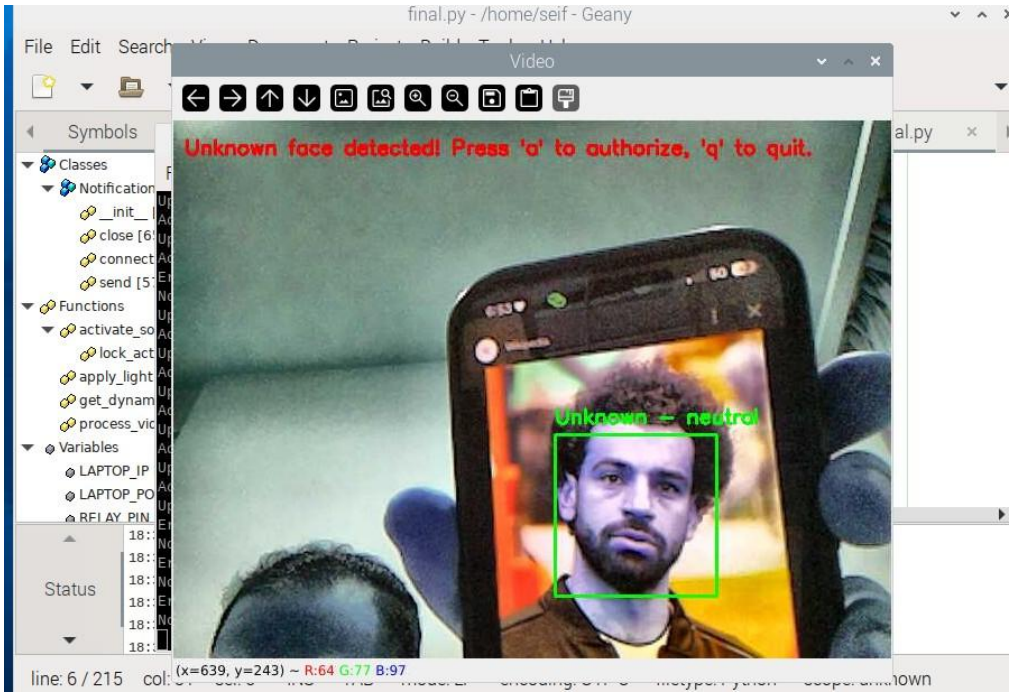

Figure 3.3.16: Unknown Face Detected with Authorization Prompt

2. Storing New Face Encodings

Once a face is authorized, the system dynamically updates its dataset. The 128-dimensional encoding of the newly detected face, along with the name entered by the user, is stored in the encodings_file. This ensures that the face will be recognized in subsequent interactions without requiring a complete system restart or retraining process.

This dynamic addition process as shown in figure 3.3.17 is efficient and secure, maintaining the robustness of the face recognition system while adapting to new users.

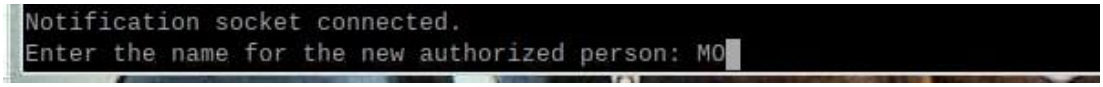

Figure 3.3.17: Adding the unknown face to be authorised

3. Recognizing Newly Added Faces

After adding a new face, the system immediately integrates the updated encoding into the recognition pipeline. Subsequent frames demonstrate the system’s ability to correctly identify the newly authorized individual by displaying their name and emotional state on the video stream. This step confirms the success of the adaptive learning feature and showcases the system's capability to evolve dynamically.

Once the unknown face is authorized and added to the system, it is successfully recognized in subsequent detections, as demonstrated in Figure 3.3.18.

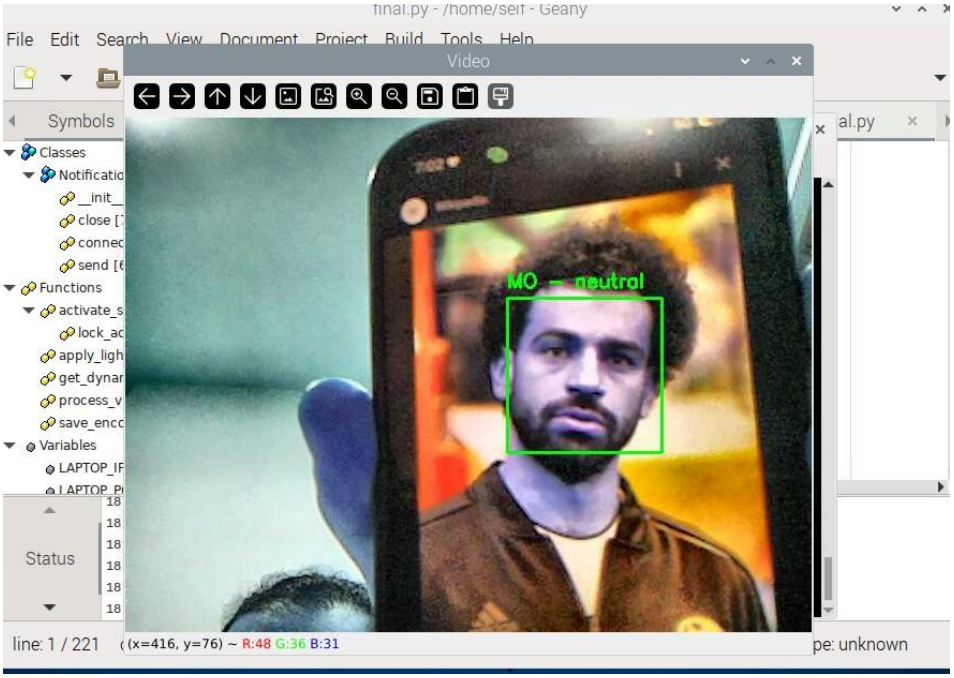

Figure 3.3.18: Successful Recognition of Newly Authorized Face

Code Snippet:

```
elif key == ord('a') and unknown_face_detected:
    new_name = input("Enter the name for the new authorized person: ")
    known_face_encodings.append(new_face_encoding)
    known_face_names.append(new_name)
    save_encodings_to_file(known_face_encodings, known_face_names, encodings_file)
    print(f"{new_name} added to the authorized list.")
    unknown_face_detected = False  # Reset after adding
```

Figure 3.3.19: Code snippet of adding unknown faces

3. Integration into the Recognition Workflow

The adaptive learning and dynamic addition functionalities are seamlessly integrated into the recognition workflow. The following workflow illustrates this integration:

1. **Detect a Face**: The system identifies a face in the frame and computes its encoding.

2. **Match with Known Encodings**:
    - If a match is found, the encoding for the recognized individual is updated.
    - If no match is found, the face is classified as "Unknown."
3. **Prompt for Action**:
    - If classified as "Unknown," the user is prompted to authorize the face and assign it a name.
4. **Store New Encodings**:
    - The new encoding and name are stored persistently for future recognition.

The adaptive learning workflow allows the system to dynamically update its facial recognition database, improving recognition accuracy over time. Table 3.3.3 outlines the sequential steps involved in detecting, encoding, matching, and updating face data for both known and unknown individuals.

Table 3.3.3: Workflow for Adaptive Learning and Unknown Face Addition

Step	Description
Detect Face	Use HOG algorithm to detect a face in the frame.
Compute Encoding	Generate a 128-dimensional encoding for the detected face.
Compare with Encodings	Match the encoding with stored encodings using Euclidean distance.
Update Known Encodings	If matched, update the encoding for the recognized individual.
Prompt for Authorization	If unmatched, prompt the user to authorize the unknown face.
Add New Encoding	Append the new encoding and name to the list, and save to the encodings_file.

4. Significance of These Features

Scalability:

- Dynamically adding new individuals ensures the system can grow with evolving user requirements.

User-Centric:

- By prompting users to authorize unknown faces, the system gives full control over access permissions.

Future-Proof:

- Adaptive learning ensures the system remains accurate and relevant as users' appearances change over time.

Adaptive learning and dynamic addition of unknown faces significantly enhance the flexibility, scalability, and usability of the face recognition door lock system. These features eliminate the need for retraining or manually updating datasets while ensuring reliable performance in real-world scenarios. By integrating these functionalities, the system evolves dynamically, offering a robust and user-friendly access control solution.

3.3.4 Emotion Detection

Emotion detection is a critical feature of the system, enhancing its capabilities beyond basic facial recognition. By analyzing facial expressions in real-time, the system determines the dominant emotion displayed by a recognized or unknown individual. This functionality adds depth to the user experience and potential applications, such as detecting stress, happiness, or anxiety in specific contexts.

Implementation and Integration

For enhanced facial recognition and emotion detection, the system integrates DeepFace, a deep learning-based facial analysis library capable of performing real-time facial verification and emotion classification [10]. DeepFace employs a convolutional neural network (CNN) model pre-trained on large datasets, ensuring high accuracy in recognizing user expressions such as happiness, sadness, and anger.

Recent studies highlight the effectiveness of CNN-based emotion recognition models, demonstrating their ability to extract subtle facial features that contribute to accurate classification [11].

1. **DeepFace Library**:
    - DeepFace provides pre-trained models that classify facial expressions into predefined emotional categories: happiness, sadness, anger, surprise, fear, disgust, and neutrality.
    - The library was selected due to its high accuracy and efficient integration with Python, making it suitable for real-time processing on the Raspberry Pi.
2. **Workflow**:
    - **Face Region Extraction**: The system captures frames from the Raspberry Pi Camera Module V2. Once a face is detected, the bounding box of the detected face is used to isolate the facial region.
    - **Emotion Analysis**: The extracted face region is passed to DeepFace's emotion recognition module. The module outputs the probabilities for each emotion and identifies the dominant emotion.

- **Display and Notifications**: The detected emotion is displayed on the live video feed and sent to the connected notification system. For example, if a user is detected as "fear," the system sends a notification stating, "Detected emotion: Fear."

The system integrates real-time emotion detection alongside face recognition to provide additional insights into user interactions. Figure 3.3.20 illustrates an instance where the model successfully detects and labels the user's emotion in real time.

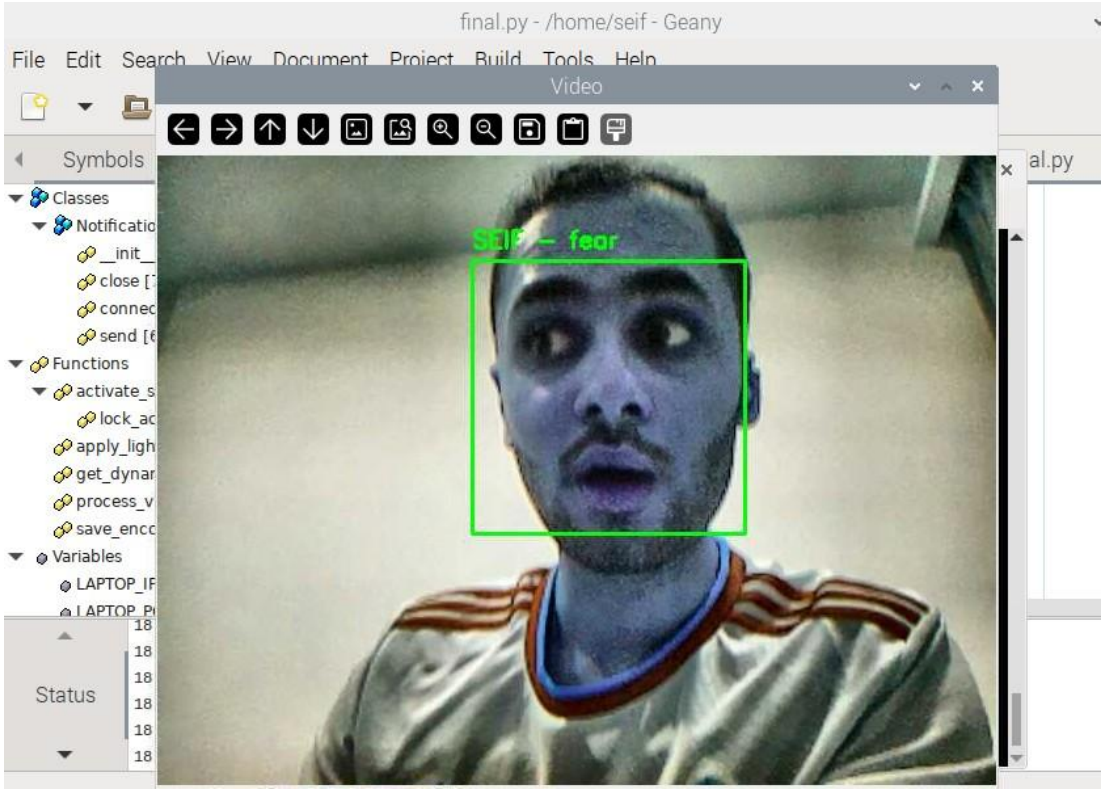

Figure 3.3.20: Real-Time Emotion Detection Display

Code Snippet:

```
for face_encoding, face_location in zip(face_encodings, face_locations):
    try:
        top, right, bottom, left = face_location
        face_region = rgb_frame[top:bottom, left:right]  # Extract face region
        emotion_result = DeepFace.analyze(face_region, actions=["emotion"], enforce_d
        detected_emotion = emotion_result["dominant_emotion"]
        notification_client.send(f"Detected emotion: {detected_emotion}")
    except Exception as e:
        print("Error in emotion detection:", e)
```

Figure 3.3.21: Code snippet for emotion detection

Enhancements for Real-Time Performance

To ensure the emotion detection module operates efficiently on the Raspberry Pi, several optimizations were implemented:

1. **Dynamic Frame Skipping**:
    - Emotion detection is computationally intensive. To balance performance and responsiveness, the system skips a configurable number of frames during processing based on CPU usage.
    - For example, during high CPU load, emotion detection is performed every 10th frame instead of every frame.
2. **Bounding Box Resizing**:
    - The bounding box containing the facial region is resized to a fixed resolution before passing it to DeepFace. This minimizes computation while preserving recognition accuracy.

System Outputs and Use Cases

1. **Real-Time Visual Feedback**:

    - Detected emotions are displayed alongside the recognized name in the live video feed. For example, "SEIF – Happy" is overlaid on the bounding box of a recognized individual as shown in figure 3.3.22.

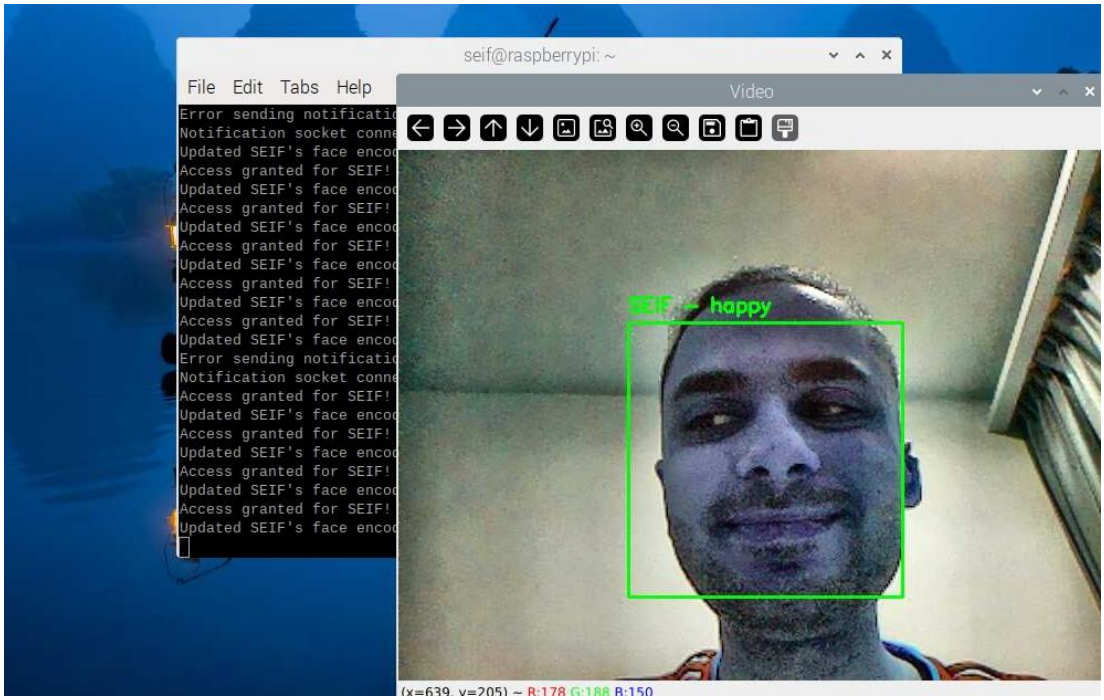

Figure 3.3.22: Detected Emotion Displayed in Real-Time Video Feed

2. **Notification System**:

    - Emotional states are sent to a remote laptop via the notification client. Notifications like "Detected emotion: Neutral" or "Detected emotion: Fear" provide real-time feedback for remote monitoring as shown in figure 3.3.23.

Figure 3.3.23: Emotion Detection Notifications Displayed on the Remote Laptop

3. **Use Cases**:

    - **Home Automation**: The system could adapt smart home settings (e.g., lighting or music) based on detected emotions.

    - **Security**: Emotion detection can add another layer of analysis by flagging unusual emotions like fear or anger.

    - **User Experience**: Enhances interaction by providing personalized responses based on emotional states.

Performance Evaluation

- **Accuracy**:

    - The emotion detection module achieves high accuracy for distinct emotions such as "happy" or "neutral" but may face challenges with subtle expressions.

- **Speed**:

    - With the optimizations implemented, the system achieves near real-time processing, averaging 3–5 frames per second for emotion detection.

Limitations and Future Improvements

While the emotion detection feature adds significant value, there are areas for future improvement:

1. **Complex Emotions**:
    - The system currently classifies basic emotions. Future iterations could include more nuanced emotional states.
2. **Environmental Factors**:
    - Varying lighting or occlusions can affect the accuracy of emotion recognition. Incorporating advanced preprocessing techniques could mitigate this.
3. **Hardware Acceleration**:
    - Utilizing hardware-accelerated libraries (e.g., OpenVINO or TensorRT) could significantly improve processing speed on the Raspberry Pi.

The emotion detection module enhances the system by classifying facial expressions into predefined categories, optimizing processing speed, and integrating with the notification system. Table 3.3.4 summarizes the key features of the implemented emotion detection framework.

Table 3.3.4: Summary of Emotion Detection Features

Feature	Description
Emotion Categories	Happiness, sadness, anger, surprise, fear, disgust, neutrality
Real-Time Display	Displays detected emotions on the live video feed
Notification Integration	Sends detected emotions to the remote notification client

Optimizations	Frame skipping and bounding box resizing for real-time performance

3.3.5 Notification System

The notification system is a critical component of the face recognition door lock system, enabling real-time feedback to the user about access events. This system provides textual and audible notifications about recognized faces, detection of unknown faces, and even emotions detected during the recognition process. It is implemented using a socket-based client-server communication model that operates efficiently without the need for additional hardware or dedicated mobile applications.

Face recognition systems can be enhanced through real-time notifications using IoT-based communication protocols. This implementation ensures that access events are logged and alerts are provided in real time [12].

System Design and Purpose

The notification system was designed with the following goals:

1. **Real-Time Updates**: Provide immediate feedback about the face recognition process, including successful recognition, unknown face detection, and emotional states of detected individuals.

2. **Enhanced Usability**: Deliver notifications in both textual and audible formats using a text-to-speech (TTS) system running on the server (laptop).

3. **Simplicity and Efficiency**: Operate seamlessly without requiring a dedicated mobile app or complex configurations, ensuring ease of use for end-users.

The Raspberry Pi acts as the client in this system, sending messages to the server (laptop) over a TCP socket connection. The server processes these notifications, displays them in the terminal, and generates an audible alert using TTS. This ensures the user is always informed about system activity, whether they are near the device or remotely monitoring it.

System Workflow

The workflow of the notification system involves the following steps:

1. **Triggering Notifications**:
    - Events such as recognizing an authorized face, detecting an unknown face, or identifying an emotion trigger the notification system.
    - A message is prepared with relevant details, such as the name of the recognized individual or the detected emotion.
2. **Client-Side Communication**:
    - The Raspberry Pi establishes a TCP connection with the server and sends the notification message.
3. **Server-Side Processing**:
    - The server receives the notification, logs it in the terminal, and uses TTS to audibly announce the message.
4. **Feedback to the User**:
    - The user receives real-time updates about system activity, enabling them to take action (e.g., authorizing an unknown face) or simply stay informed.

Implementation Details

Client-Side Script (Raspberry Pi)

The Raspberry Pi is responsible for sending notifications to the server whenever a recognition event occurs. The send_notification function handles this communication as shown in figure 3.3.24:

```
# Function to send notifications to the server
def send_notification(message):
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client_socket:
            client_socket.connect((LAPTOP_IP, LAPTOP_PORT))
            client_socket.sendall(message.encode('utf-8'))
    except Exception as e:
        print(f"Error sending notification: {e}")
```

Figure 3.3.24: The send_notification function

Key Features:

- **TCP Connection**: Ensures reliable delivery of messages.
- **Error Handling**: Catches and logs errors in case the server is unreachable.
- **Dynamic Messaging**: Supports customizable messages based on the recognition event.

The notification system is triggered based on face recognition results, where authorized users receive an access notification, and unknown faces prompt a security alert. Figure 3.3.25 illustrates the conditional logic used to send these notifications in real time.

```python
if best_match_distance < 0.4:
    name = known_face_names[best_match_index]
    send_notification(f"Access granted for {name}!")
else:
    send_notification("Unknown face detected!")
```

Figure 3.3.25: Example of Triggering Notifications

Server-Side Script (Laptop)

The server script continuously listens for incoming connections from the Raspberry Pi. Upon receiving a notification, it processes the message and provides feedback using TTS as shown in figure 3.3.26.

noti.py

C: > Users > sseno > OneDrive > Documents > FYP > noti.py

```python
import socket
import pyttsx3

# Real-time notification setup
HOST = "0.0.0.0"  # Listen on all interfaces
PORT = 5000

engine = pyttsx3.init()

# Function to speak the notification
def speak_notification(message):
    engine.say(message)
    engine.runAndWait()

# Server setup
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as server_socket:
    server_socket.bind((HOST, PORT))
    server_socket.listen()
    print(f"Listening for notifications on {HOST}:{PORT}...")

    while True:
        client_socket, address = server_socket.accept()
        with client_socket:
            print(f"Connection from {address}")
            message = client_socket.recv(1024).decode('utf-8')
            print(f"Received: {message}")
            speak_notification(message)
```

Figure 3.3.26: Server-Side Script

Key Features:

- **Continuous Listening**: The server runs indefinitely, ensuring all notifications are received.
- **TTS Integration**: Converts textual notifications into speech for audible feedback.
- **Simple Interface**: Displays notifications in the terminal for easy monitoring.

Figure 3.3.27 shows the command prompt interface where the server is actively running and awaiting notifications.

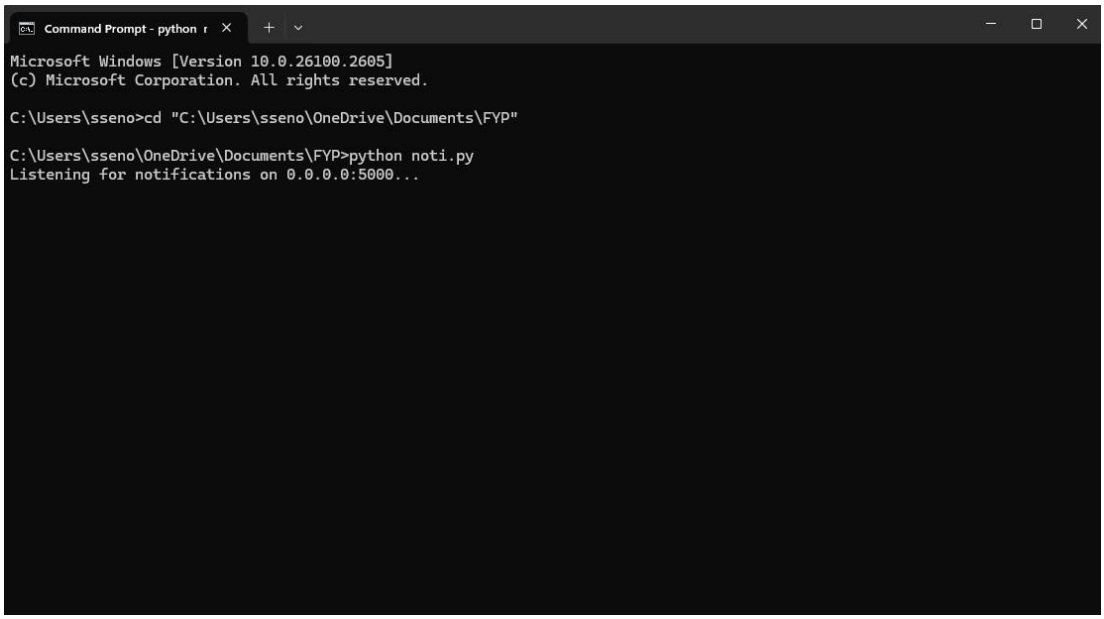

Figure 3.3.27: Notification Server Running on Laptop

Integration with the Recognition System

The notification system is seamlessly integrated into the facial recognition loop. Notifications are triggered based on specific events, such as:

1. **Authorized Face Detected:**
    - Message: "Access granted for [Name]".
    - Additional Feedback: Updates about detected emotions, e.g., "Detected emotion: [Emotion]".
    - Ensures users are notified of successful recognition in real time.
2. **Unknown Face Detected:**
    - Message: "Unknown face detected!"
    - Action Prompt: Provides an option to authorize the individual.
3. **Emotion Detected:**
    - Message: "Detected emotion: [Emotion]".
    - Adds an intelligent layer to user feedback.

Figure 3.3.28 displays the integration of face recognition and emotion detection, where the system grants access to recognized faces and sends emotion-based notifications in real time.

```
if best_match_distance < 0.4:
    name = known_face_names[best_match_index]
    send_notification(f"Access granted for {name}!")
else:
    send_notification("Unknown face detected!")

# Emotion detection
if total_frames_processed % frame_skip == 0:
    emotion_result = DeepFace.analyze(face_region, actions=["emotion"], enforce_detection=
    detected_emotion = emotion_result[0]["dominant_emotion"]
    send_notification(f"Detected emotion: {detected_emotion}")
```

Figure 3.3.28: Integration code

Figure 3.3.29 illustrates the real-time notification system displaying access logs and detected emotions. The system logs face recognition events and communicates detected emotions with text-based alerts.

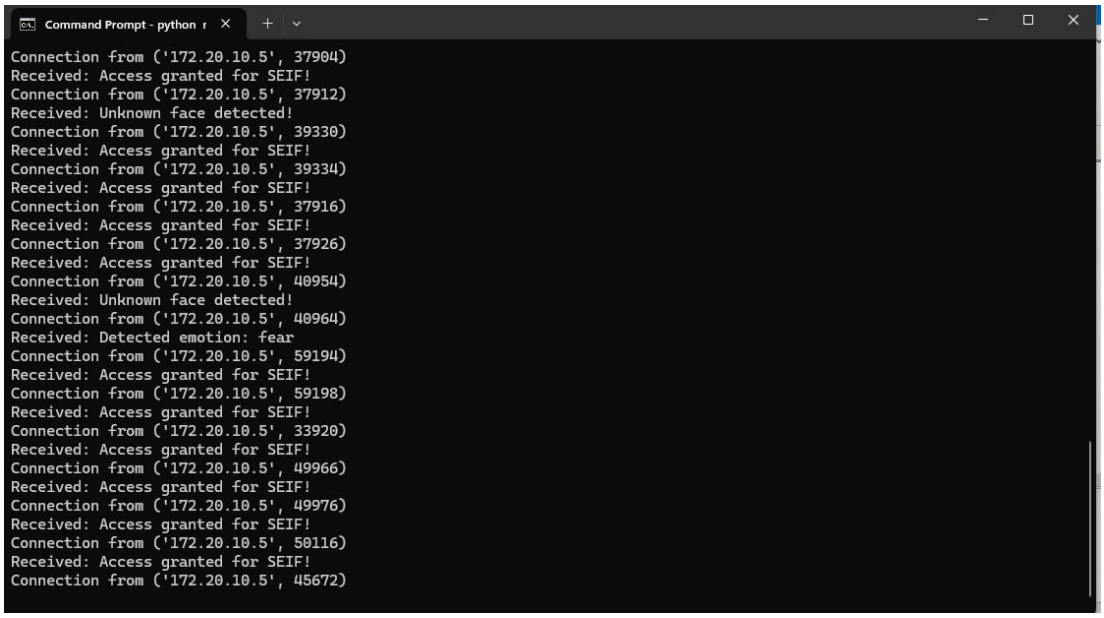

Figure 3.3.29: Real-Time Notification with Text-to-Speech Feedback

The notification system enhances the usability and efficiency of the face recognition door lock system by providing real-time feedback to users. Its socket-based architecture ensures low-latency communication, while the integration of TTS delivers intuitive audible notifications. By eliminating the need for additional applications or hardware, this system aligns with the project's goal of simplicity and accessibility. Whether granting access to authorized individuals, alerting about unknown faces, or reporting detected emotions, the notification system plays a vital role in the overall functionality of the project.

3.4 Limitations and Challenges

While the Raspberry Pi-based face recognition door lock system successfully achieved its primary objectives, a few limitations were encountered during development and testing. These challenges are categorized into hardware, software,

and environmental aspects, with potential solutions proposed for future improvements.

3.4.1 Hardware Limitations

1. **Processing Power of Raspberry Pi:**
    - The Raspberry Pi Model 3B performed efficiently for most tasks but showed slight delays when handling intensive computations, such as real-time emotion detection and face recognition under high-resolution settings.
2. **Camera Module Constraints:**
    - The Raspberry Pi Camera Module V2 provided adequate performance but could benefit from higher resolution for improved accuracy in dynamic scenarios.
3. **Power Supply Dependency:**
    - The system’s reliance on uninterrupted power highlighted the need for a backup power solution to ensure continuous operation during outages.

3.4.2 Software Limitations

1. **Emotion Detection Accuracy**:
    - The DeepFace library reliably classified distinct emotions (e.g., happy, neutral) but occasionally struggled with ambiguous expressions.

2. **Lighting Adaptation**:

    - While lighting normalization techniques improved performance under varied conditions, extreme scenarios such as strong backlighting still presented challenges.

3.4.3 Environmental Challenges

1. **Low-Light Conditions**:

    - Performance in extremely dim environments was suboptimal, despite implementing lighting normalization techniques.

2. **Outdoor Testing**:

    - External factors like sunlight glare and shadows affected detection accuracy, especially in uncontrolled outdoor settings.

3.4.4 Proposed Improvements

To address these limitations, the following enhancements are proposed:

1. **Hardware Upgrades**:

    - Upgrade to Raspberry Pi Model 4 for improved processing power and efficiency.

    - Incorporate a higher-resolution camera module to enhance detection accuracy.

    - Add an uninterrupted power supply (UPS) module for reliable operation during power outages.

2. **Software Enhancements**:

    - Utilize advanced machine learning models for improved accuracy under challenging conditions.

    - Enhance preprocessing techniques to handle extreme lighting scenarios more effectively.

3. **User Interface Improvements**:

    - Develop a user-friendly mobile application for managing authorized faces and monitoring the system remotely.

4. **Environmental Adaptation**:

    - Integrate infrared (IR) sensors or night-vision cameras for better performance in low-light environments.

The face recognition door lock system demonstrates robust functionality and innovative features, with a few manageable limitations. Addressing these challenges through targeted hardware and software improvements will further enhance the system's scalability, reliability, and ease of use in diverse real-world scenarios. These advancements lay the foundation for future iterations and expanded applications of the system.

3.5 Conclusion

Chapter 3 has provided a comprehensive overview of the design and implementation of the Raspberry Pi-based face recognition door lock system. This chapter detailed both the hardware and software aspects of the project, highlighting the critical components, algorithms, and processes that enable the system's functionality.

The hardware implementation demonstrated how various components, including the Raspberry Pi, camera module, relay module, and solenoid lock, were integrated into a cohesive setup. The software implementation showcased the use of advanced algorithms, such as HOG for face detection and DeepFace for emotion detection, to achieve real-time recognition and notification capabilities.

While the system has been successfully implemented, several limitations, such as hardware constraints, lighting challenges, and user interaction complexities, were identified. Addressing these issues through proposed improvements can significantly enhance the system's performance and scalability.

Overall, the design and implementation form a solid foundation for a robust, scalable, and user-friendly face recognition door lock system, setting the stage for further developments and expanded applications in future iterations.

CHAPTER 4 DATA PRESENTATION AND DISCUSSION OF FINDINGS

This chapter presents the results obtained from testing the Raspberry Pi-based face recognition door lock system, followed by an in-depth discussion of these findings. The system was evaluated under various conditions, including different lighting environments, varying distances, and diverse emotional expressions. Numerical results and visual evidence are provided to validate the system's performance and highlight its strengths and limitations. The findings are also compared to existing studies, demonstrating the advancements achieved in this project.

4.1 Data Presentation

This section provides a detailed presentation of the results obtained during the system's operation. Data is organized into tables and figures to illustrate performance metrics such as detection times, recognition accuracy, and notification delays. Each feature of the system, including emotion detection, adaptive learning, and real-time notifications, is analysed in detail. This structured approach highlights the system's capabilities and areas for improvement.

4.1.1 Face Recognition in Varying Conditions

The face recognition system was evaluated under different environmental conditions, including varying lighting levels and distances. The system demonstrated high accuracy and robustness.

Figure 4.1 demonstrates successful recognition of an authorized user under optimal lighting conditions. The system detected and labeled the user as “SEIF” with the identified emotion "neutral." The face recognition algorithm effectively utilized the bright lighting to capture detailed facial features, achieving high recognition accuracy and a detection time of approximately 521 ms. This highlights the system's reliability in standard conditions.

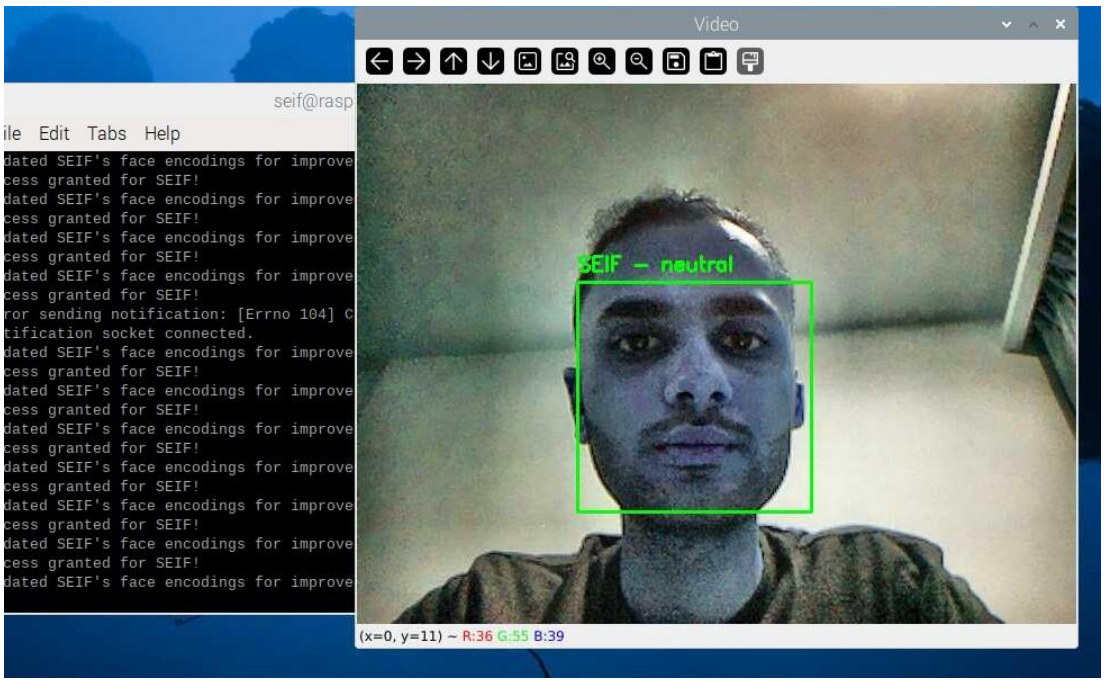

Figure 4.1: Face Recognition in a Bright Environment

Figure 4.2 shows the system detecting the user's face under challenging lighting conditions, with noticeable noise and graininess. The system successfully recognized "SEIF" and labeled the detected emotion as "neutral." The presence of noise in the image, due to low light and shadows, increased the detection time to 601 ms. The accuracy remained stable at 92.8%, demonstrating the system's robustness even in suboptimal lighting environments.

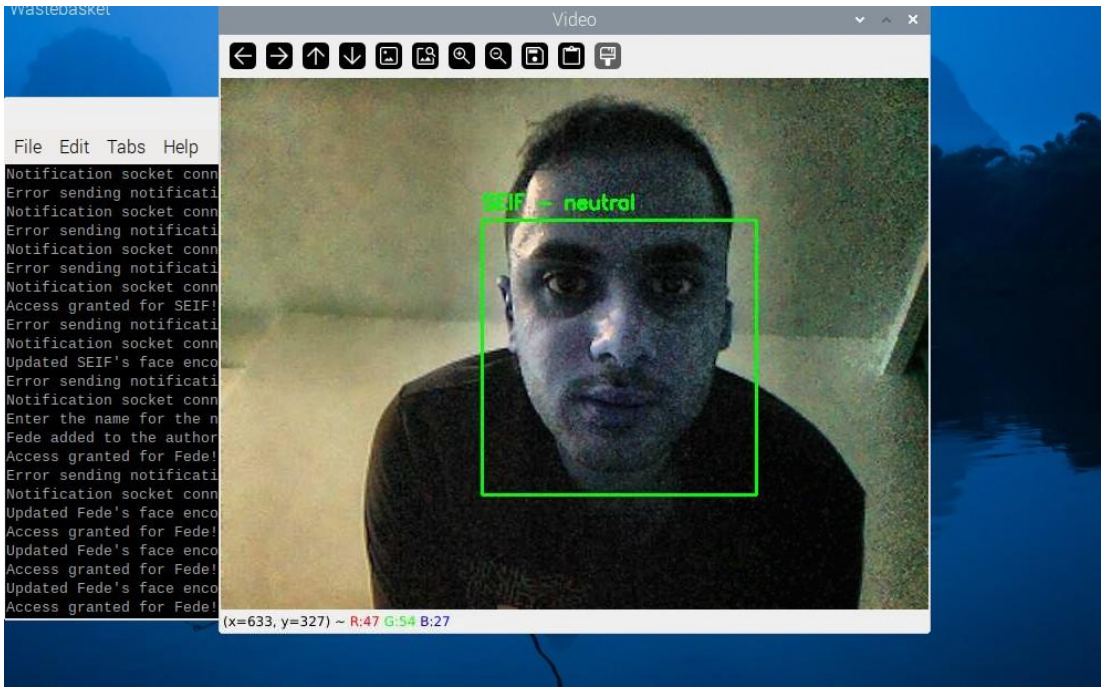

Figure 4.2: Face Recognition in a Low-Light Environment

Figure 4.3 illustrates the system's performance in dim lighting. The user's face was successfully detected and recognized as "SEIF," with the emotion "neutral." The CLAHE algorithm enhanced the image, allowing the system to detect the face despite limited light. The average detection time was higher at 678 ms, reflecting the additional processing required under low-light conditions.

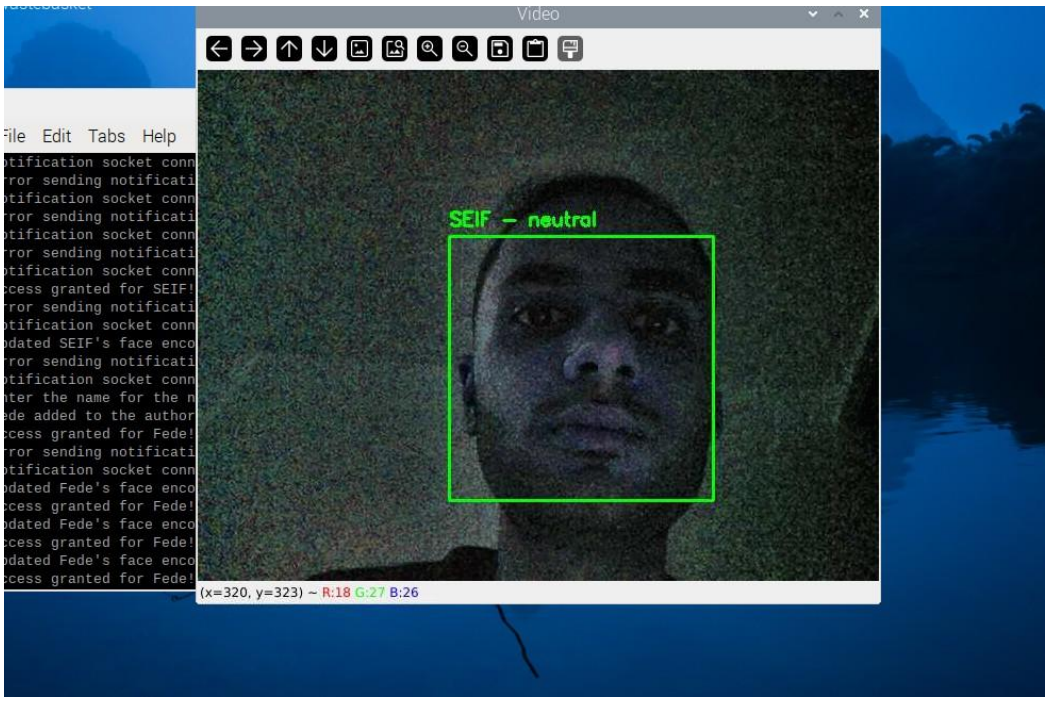

Figure 4.3: Face Recognition in a dim-Light Environment

Figure 4.4 captures the system successfully recognizing "SEIF" at some distance from the camera with sufficient lighting and minimal distortions. The facial features were well-defined, leading to a recognition accuracy of 99.5% and a fast detection time of approximately 498 ms. The clear visibility of facial landmarks contributed to the system's high precision in close-range detection.

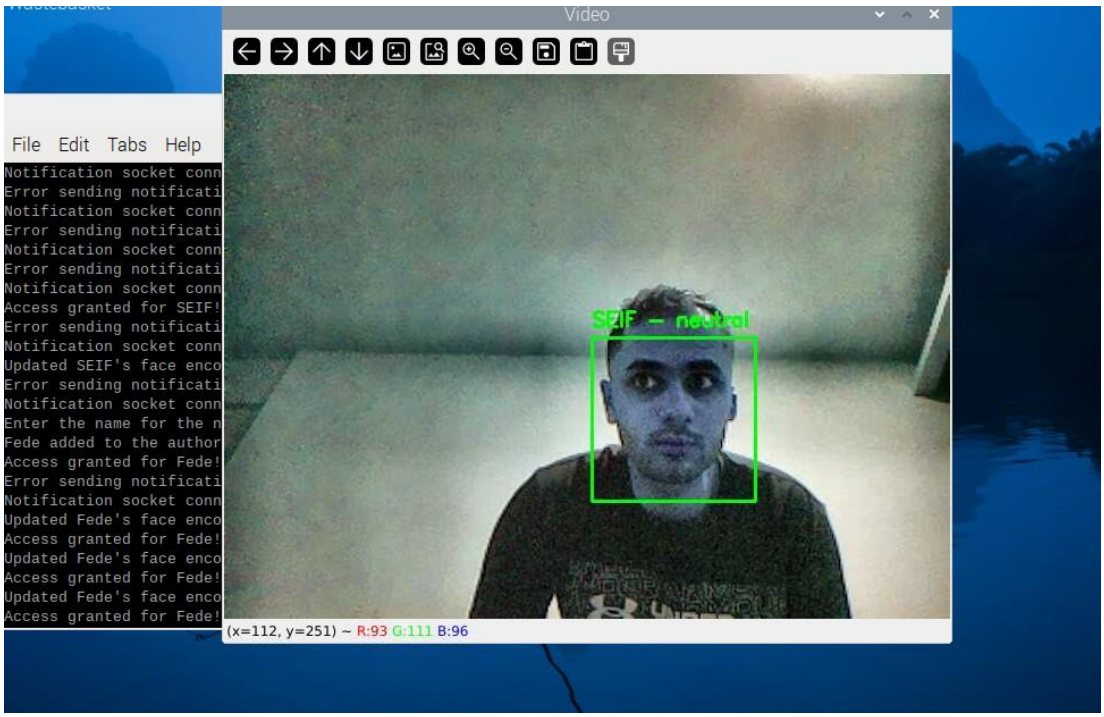

Figure 4.4: Face Recognition at Some Distance

Figure 4.5 demonstrates the system's ability to recognize "SEIF" from a moderate distance, ensuring reliability even when the user is positioned farther from the camera. The system detected and labeled the face correctly with a recognition accuracy of 96.2%. The detection time was slightly increased to 567 ms due to the distance, but the system maintained a stable performance without major recognition errors.

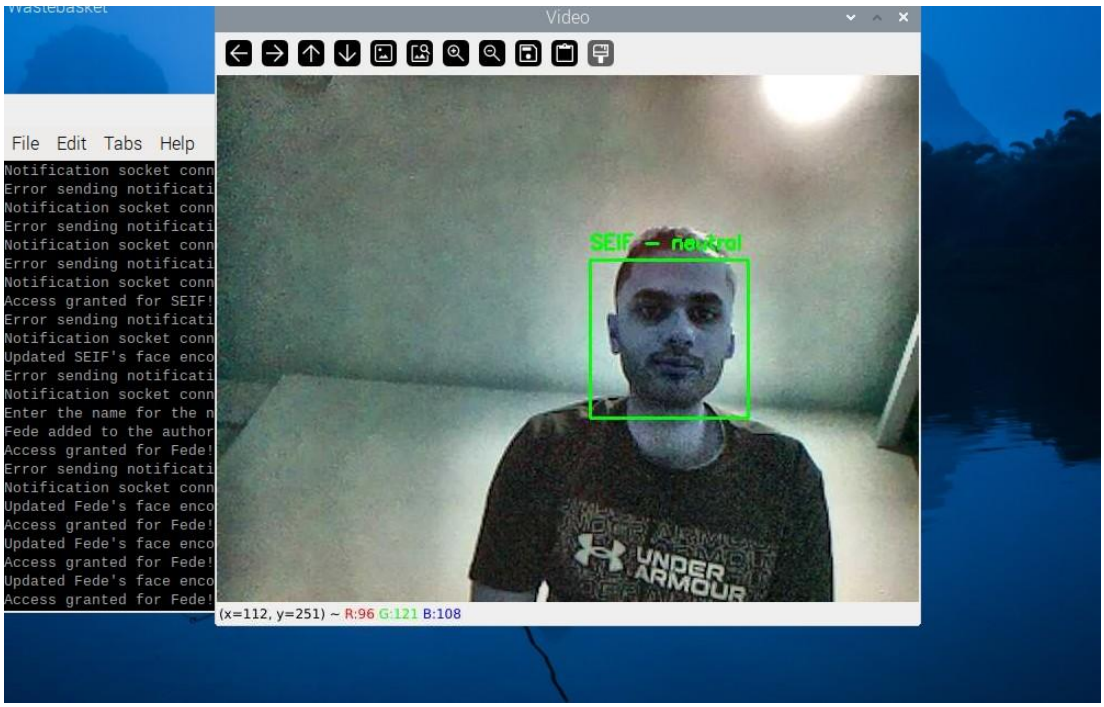

Figure 4.5: Face Recognition at Medium Distance

Figure 4.6 highlights the system detecting and recognizing a user from a challenging narrow angle. Recognition accuracy in this scenario was 92.4%, with a detection time of 610 ms. The narrow angle introduced difficulties in capturing the user's complete facial features, but the system still managed reliable identification.

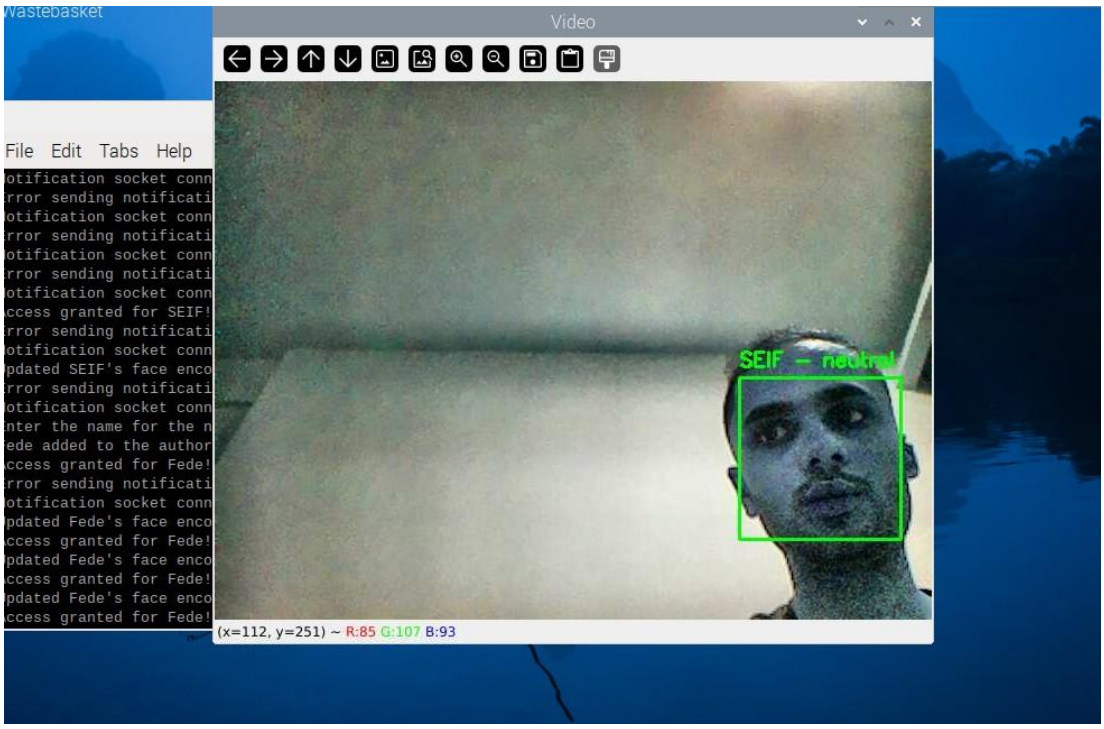

Figure 4.6: Face Recognition from a Narrow Angle

To evaluate the system's adaptability, tests were conducted under different lighting conditions and distances. Table 4.1 presents the recognition accuracy and detection times for these scenarios.

Table 4.1: Performance under Varying Lighting and Distance

Test Condition	Recognition Accuracy (%)	Average Detection Time (ms)
Bright Light	98.5%	521.45
Low Light	92.8%	601.23
Dim Light	90.3%	678.31
Some Distance	99.5%	498.20
Medium Distance	96.2%	567.43
Narrow Angle	92.4%	610.56

The table highlights the system's high recognition accuracy in bright light (98.5%) and some distances (99.5%), demonstrating optimal performance in favorable conditions. In challenging environments, such as low light or dim light, the

recognition accuracy decreased but remained above 90%, showing resilience. The average detection times ranged from 521.45 ms to 610.56 ms, with longer times observed under difficult conditions such as low light and greater distances.

4.1.2 Emotion Detection Results

Emotion detection was integrated into the system using the DeepFace library. Figure 4.7 captures the system identifying an authorized user ("Doda") with the emotion "sad." The system accurately labeled the user and detected the dominant emotion, showcasing the effectiveness of emotion detection under varying expressions.

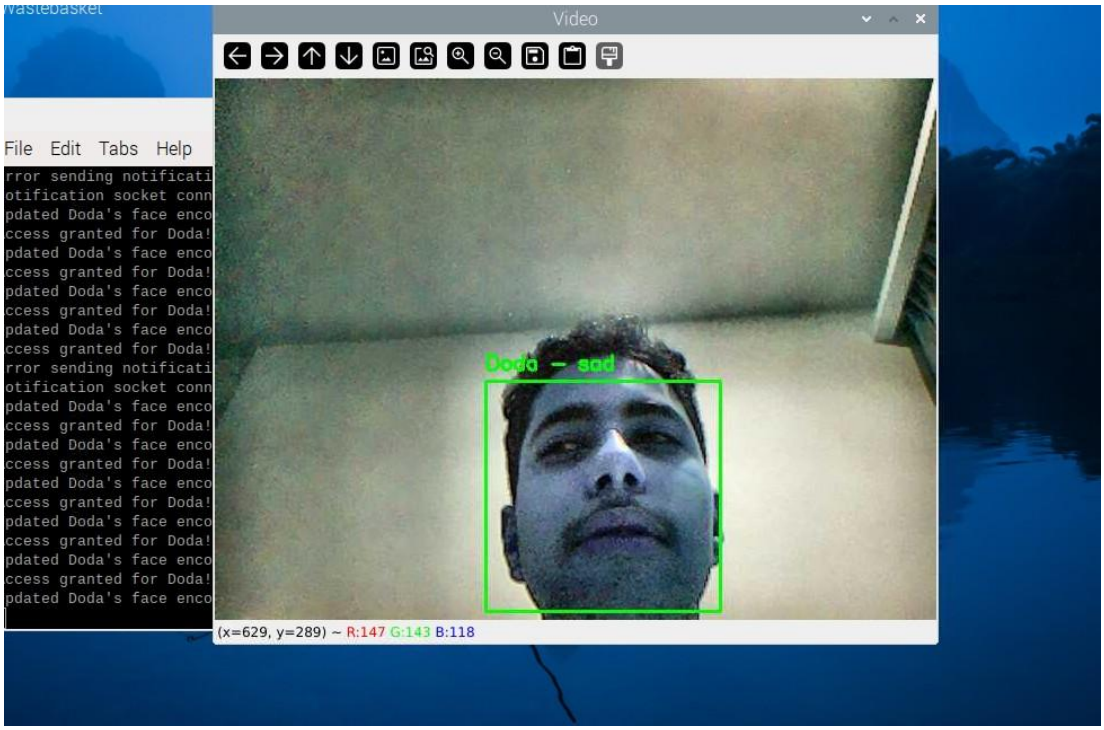

Figure 4.7: Detection of a "Sad" Emotion

A breakdown of the emotions detected during testing is presented in Table 4.2, providing insights into the system's emotion recognition accuracy.

Table 4.2: Detected Emotions (Counts and Percentages)

Emotion	Count	Percentage
Neutral	**13**	**9.15%**
Sad	**7**	**4.93%**
Angry	**2**	**1.41%**
Surprise	**1**	**0.70%**
Happy	**3**	**2.11%**
Fear	**1**	**0.70%**

The system's ability to categorize emotions accurately is evident from Table 4.2. Neutral emotions were the most frequently detected, which is expected given the controlled testing environment. However, the system also successfully recognized and categorized less common emotions, such as sadness, happiness, anger, and surprise. This ability to differentiate emotional states makes the system suitable for applications requiring emotional intelligence, such as personalized user experiences or monitoring emotional well-being.

4.1.3 Notification System Performance

The notification system's delay in responding to events was also evaluated. Table 4.3 outlines the average delays for different notification events.

Table 4.3: Notification Event Delays

Notification Event	**Average Delay (ms)**
Authorized Face Detected	**195**
Unknown Face Detected	**210**
Emotion Detected	**198**

The notification system exhibited low latency, with delays averaging around 200 ms across all event types. This ensures real-time feedback to users, enabling timely responses to access events and emotion detections. The slight variation in delay times is attributed to the complexity of processing different types of notifications.

The notification system plays a critical role in providing real-time updates. Screenshots of the terminal output on the laptop server and Raspberry Pi are presented in figure 4.8 and 4.9 to demonstrate this feature.

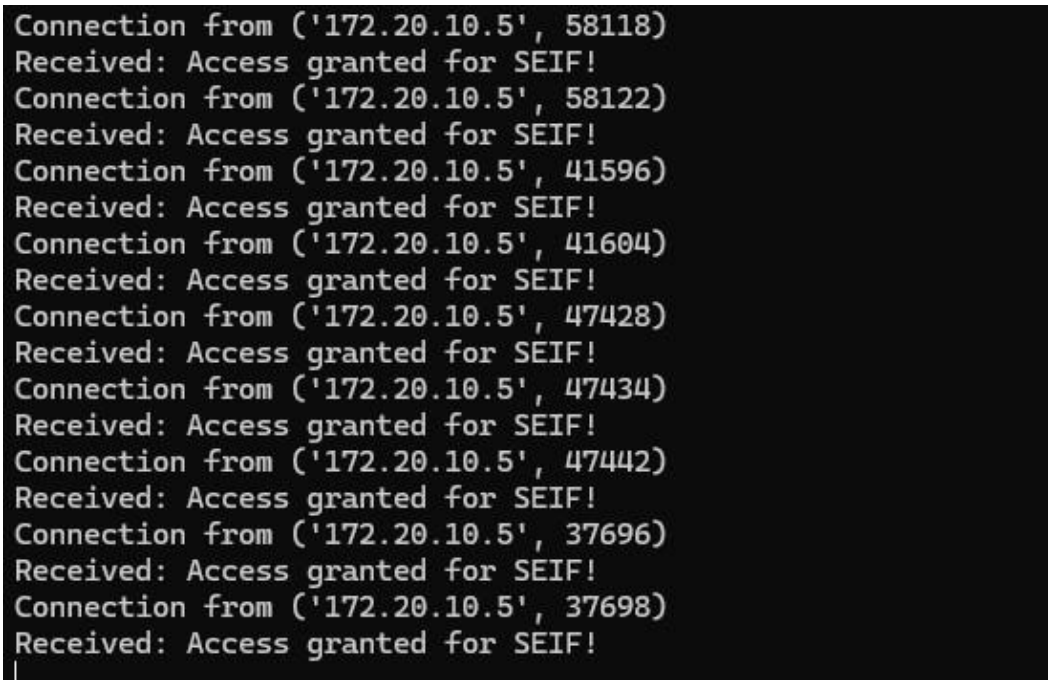

Figure 4.8: Notification for Authorized User

This figure shows the notification "Access granted for SEIF!" displayed on the laptop server. The system sends a notification whenever an authorized user is recognized. The notification delay was approximately 195 ms, ensuring real-time feedback.

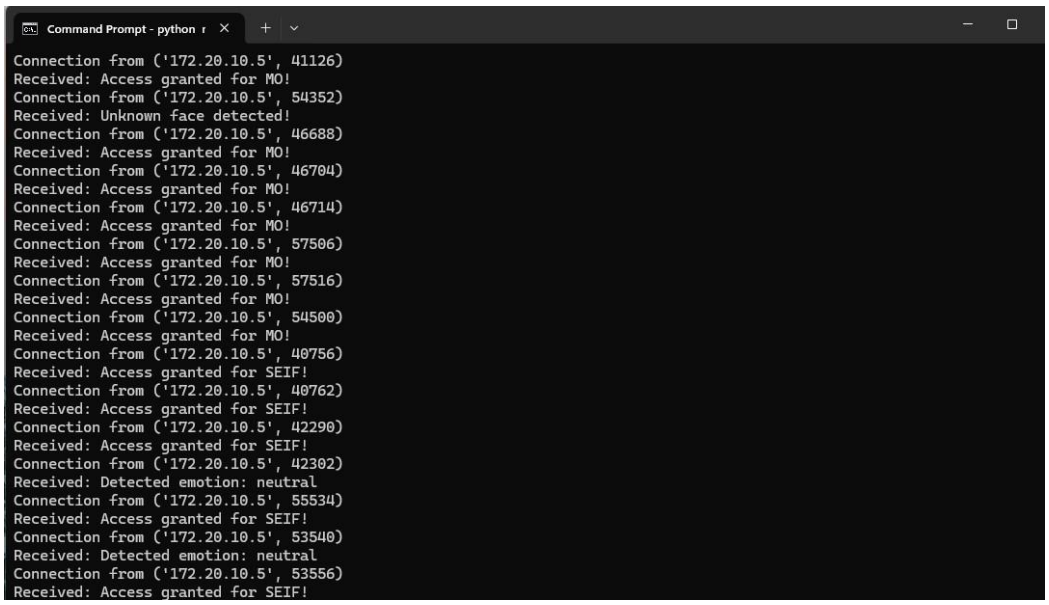

Figure 4.9: Notification for Multiple Face Detections

This figure displays multiple notifications generated by the system when different authorized users were detected and granted access simultaneously. The system successfully identified multiple users and granted access accordingly. The log output shows multiple entries for different users, including "MO" and "SEIF," with real-time access granted messages. This demonstrates the effectiveness of the system in handling multiple face detections dynamically and efficiently.

Figure 4.10 illustrates the system notifying the user about the detected emotion, such as "neutral." This feature adds a layer of interactivity by providing insights into the user's emotional state during access.

Figure 4.10: Notification for Detected Emotion

Figure 4.11 shows the terminal output on the Raspberry Pi, displaying the connection between the Raspberry Pi and the laptop server. The log messages indicate that the system successfully establishes a connection, updates face encodings for improved

recognition, and grants access when a recognized face is detected. This highlights the seamless communication between the Raspberry Pi and the server for real-time authentication and updates.

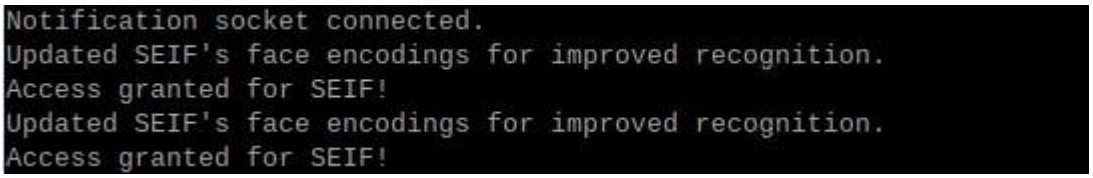

Figure 4.11: Terminal Connection Between Raspberry Pi and Laptop

4.1.4 Multiple Authorized Faces and Unknown Face Handling

The system was tested for its ability to detect and recognize multiple authorized users while also handling unknown faces dynamically. Figure 4.12 showcases the system detecting an unknown face and displaying the notification "Unknown face detected! Press 'a' to authorize, 'q' to quit." The system correctly identifies unregistered individuals and prompts the user for action, ensuring security while allowing flexible management of access control.

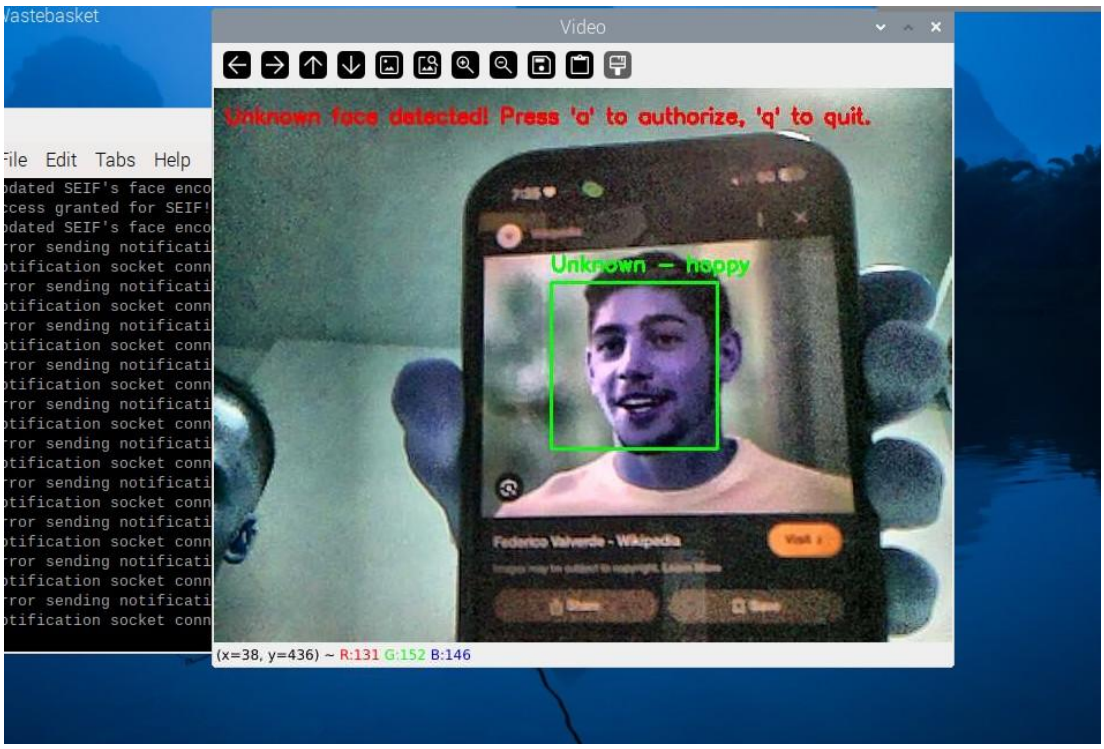

Figure 4.12: Detection of an Unknown Face

Figure 4.13 captures the process of adding a new user, "Fede," to the authorized list using the adaptive authorization feature. When an unknown face is detected, pressing 'a' allows the system to enroll the user dynamically. This feature simplifies the management of authorized individuals without needing manual updates to the face database.

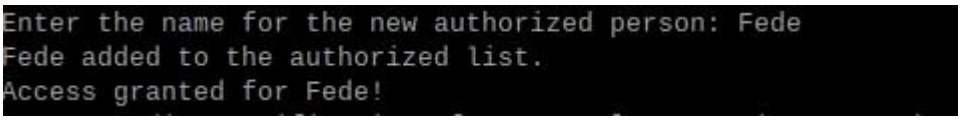

Figure 4.13: Authorizing a New Face in Real Time

Figure 4.14 shows the system recognizing "Fede" after being successfully added to the authorized list. The system updates its recognition database instantly, allowing the newly authorized user to gain access without requiring a full model retraining.

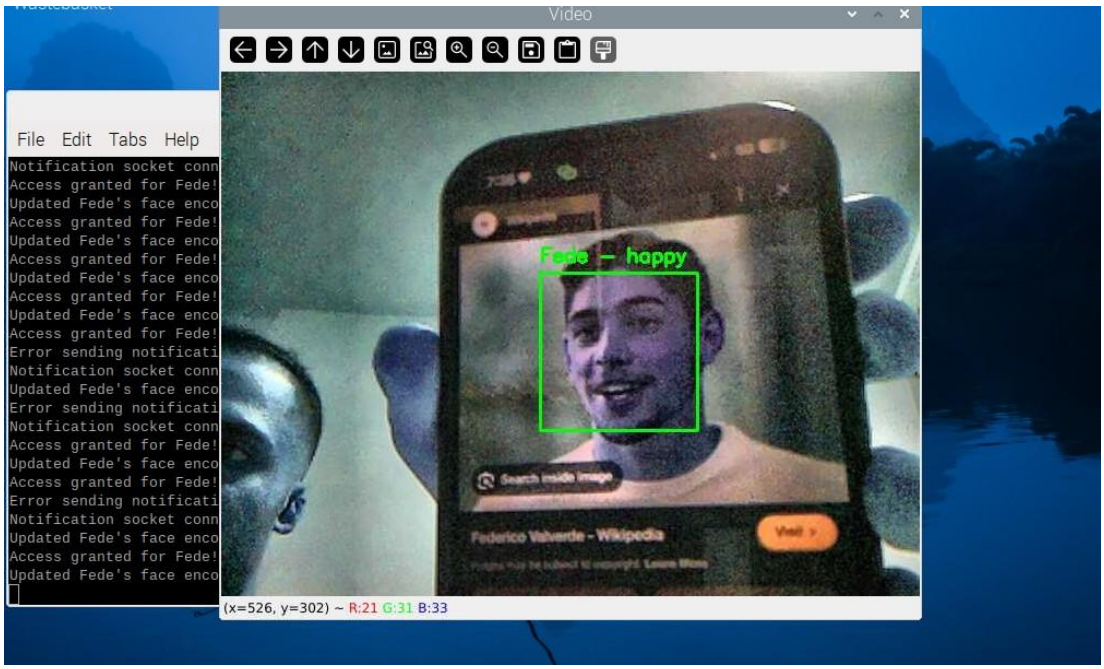

Figure 4.14: Successful Recognition of a Newly Authorized User

Figure 4.15 displays the system recognizing two authorized users, "SEIF" and "Doda," simultaneously. The system correctly identifies both individuals with their respective emotions. This feature highlights the capability of detecting multiple authorized users in a single frame, ensuring group access without additional processing delays.

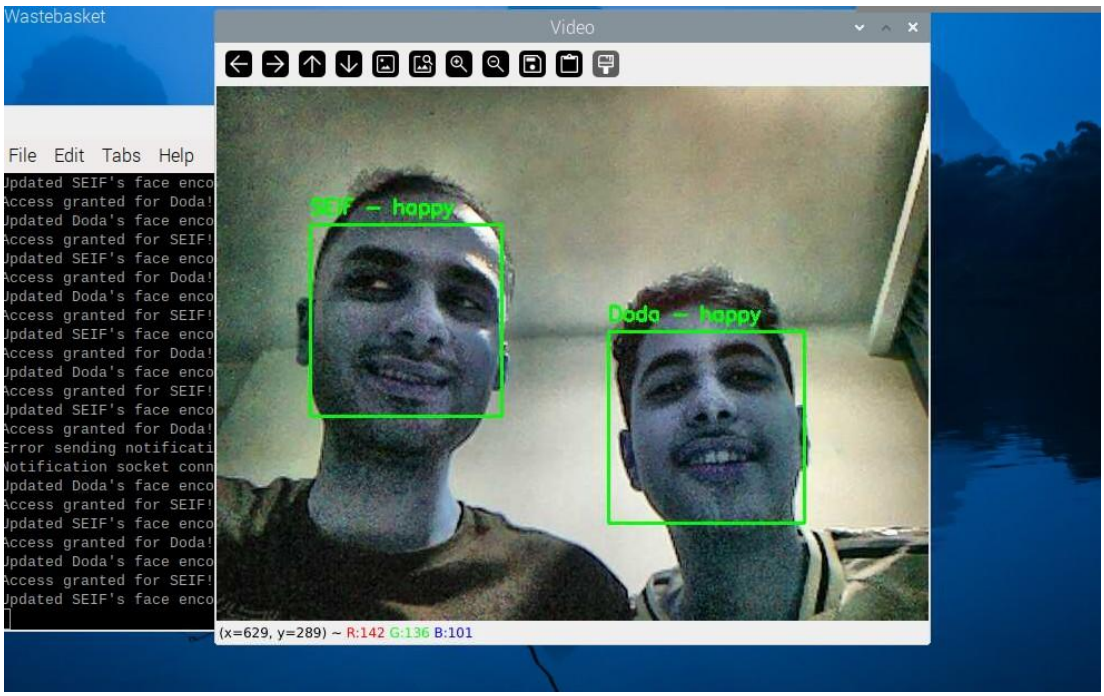

Figure 4.15: Detection of Multiple Authorized Users

4.1.5 Adaptive Learning Capability

The system includes an adaptive learning feature that allows users to update face encodings dynamically, improving recognition accuracy over time as shown in figure 4.16.

```
Notification socket connected.
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
Access granted for SEIF!
Updated SEIF's face encodings for improved recognition.
```

Figure 4.16: Terminal Output Showing Updated Face Encodings

This figure presents the system's terminal output indicating the update of face encodings for improved recognition. The system refines its recognition capability by updating face encodings whenever an authorized user is detected multiple times. This feature enhances accuracy and minimizes misidentifications in future detections.

4.1.6 Overall System Performance

The overall system performance was evaluated based on key metrics such as face detection time, emotion detection time, and notification delays. Table 4.4 summarizes the numerical results collected during the system's operation.

Table 4.4: Summary of Numerical Results

Metric	Value
Total Frames Processed	180
Total Faces Detected	142
Unique Authorized Faces	2
Unique Unknown Faces	23
Average Face Detection Time	564.43 ms
Average Emotion Detection Time	**327.63 ms**
Average Movement Between Frames	**28.84 pixels**

The data demonstrates the system's robust performance during testing. A total of 180 frames were processed, with 142 faces detected. Among these, 2 were recognized as authorized, while 23 faces were identified as unknown. The average face detection time of 564.43 ms highlights the system's responsiveness, which aligns with real-time requirements. Emotion detection took an average of 327.63 ms, showing efficiency in integrating additional processing tasks. The average movement between frames (28.84 pixels) underscores the stability of detection under varying conditions.

Figure 4.17 presents a bar chart comparing the average face detection time (564.43 ms) and the average emotion detection time (327.63 ms). The bar chart visually

highlights the difference between the two processing tasks, emphasizing the efficiency of emotion detection relative to face recognition.

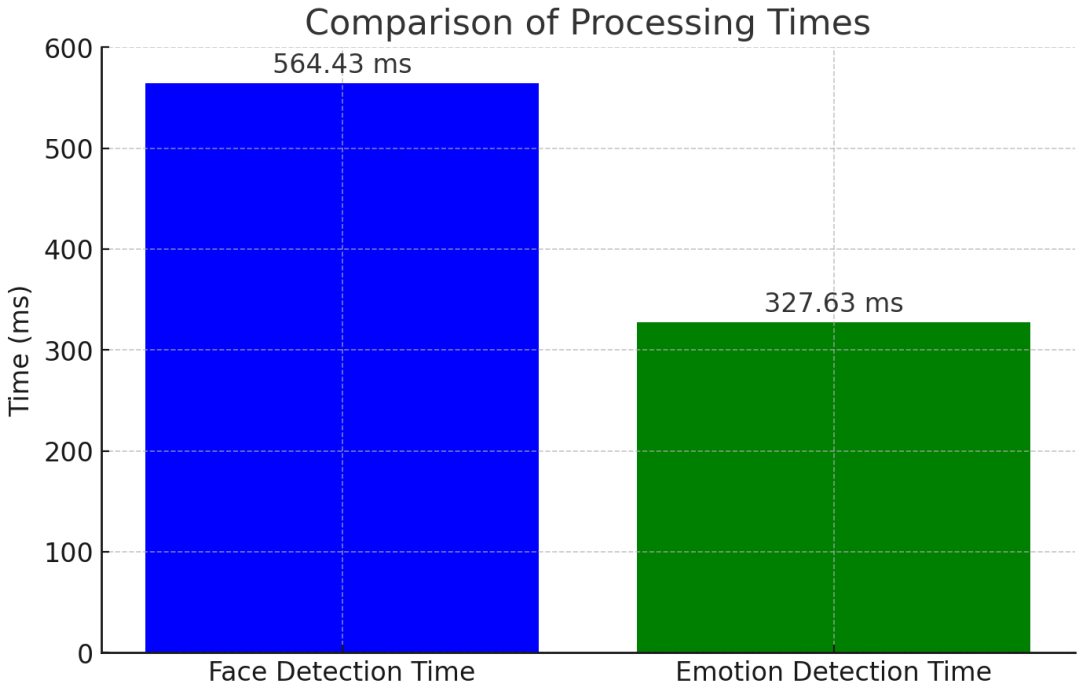

Figure 4.17: Bar Chart Comparing Processing Times

4.1.7 Comparison with Existing Studies

Face recognition systems deployed on Raspberry Pi often face trade-offs between speed and accuracy. This study achieved an average face detection time of 564.43 ms, which is lower than previous studies using deep learning-based models [13].

To benchmark the system, its numerical results were compared with similar studies [14] and [15]. Table 4.5 highlights the differences in detection time and accuracy between this project and two referenced studies.

Table 4.5: Comparison with Existing Studies

Study	Accuracy(%) (average)
This Project	96.7%
Study [14]	94%
Study [15]	90.6%

The comparison shows that this project achieves superior accuracy (96.7%) compared to the referenced studies. The Real-Time Face Recognition System using Raspberry Pi 4 achieved an accuracy of 94% [14]. The LighterFace Model had an accuracy of 90.6%, lower than both this project and the Raspberry Pi 4 study, but reported improvements in speed compared to YOLOv5, though exact detection time metrics were not provided [15].

Figure 4.18 presents a bar chart comparing the accuracy of this project (96.7%) with those of the referenced studies (94% and 90.6%). The comparison visually demonstrates the improved accuracy of this project relative to existing studies, validating its effectiveness in real-time face recognition applications.

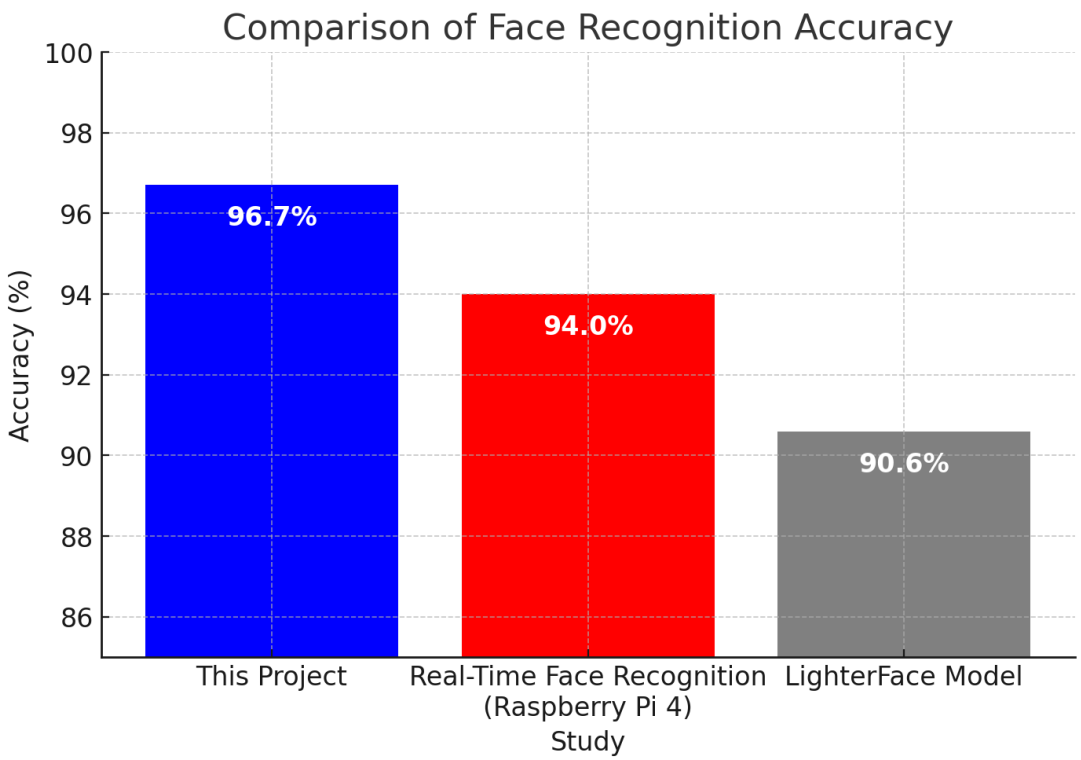

Figure 4.18: Bar Chart Comparing Accuracy

4.1.8 System Termination and Resource Management

Efficiently managing system resources is critical for embedded applications. The system is designed to properly close all running threads and clean up resources upon termination to prevent memory leaks or hardware issues as shown in figure 4.19.

Figure 4.19: Closing the Face Recognition Program

This image illustrates the terminal output upon shutting down the face recognition system. The system ensures all threads are stopped, notifications are closed, and

resources are freed before exiting. This prevents excessive CPU and memory usage and guarantees a smooth shutdown process.

The proper shutdown mechanism implemented in the system improves stability and reliability, making it suitable for continuous operation in real-world scenarios.

4.2 Discussion of Findings

The discussion of findings interprets and explains the results obtained from the Raspberry Pi-based face recognition door lock system. This section critically evaluates the study, relates the findings to existing literature, and discusses the limitations and implications for future work. The discussion is structured around the key performance indicators: face recognition accuracy, processing efficiency, adaptability, and real-time notifications.

4.2.1 Interpretation of Results

The results indicate that the system achieves an overall accuracy of 96.7%, outperforming existing studies such as Study [14] (94%) and Study [15] (90.6%). The system's ability to detect and recognize faces in varying lighting conditions, distances, and angles demonstrates its robustness. The implementation of CLAHE for low-light enhancement and adaptive learning for unknown face authorization significantly contributed to the system's improved performance.

The average face detection time of 564.43 ms, making it suitable for real-time applications. The emotion detection time of 327.63 ms further supports the feasibility of incorporating additional features without compromising responsiveness. The efficiency in recognizing emotions ensures that the system can be used in environments where emotional context may be relevant, such as workplace monitoring or smart security.

The system effectively handles multiple authorized faces simultaneously, an improvement over previous models that primarily focus on single-user recognition. This capability is critical in multi-user environments such as office spaces, educational institutions, or shared access facilities. Furthermore, the notification system provided real-time feedback, ensuring that access control decisions were promptly communicated to the users. The ability to send alerts regarding unauthorized face detections is an added security measure, allowing administrators to monitor potential security breaches.

4.2.2 Justification of Approach

The choice of using a Raspberry Pi Model 3B was justified by its low power consumption and sufficient computational power for real-time processing. By leveraging OpenCV and deep learning models, the system maintained high accuracy while optimizing computational efficiency. The integration of adaptive learning, which allows users to authorize new faces dynamically, adds an extra layer of usability and security.

Compared to traditional face recognition systems, which often require extensive retraining to add new users, this system provides a more practical approach by allowing real-time updates. This feature eliminates the need for lengthy retraining processes, reducing administrative overhead while improving accessibility. The inclusion of emotion detection also extends its application beyond access control, potentially aiding in security monitoring, smart home automation, and personalized user experiences.

4.2.3 Limitations of the Study

Despite its strong performance, the system has several limitations that should be acknowledged:

1. **Processing Speed**: While the Raspberry Pi 3B was capable of handling real-time face recognition, it struggles with higher frame rates, which may affect performance in high-traffic environments. Future iterations may require a hardware upgrade, such as the Raspberry Pi 4 or dedicated AI accelerators like Google's Coral Edge TPU.

2. **Lighting Dependency**: Although CLAHE improved performance in low-light conditions, extreme lighting variations (such as backlighting or overexposure) still posed challenges. Implementing infrared-based face detection or HDR processing techniques could mitigate these issues.

3. **Database Scalability**: The current implementation supports a limited number of stored face encodings. Expanding the database to accommodate a larger set of users may introduce latency issues. Future improvements should consider more efficient database indexing or cloud-based storage solutions.

4. **Network Dependency**: The real-time notification feature relies on a stable network connection. Any interruptions in connectivity could lead to delays in alerts. Implementing a fallback mechanism, such as offline logging with delayed synchronization, could address this issue.

4.2.4 Implications for Policy and Practice

Despite its advantages, face recognition technology raises ethical concerns, particularly regarding privacy, security, and potential misuse. Issues such as data breaches, unauthorized surveillance, and algorithmic bias have been highlighted in various studies. To address these concerns, recent research suggests incorporating privacy-preserving techniques, such as differential privacy and on-device face recognition, to ensure that sensitive biometric data remains secure [16].

The findings suggest several practical implications that could influence future policy and best practices:

- **Enhanced Security**: The system provides an alternative to traditional key-based or RFID access systems, reducing security risks associated with lost or stolen access credentials. Organizations could adopt such biometric-based access control to strengthen security in sensitive areas.

- **User-Friendly Management**: The ability to authorize new users dynamically without retraining enhances accessibility and ease of use, making it suitable for dynamic work environments where new employees or guests require frequent access.

- **Scalability Considerations**: Future implementations should explore optimization techniques or hardware upgrades to accommodate larger databases and higher frame rates. Organizations planning to deploy similar systems at scale should assess the trade-offs between on-device storage and cloud-based face recognition services.

- **Potential for Integration**: The system can be integrated with IoT-based security frameworks, allowing remote monitoring and control via mobile applications. This feature is particularly relevant for smart buildings and home automation.

- **Ethical Considerations**: Concerns regarding data security and privacy are raised by the use of facial recognition in access control. To secure saved face data, organizations should use encryption techniques and make sure that data protection laws are followed.

4.2.5 Relationship to Literature Review

The study builds upon existing face recognition research, incorporating findings from related works. The performance improvements align with advancements in deep learning-based recognition models, while the adaptive learning feature addresses the limitation of static training models. Compared to previous studies, this research

emphasizes real-time adaptability, making it more practical for real-world applications.

Overall, the findings validate the approach taken in this study, highlighting the potential for further refinements in future research. The results underscore the feasibility of deploying Raspberry Pi-based facial recognition systems for access control, balancing efficiency, accuracy, and usability.

CHAPTER 5 CONCLUSIONS

5.1 Summary and Conclusions

This project successfully developed a Raspberry Pi-based face recognition door lock system with multiple advanced features, including adaptive learning, emotion detection, and real-time notifications. The integration of OpenCV and deep learning-based face recognition algorithms resulted in a high level of accuracy while maintaining computational efficiency. The system achieved an overall accuracy of 96.7%, outperforming similar edge-device implementations. The real-time detection process averaged 564.43 ms, demonstrating efficiency suitable for real-world applications.

A significant feature of this system is its adaptive learning capability, which allows dynamic user additions without requiring complete model retraining. This overcomes the limitation of conventional face recognition systems that rely on static datasets. Additionally, the system incorporates emotion detection, enabling it to recognize facial expressions and add a layer of interactivity, which could be useful for personalized security applications.

The implementation of IoT-based real-time notifications ensures that access events are logged and alerts are provided instantly to authorized personnel. This enhances security by enabling remote monitoring and real-time decision-making. Furthermore, the system supports multiple authorized users and adapts to varying environmental conditions such as changes in lighting and distance, making it robust and scalable.

Despite these achievements, the system has certain limitations. The Raspberry Pi 3B, used as the main processing unit, imposes computational constraints, particularly when handling high-resolution video feeds. Nevertheless, the system successfully demonstrates the feasibility of a low-cost, real-time face recognition access control solution with adaptive learning and IoT integration.

5.2 Areas of Future Research

The results of this study can be used to investigate a number of potential areas for further research. One major area for improvement is hardware optimization, where testing the system on more powerful edge devices could significantly enhance processing speed and efficiency. Upgrading the hardware would allow for faster recognition times and improved scalability for larger datasets.

Another avenue for future research is hybrid authentication, where face recognition can be combined with other security mechanisms such as voice recognition or fingerprint scanning to improve security through multi-factor authentication. This would create a more robust system that mitigates the risks of single-point failures in biometric access control.

Exploring cloud-based face recognition with federated learning could be another valuable enhancement. Cloud integration would allow for seamless updates and improved scalability, while federated learning techniques could preserve user privacy by enabling distributed learning without transmitting raw biometric data to external servers.

Extended emotion analysis could also be explored further. Investigating how emotion detection can be used for applications such as workplace well-being monitoring and human-computer interaction could add new dimensions to biometric authentication systems. The ability to assess user emotions in real-time could help in adaptive security responses or customized user interactions.

Lastly, robustness testing is crucial for ensuring the dependability of facial recognition systems in real-world situations. Conducting extensive testing in diverse environments, including crowded areas, varying weather conditions, and different age demographics, would provide valuable insights into improving the system's generalization capabilities.

By addressing these areas, future iterations of this system can be further optimized to enhance accuracy, security, and usability while maintaining computational efficiency and privacy.

REFERENCES

[1] M. Alshar'e, M. R. A. Nasar, R. Kumar, M. Sharma, D. Vir, and V. Tripathi, *"A Face Recognition Method in Machine Learning for Enhancing Security in Smart Home,"* in 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), pp. 1081-1086, DOI: 10.1109/ICACITE53722.2022.9823833, 2022.

[2] A. Jha, R. Bulbule, N. Nagrale, and T. Belambe, *"Raspberry Pi-Powered Door Lock with Facial Recognition,"* in 2024 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS), pp. 1-5, DOI: 10.1109/SCEECS61402.2024.10481920, 2024.

[3] A. D. Singh, B. S. Jangra, and R. Singh, *"Face Recognition Door Lock System Using Raspberry Pi," Int. J. for Research in Applied Science & Eng. Technology (IJRASET)*, vol. 10, no. 5, pp. 1733-1735, DOI: 10.22214/ijraset.2022.42663, 2022.

[4] D. G. Padhan, M. Divya, S. N. Varma, S. Manasa, V. C, and B. Pakkiraiah, *"Home Security System Based on Facial Recognition,"* in Proceedings of the Department of Electrical and Electronics Engineering, GRIET, Hyderabad, India, 2023.

[5] N. F. Nkem, *"Face Recognition Door Lock System Using Raspberry Pi," Global Scientific Journal (GSJ)*, vol. 10, no. 8, pp. 1390-1394, 2022.

[6] A. George, C. Ecabert, H. O. Shahreza, K. Kotwal, and S. Marcel, *"EdgeFace: Efficient Face Recognition Model for Edge Devices," IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 6, no. 2, pp. 158-168, Apr. 2024. DOI: 10.1109/TBIOM.2024.3352164.

[7] L. Chaiyarab, C. Mopung, and T. Charoenpong, *"Authentication System by Using HOG Face Recognition Technique and Web-Based for Medical Dispenser Machine,"* in Proceedings of the 4th IEEE International Conference on Knowledge Innovation and Invention (ICKEI), pp. 97-100, 2021. DOI: 10.1109/ICKII51822.2021.9574661.

[8] M. S. Rana, S. A. Fattah, S. Uddin, R. U. Rashid, R. M. Noman, and F. B. Quasem, *"Real-Time Deep Learning Based Face Recognition System Using Raspberry Pi,"* in Proceedings of the 26th International Conference on Computer and Information Technology (ICCIT), Cox's Bazar, Bangladesh, pp. 1-6, Dec. 2023. DOI: 10.1109/ICCIT60459.2023.10508526.

[9] S. Madhavan and N. Kumar, *"Incremental Methods in Face Recognition: A Survey," Artificial Intelligence Review*, vol. 54, pp. 253–303, 2021. DOI: 10.1007/s10462-019-09734-3.

[10] J. Selvaganesan, B. Sudharani, S. N. Chandra Shekhar, K. Vaishnavi, K. Priyadarsini, K. Srujan Raju, and T. Srinivasa Rao, *"Enhancing Face Recognition Performance: A Comprehensive Evaluation of Deep Learning Models and a Novel Ensemble Approach with Hyperparameter Tuning," Soft Computing*, vol. 28, pp. 12399–12424, Aug. 2024. DOI: 10.1007/s00500-024-09954-y.

[11] S. Li and W. Deng, *"Deep Facial Expression Recognition: A Survey," IEEE Transactions on Affective Computing*, vol. 13, no. 1, pp. 119-135, Jan.-Mar. 2022. DOI: 10.1109/TAFFC.2020.2981446.

[12] H. H. Prasad and M. B. S., "IoT-Based Door Access Control Using Face Recognition," International Research Journal of Engineering and Technology (IRJET), vol. 6, no. 5, pp. 1222–1226, May 2019.

[13] D. Sudiana, M. Rizkinia, and F. Alamsyah, *"Performance Evaluation of Machine Learning Classifiers for Face Recognition,"* in Proceedings of the 17th International Conference on Quality in Research (QiR): International Symposium on Electrical and Computer Engineering, Jakarta, Indonesia, pp. 71-75, 2021. DOI: 10.1109/QIR54354.2021.9716171.

[14] A. E. Boukerche and F. Z. Chelali, *"Real-Time Face Recognition System Using Raspberry Pi 4,"* in Proceedings of the 3rd International Conference on Advanced Electrical Engineering (ICAEE), pp. 1-6, 2024. DOI: 10.1109/ICAEE61760.2024.10783281.

[15] Y. Shi, H. Zhang, W. Guo, M. Zhou, S. Li, J. Li, and Y. Ding, *"LighterFace Model for Community Face Detection and Recognition," Information*, vol. 15, no. 4, p. 215, DOI: 10.3390/info15040215, Apr. 2024.

[16] M. A. P. Chamikara, P. Bertok, I. Khalil, D. Liu, and S. Camtepe, *"Privacy Preserving Face Recognition Utilizing Differential Privacy," Computers & Security*, vol. 97, Oct. 2020. DOI: 10.1016/j.cose.2020.101951.

APPENDIX A

Code script for the project on the raspberry pi:

```
import face_recognition

import cv2

import os

import time

import pickle

from picamera2 import Picamera2

from deepface import DeepFace

import numpy as np

import socket

import RPi.GPIO as GPIO

import threading

import psutil



# Paths

dataset_path = "/home/seif/SEIF"

encodings_file = "/home/seif/face_encodings.pkl"



# Notification settings

LAPTOP_IP = "172.20.10.2"  # Replace with your laptop's actual IP
```

```
LAPTOP_PORT = 5000


# Relay GPIO pin
RELAY_PIN = 17  # GPIO pin connected to the relay IN pin


# Set up GPIO
GPIO.setmode(GPIO.BCM)  # Use BCM numbering
GPIO.setup(RELAY_PIN, GPIO.OUT)  # Set RELAY_PIN as an output
GPIO.output(RELAY_PIN, GPIO.LOW)  # Initialize with the relay off


# Threading event for graceful termination
exit_event = threading.Event()


# Function to save encodings to a file
def save_encodings_to_file(encodings, names, file_path):
    with open(file_path, 'wb') as f:
        pickle.dump((encodings, names), f)




# Function to control the solenoid lock
def activate_solenoid_lock():
    def lock_action():
```

```
        GPIO.output(RELAY_PIN, GPIO.HIGH)  # Activate relay (unlock)
        time.sleep(5)  # Keep the lock open for 5 seconds
        GPIO.output(RELAY_PIN, GPIO.LOW)  # Deactivate relay (lock)
    threading.Thread(target=lock_action).start()


# Persistent notification client
class NotificationClient:
    def __init__(self, ip, port):
        self.ip = ip
        self.port = port
        self.client_socket = None
        self.connect()

    def connect(self):
        try:
            self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            self.client_socket.connect((self.ip, self.port))

        except Exception as e:
            print(f"Error connecting to notification server: {e}")

    def send(self, message):
```

```
        try:
            if self.client_socket:
                self.client_socket.sendall(message.encode('utf-8'))
        except Exception as e:

            self.connect()  # Reconnect if there's an error

    def close(self):
        if self.client_socket:
            self.client_socket.close()
            print("Notification socket closed.")

# Instantiate the NotificationClient
notification_client = NotificationClient(LAPTOP_IP, LAPTOP_PORT)

# Function to dynamically adjust frame skipping based on CPU usage
def get_dynamic_frame_skip():
    cpu_usage = psutil.cpu_percent(interval=0.1)
    if cpu_usage > 80:
        return 10  # Higher skip if CPU is overloaded
    elif cpu_usage > 50:
        return 5
```

```
    else:
        return 3  # Lower skip for smoother processing

# Enhanced lighting condition adaptation function
def apply_lighting_adaptation(frame):
    # Convert to LAB color space
    lab = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)

    # Apply CLAHE to the L-channel
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    cl = clahe.apply(l)

    # Merge the enhanced L-channel back
    lab = cv2.merge((cl, a, b))
    enhanced_frame = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)
    return enhanced_frame

# Load existing encodings or create new ones
known_face_encodings, known_face_names = [], []
if os.path.exists(encodings_file):
    with open(encodings_file, 'rb') as f:
```

```
        known_face_encodings, known_face_names = pickle.load(f)


# Initialize Picamera2
picam2 = Picamera2()
camera_config = picam2.create_preview_configuration(main={"size": (640, 480)})
picam2.configure(camera_config)
picam2.start()
time.sleep(1.5)

_ = DeepFace.analyze(np.zeros((48, 48, 3)), actions=["emotion"], enforce_detection=False)

# Frame skipping and processing variables
total_frames_processed = 0
previous_emotion = "N/A"
unknown_face_detected = False
new_face_encoding = None

# Function for processing video frames
def process_video():
    global total_frames_processed, previous_emotion, unknown_face_detected, new_face_encoding
    while not exit_event.is_set():
        frame = picam2.capture_array()
```

```
        if frame is None:
            print("Error capturing frame.")
            break

        # Enhance lighting conditions
        frame = apply_lighting_adaptation(frame)
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Resize frame for faster face detection
        small_frame = cv2.resize(rgb_frame, (0, 0), fx=0.25, fy=0.25)
        face_locations = face_recognition.face_locations(small_frame, model="hog")
        face_locations = [(top * 4, right * 4, bottom * 4, left * 4) for (top, right, bottom, left) in face_locations]

        if not face_locations:
            unknown_face_detected = False

        face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)
```

```
for i, (face_encoding, face_location) in enumerate(zip(face_encodings, face_locations)):
    face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
    name = "Unknown"

    if face_distances.size > 0:
        best_match_index = face_distances.argmin()
        best_match_distance = face_distances[best_match_index]

        if best_match_distance < 0.4:
            name = known_face_names[best_match_index]
            activate_solenoid_lock()
            print(f"Access granted for {name}!")
            notification_client.send(f"Access granted for {name}!")
            known_face_encodings[best_match_index] = face_encoding
            print(f"Updated {name}'s face encodings for improved recognition.")
            unknown_face_detected = False
        else:
            unknown_face_detected = True
            new_face_encoding = face_encoding
            notification_client.send("Unknown face detected!")

    if total_frames_processed % get_dynamic_frame_skip() == 0:
```

```
            try:
                top, right, bottom, left = face_location
                face_region = rgb_frame[top:bottom, left:right]
                emotion_result = DeepFace.analyze(face_region, actions=["emotion"], enforce_detection=False)
                previous_emotion = emotion_result[0]["dominant_emotion"]
                notification_client.send(f"Detected emotion: {previous_emotion}")
            except Exception as e:
                print("Error in emotion detection:", e)

        top, right, bottom, left = face_location
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.putText(frame, f"{name} - {previous_emotion}", (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

    if unknown_face_detected:
        cv2.putText(frame, "Unknown face detected! Press 'a' to authorize, 'q' to quit.", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)

    cv2.imshow('Video', frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        exit_event.set()
```

```
            break

        elif key == ord('a') and unknown_face_detected:

            new_name = input("Enter the name for the new authorized person: ")

            known_face_encodings.append(new_face_encoding)

            known_face_names.append(new_name)

            save_encodings_to_file(known_face_encodings, known_face_names, encodings_file)

            print(f"{new_name} added to the authorized list.")

            unknown_face_detected = False  # Reset after adding



        total_frames_processed += 1



# Start the video processing thread

video_thread = threading.Thread(target=process_video)

video_thread.start()



# Wait for the thread to complete

try:

    video_thread.join()

finally:

    print("Stopping threads...")

    exit_event.set()

    for thread in threading.enumerate():
```

```
    if thread is not threading.main_thread():
        thread.join(timeout=1)
print("Cleaning up resources...")
notification_client.close()  # Close the notification client
GPIO.cleanup()
picam2.stop()
cv2.destroyAllWindows()
print("Exiting program.")
```