ASHESI UNIVERSITY COLLEGE

DESIGN AND IMPLEMENTATION OF FACE RECOGNITION-BASED DOOR ACCESS CONTROL SYSTEM

CAPSTONE PROJECT

B.Sc. Computer Engineering

Alain Honore Kubwayo

2020

ASHESI UNIVERSITY COLLEGE

DESIGN AND IMPLEMENTATION OF FACE RECOGITION-BASED DOOR ACCESS CONTROL SYSTEM

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Alain Honore Kubwayo

2020

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

Alain Honore Kubwayo

……

Candidate's Name:

Alain Honore Kubwayo

……

Date:

May 30, 2020

……

I hereby declare that preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University College.

Supervisor's Signature:

……

Supervisor's Name:

……

Date:

……

Acknowledgements

To my supervisor, Dr Stephen K. Armah whose encouragement and academic advice helped me undertake this project.

Abstract

With recent dramatic development in the field of artificial intelligence (AI), smart access control has become crucial part of our modern everyday lives. This paper presents a door security system designed to prevent trespassing in a highly secure areas like home environment. The implemented system is cheaper and more reliable for intruder detection and door security. The door access is solely based on face recognition to ensure that only authorized individuals go through the door. The principle on which face recognition works is that the sysadmin defines an individual or a group of individuals who are allowed to enter the area, a room or a building, through a door, a gate, or any physical barrier. Thus, the access is limited to these individuals.

Faces of authorized individuals are captured, stored, and trained by the system, and a real-time face is captured and matched against the ones of authorized individuals to identify the individual gaining access through the door after which the access is granted if that individual is authorized and denied otherwise. Haar-based cascade classifier was used for face detection while Local Binary Pattern Histograms (LBPH) algorithm was used for face recognition. 150 faces of individual were captured and trained after which real-time faces were tested on the system to determine the accuracy. The implemented recognition system achieved a recognition or accuracy rate of 88.6%. Notification has been achieved, using Simple Mail Transfer Protocol (SMTP), by sending emails to the sysadmin in case of any successful or unsuccessful attempt to gain access through the door. Python's Open Source Computer Vision (OpenCV) library was used for face recognition. The implemented system is cheap, user-friendly, and reliable.

Contents

Chapter 1: Introduction

1.1 Background

In the past few years, biometric technology has evolved as the most promising choice for recognizing individuals. The technology embodies different techniques, such as DNA matching, signature, fingerprint, hand, face, iris, and retina recognition, in order to determine individuals' identities. Usually, these techniques analyze individual's physiological and behavioral characteristics to show their identity [1]. With the capability to recognize individuals, these techniques find applications in many sectors one of which is security, especially for granting access to secure locations or resource.

1.2 Background

South African Government News Agency reported that 2018/2019 period left 1.3 million incidents of housebreaking. This is an average of 3,561 incidents per day. Hence while looking to reinforce reliable home security, biometrics is a lot more secure compared to similar techniques that look to authenticate individuals and grant them access to locations by using smart cards, plastic cards, tokens, passwords, PINs, or even keys. While magnetic cards can become corrupted and unreadable, it is easy to forget, steal, or guess passwords and PINs the same way it is to misplace, duplicate, or forget cards, keys, and tokens. Consequently, biometrics is a suitable and reliable solution for ensuring highly secure access control since individual's biological traits cannot be misplaced, forgotten, stolen or forged [1]. Unlike other forms of biometric solutions however, face recognition is contactless. meaning that the subject does not physically interact with the system, which makes it one of the more passive, user-friendly, and less intrusive forms of biometrics. Jaiswal et al. defines a contactless biometric as one that does not require undesirable contact in order to extract the required data sample of the biological

characteristic and for that reason, it is most adaptable of variable ability levels. Face recognition technology helps in identifying or verifying individuals by comparing the captured input faces with the face images stored in the database [4]. Saini and Rana [3] have argued that the system can identify an individual from a massive crowd when set up in open public areas like airports, streets, churches and so on. Similarly, Zeenathunisa et al [4] reveals that face recognition has high social acceptability in comparison to other biometric applications such as iris and fingerprint recognition. Moreover, the system's ability to continuously track individuals makes surveillance and incident monitoring possible while other methods like fingerprint, iris, retina, and signature among others cannot offer the same benefits. Face recognition has countless applications particularly in surveillance, law enforcement, security, automated screenings at the airports, shopping malls, hotels, and border crossing among others [4]. Today, most computers and smartphones rely on face recognition to ensure that only authorized owners can unlock and use them. With this technology, one's face is the key unlike other systems that use physical objects, like RFID tags, keys, passwords, biometric fingerprint, that are prone to vulnerability.

Access control necessitates that access to a physical place, or a resource, is only granted to a group of people and restricted to the rest who are considered intruders by the system. In other words, access control selectively determines who can have access to a physical location such as our homes and offices, so far as physical security is concerned, as well as our resources, varying from unlocking our smartphones to opening our mailbox and so on, so far as information security is concerned. As a result, face recognition technology is portentous as it first detects if there is a face in the image or a video (face detection) and then goes on to recognize who the face belongs to (face recognition), if there is a face in the image that is. If the individual trying to gain access to the physical

location or a resource is recognized by the system, then, that individual will have access otherwise, the entry will be denied, and the necessary steps will be taken thereafter. In fact, the system tightens security of our offices, homes or any other building or secure rooms within these buildings thereby exposing the vulnerabilities presented by using keys, fobs, and cards. The system's ability to keep record of all successful and unsuccessful attempts to gain access and its real-time alerts all the system owners to have total control of the system, thus, providing them with security, comfort and convenience. It guarantees that only authorized individuals can gain access to the facility while requiring far less personal information than other forms of identification. Inspired by the rapid advancement of Artificial Intelligence in recent years and the security vulnerabilities of most existing access control techniques, this paper presents the findings from the design and implementation of a cheap yet robust and energy efficient access control system based on face recognition placing the need for safety and security of people's homes, offices, and similar places at its core.

1.3 Problem Definition

From the numbers reported, it is clear that South African people need reliable door security system to deal with the persisting housebreaking. This project was proposed and developed to come up with a reliable door security system that would assist South African in dealing with the persisting housebreaking problem.

1.4 Objectives

The following are the set objectives of the project:

- System should recognize individuals before granting them access
- System should operate on a threshold of at least 80% recognition rate
- System should notify the house owner of any successful and unsuccessful attempt

1.5 Scope

The project seeks to design and implement the home door security system that is based on face recognition. Here, the access is restricted to authorized individuals. The system has a buzzer module for beeping or making noise in case of an intruder or in case an unauthorized individual tries to access the door and system does not recognize them. It has a face recognition module that has face detection and recognition as stages through which the real-time face is recognized. It also has a locking mechanism responsible for automatically unlocking the door in when authorized individual is given access. The system is expected to operate at a threshold of 80% recognition rate. Success will be measured when the system reaches a recognition rate of at least 80% and is able to sound a buzzer and send notifications to the house owner of any successful and unsuccessful attempts.

The rest of the paper is organized as follows: the literature review is presented in chapter 2, design and methodology in chapter 3, followed by implementation in chapter 4, and finally, conclusion and recommendation in chapter 5 and references thereafter.

Chapter 2: Literature Review and Related Work

2.1 Literature Review

Manjunata and et al [3], presents the implementation of a reliable and standalone security system for home environment. The system is built in two parts: door lock access using face recognition and intruder detection with auto alerting mechanism and its purpose is twofold: intruder detection and door security. It is based on face recognition technology to grant access to only authorized people. Here, face recognition is partly achieved by using standard PCA (Principal Component Analysis) algorithm owing to its simplicity and effectiveness in extracting features of facial images and reducing dimension of captured images while holding the primary information at the same time [3]. The system is implemented using Raspberry Pi electronic development board and operates on battery power supply and wireless internet connectivity by using USB modem [3], all of which are effective yet cost friendly. Its ability to consume relatively less power and send notifications in form of both SMSs and emails makes it reliable. Though the use wireless internet connectivity to send notifications is efficient, the constant button-pressing mechanism by a user to establish authentication, to lock and unlock the door makes the system less contactless (less interactive) and thus, less passive. Moreover, the use of two cameras: web camera and Raspberry Pi Camera unnecessarily wastes resources since one camera can be configured and used to achieve the same results of both face recognition and intrusion detection.

Patil et al [4] proposed a door security system based on face recognition for unlocking the door. The system is implemented using Raspberry pi B+ model, Pi camera, relay for door unlock, and SIM300 GSM module. The Raspberry pi is a credit card sized computer powerful enough to run face recognition algorithms and issue relevant

commands like unlocking doors based on the outcome of the algorithms. The system works by interfacing the camera to capture image, creating a database of authorized people, capturing current image and saving it for matching with the database images, GSM module to send security alerts for any attempt to gain access, and finally, interfacing relay module for door unlocking mechanism[4]. The face recognition uses CPA algorithm to first capture and create database and then, Eigen faces methodology is used to find the matches in the database. The use of Raspberry pi and Pi camera makes this system cost-effective and accounts for less power consumption however, as far as security is concerned, to make it more secure, the system should have incorporated a locking mechanism after the person has successfully gained access to the door thereby ensuring that the door is not left open to unauthorized people in which case, the harm could be done even in the presence of GSM module for message notifications. Therefore, the system could further be improved to include locking mechanism after access has been granted and an alarm system in case administrators cannot directly have access to the messages.

Gupta et al. [5] developed a system that grants access to the door through face detection and recognition with raspberry pi. The hardware design comprised of "Raspberry Pi2 development kit, connecting cables, LCD, DC Motor driver IC, DC motor, Power Supply, USB keyboard, USB mouse and USB webcam"[5]. The interface of LCD, camera, and a motor to the Raspberry pi board proved vital to setting up the real-time application. The face recognition aspect of the project was inevitably divided into two parts: face detection conducted using Paul Viola and Michael Jones' effective method called Haar-based cascade classifiers and face recognition deployed using Principal Component Analysis algorithm with the Eigen face approach. LCD displays the person's name in an event where a match is found as a result of comparing a person's face with faces in the database, thereby triggering the motor driven by the motor

driver to rotate 15 seconds clockwise and five seconds anticlockwise demonstrating gate's opening and closing mechanism. Coding for face recognition and detection, and the hardware in general, was done in Python with the help of relevant Python libraries. The project was designed with cost in mind explaining the use of Raspberry pi kit, but it was deployed with too many components with the likes of USB keyboard and USB mouse which has a cost implication and reliability on terminal to execute the modular portions of the system is tedious and makes a system a little less standalone. Finally, with the newest versions of Raspberry Pi, such as like Pi 3 model B+ and Raspberry Pi 4 available, the project can be further be improved and extended to include notification and alarm modules.

Khodaskar et al developed a face detection system using Raspberry Pi2 which they programmed using Python and its OpenCV library whereby both face detection and face recognition were successfully tested. They leveraged on fast image processing owing to the fact that though most face detection algorithms have high detection rate, they require several seconds to detect faces in a single image, a processing speed that is insufficient for real-time applications [6]. With features like Bluetooth and Wi-Fi wireless technologies, Raspberry Pi was backed for being cheap and having enough power for HD video and image manipulation [6]. To examine its efficiency, the system was tested several standard face databases with the likes of AT&T, Caltech, JAFFE, Face96 and Grimace to mention few with and without noise and blurring effects. The analyses showed that the system performed efficiently and could be used for face detection even for poor quality images [6]. It was suggested however, that the system can be upgraded via interfacing infrared camera to extend its applications to class attendance system and smart surveillance monitoring security system deployed by most public security. Interfacing Raspberry Pi and Arduino UNO is said to come in handy for alcohol

detection, finger detection, agriculture humidity sensing, etc. Moreover, the system can be improved by using Raspberry Pi3 or 4 that is, in fact, more powerful than earlier versions.

Shrutika et al recalls that today's incidents of thefts, robbery, identity fraud, unwanted entrance call for security measures to protect people and their belongings. That, coupled with the deficiencies presented by traditional security systems, inspired them to propose a security system based on face recognition. The system uses Raspberry Pi model B3 and Raspberry Pi was selected over several other available microcontrollers for its high processing capacity, relatively low price, and ability to adapt in different programming modes [7] as far as cost, processing, and user friendliness are concerned. Face detection makes use of Haar Cascade Haar-like features; digital image features vital for recognizing objects; and face recognition uses the histogram of oriented gradients, abbreviated as HOG; a feature descriptor used in computer and image processing for recognition purpose [7]. HOG method counts occurrences of gradient orientation in localized portions of an image and its overlapping local contrast normalization offers better invariance to changes in illumination and shadowing hence, improved accuracy after which the Support Vector Machine (SVM) classifier makes decisions regarding the presence of an object [7]. However, Pravin Kumar et al[8] criticize HOG and SVM for not being able to detect a face when it is slightly titled or at an odd angle. Thus, alternatives such as CNN could be explored to improve system accuracy. The programming was done on Linux based operating system due to the open source code and the fact that it is freer to do software development on Linux. The system has excellent performance efficiency and Shrutika et al claims that new studies are looking into processing images on Raspberry Pi GPU to achieve better results via the use of specific libraries and so, it could improve this system further.

Pravin Kumar et al implemented a fast and cost-efficient detection system using Raspberry Pi as a microcontroller. The system uses Convolutional Neural Network (CNN) to train the model and detect faces whereas Gabor filter was deployed to extract important facial features. Raspberry Pi, found to be cost efficient, was connected to cloud service to make sending SMSs and e-mails to the administrators possible. HOG and linear SVM, referred to as a traditional face detection technique, could have been used to detect faces but, it is criticized of its inability to pick faces at odd angles. Therefore, CNN was used due to its ability to pick faces with any odd angles mitigating issues arising from posing. The system cuts off the need for the person to monitor from the screens for surveillance and takes advantage of cloud storage hence eliminating the need for storage and preventing scaling issues. The system was tested and worked well under crowded situations and fulfilled the application of a security system [8], however areas of improvement include distance, light and glare because longer distance and illumination hindered the system performance.

Venkatesh Bhutra et al proposed a door security system based on face recognition using Raspberry Pi. On one hand, Viola Jones method for face detection due to its fast detection rate. On the other hand, Principal Component Analysis (PCA) was used for face recognition. PCA, a statistical approach for face recognition and image compression, abridges huge high dimensional data [9] since image is represented in a matrix form. The hardware and software communicate to each other via the use of a Linux based operating system from MATLAB. The camera, connected to the Raspberry Pi, capture the image and, with the help of Viola Jones, the face is detected. It is this detected face that is matched with the training facial dataset using MATLAB code. If the face is recognized, MATLAB communicates with the Raspberry Pi to command the servo motor to open the doors, otherwise MATLAB instructs the Raspberry Pi to turn on the alarm [9]. The system

which works with a self-trained database was found to be working even with multiple faces as input images. It was found that changes in light intensities affect the efficiency of PCA and carefully increasing the number of training images with variation can compensate for the effects. Similarly, though the system works well with different image format, it is affected with variation in image size.

Najmurrokhman et al. [10] developed a secured room access system that uses face recognition to grant access to authorized people. Haar-cascade classifier was used, due to its high accuracy, to detect faces and process images. Nine users, with 15 faces each, were registered and the face recognition was tested under normal condition, less light intensity, and facial angle. System was designed for a 10 second delay to allow the authorized person to enter whereby after that delay the door is automatically locked. Web camera was used to provide wider view for face detection, an initial step in face recognition, as opposed to the use of a light proximity sensor consisting of photodiodes components. Raspberry Pi was used as the main control unit while OpenCV library facilitated image processing. The system is made to communicate with the master via Android-based smartphone application, Telegram, a cloud-based instant messaging and voice over IP service developed by the Russian entrepreneur Pavel Durov with so many advantages over other messenger services[10]. In case the face is not recognized, the master and in an unlikely event where the system fails to recognize a face that is in the database due to illumination, etc. the master is alerted and can decide to remotely unlock the door if he/she wants the person in. Coupled with solenoid lock and webcam, buzzer was used in case the access is not granted. The system works well however, some advanced techniques could be applied to enhance the performance of Haar-cascade classifier.

Attempting to replace traditional door locks, Pawar [11] proposed an advanced door lock system for home automation. The proposed system uses Raspberry Pi,

magnetic door lock, camera module, LCD screen, and Wi-Fi connection to develop a door lock system, based on face recognition, that is more reliable, hassle-free, and user friendly. Part of the system is intrusion detection. If someone tries to enter the house in unauthorized way, the house owner will be alerted and can act accordingly or take necessary measures. The face recognition aspect was divided into three major categories: preprocessing, feature extraction, and classification. Preprocessing involved capturing input image from the camera and locating the face in the image using Haar-cascade classifier for frontal face. Image is cropped and aligned in this section. Then, LBPH algorithm was used to learn on the dataset obtained from preprocessing stage. The accuracy of the classifier varies depending on the size of the data samples. Finally, the face recognizer is tested, and real time video check was done to verify the correctness of the trained model.

2.2 Related Works

In investigating the use of LBPH algorithm for face recognition to find missing people in Zimbabwe, Peace[17] implemented the same system and did two tests: First test considered 20 images with 18 images recognized correctly, leading to a 90% recognition rate. Second test considered 15 images in total with 13 images recognized correctly, leading to an 86.66% recognition rate. On average, the system achieved the average recognition rate of 67.5%. Eigenfaces and Fisher faces algorithms were also tested and the results showed that Local Binary Patterns Histogram algorithm performed better than them.

Similarly, Ibrahim and Zin [18] developed the system for office door security. Their system uses Eigenfaces algorithm, which is part of Computer Vision, derived from Principal Component Analysis (PCA). Their system achieved 80% recognition rate. The system operated within -20 degrees and +20 degrees at a distance of 40cm to 60 cm.

Chapter 2: Design and Methodology

The project consists of a Raspberry Pi camera mounted at the door. The camera is controlled by a Raspberry pi which does not just do all the computations regarding image processing and verification but command the solenoid lock to open the door in case the user has been recognized. One Raspberry Pi is used to act as a control unit or the heart of the system. A 48.4w, 12V solenoid lock was used for unlocking the door alongside the camera for capturing real-time images and images collection.The camera, placed on two servo motors for panning and tilting, is connected to the Raspberry Pi (board) via a bus and as it captures the image it sends it to the Raspberry Pi for processing and face-matching. It is the same Raspberry Pi that, after establishing a match or a no-match, send signals to the servo motor attached to the door to either keep the door locked in case the face is not recognized or unlock the door otherwise. The choice of the microcontroller depends on the nature of the project. Irshad and Feroz [12]
argues that Arduino Uno will best fit the project that does not require much processing power like controlling motors, sensors, LEDs, etc. However, project involving much physical computation and graphics like home/industrial automation, environmental sensing and monitoring, wireless access point etc., then Raspberry pi 3 is ideal. The figure below gives an overview or the architecture of the entire system.

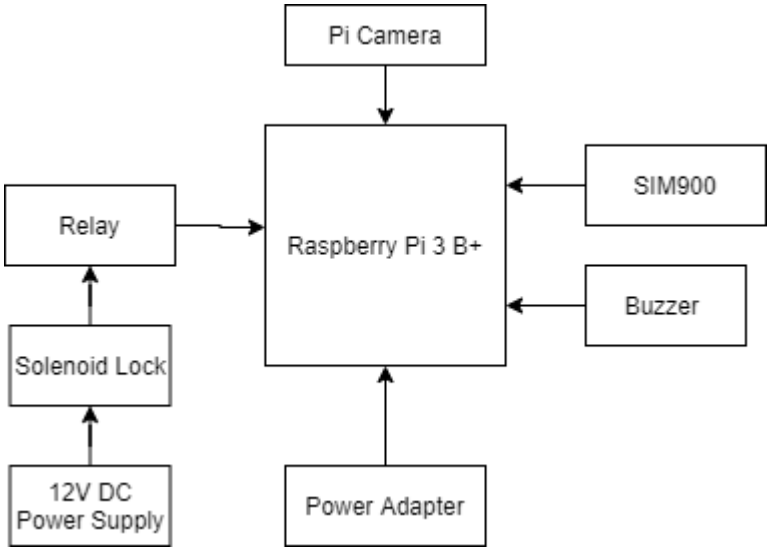

Figure 1. The system architecture

2.1. Design Decisions

In this section, a decision matrix was used to make informed and well-thought decision on the microcontroller to use based on the secondary research data from different manufacturers' data sheets. Developed by JASON, Engineering Design Decision Matrix involves using the selection criteria (also known as user requirements), and secondary research data to examine a choice of material using a scoring matrix. The criteria used were deduced from a list of user requirements stated in section? above. Each criterion is assigned a weight and each option is also assigned a weight. The scale used ranges from 1 to 5, 5 implying a better choice. Finally, each option weights were multiplied by criteria weights and the sum was used to inform the decision as demonstrated in figure # below.

2.1.1. Microcontroller

Microcontroller Decision Matrix								
			Options					
			Arduino Uno		Raspberry Pi 3 B+		ESP32	
Selection Criteria	**Weight (1-5)**	**Multiply**	**Score**	**Weighted Score**	**Score**	**Weighted Score**	**Score**	**Weighted Score**
Size	3	times	2	6	2	6	5	15
Reliability	4	times	1	4	4	16	3	12
Cost	4	times	4	16	2	8	4	16
Programming Skills	4	times	4	16	4	16	3	12
Hardware Interface	4	times	3	12	5	20	4	16
Power Efficiency	3	times	5	15	3	9	5	15
Temperature Tolerance	4	times	1	4	5	20	2	8
Processing Power	5	times	1	5	5	25	2	10
Memory	5	times	4	20	4	20	5	25
	Total Score	equals		**98**		**140**		**129**

Table 1. Design matrix for the selection of a microcontroller

From the scoring matrix table above, Raspberry Pi 3 B+ was found to be more suitable because of its decent processing power considering the computation to be done in this project. Among other factors, it has enough memory and it is relatively easy to work with compared to ESP32. An extra advantage with Pi is the fact that it comes with easily configurable camera and a Wi-Fi module part of it, not to mention the fact that it supports Python which is the programming language used in this project. Python has OpenCV, an open source library with various programming functions for real-time computer vision, which was used for writing face recognition algorithm for this project. (What is Raspberry Pi and insert its image here!)

2.1.2 Programming Language

For this project, Python programming language was selected over MATLAB for the factors summarized in the table below[13].

Table 1. Design matrix for the selection of a programming language

Criteria	Python	MATLAB
Description	Library of programming functions for real time computer vision, cross-platform, free for use under BSD license.	Numerical computing environment, developed by Cleve Moler, allows matrix manipulation, plotting of functions, supports algorithm implementation, the creation of user interfaces.
Speed of execution	Executes much faster,	Slower, analyze 3-4 frames

	examine 30 frames per second.	per second.
Operating system	Runs well on Windows, Linux, macOS, Android, iOS etc. Any device that can run ‘C’ can run ‘OpenCV’	Runs well on Windows, Linux, macOS. It can call functions written in ‘C’ or ‘Fortran.’ MATLAB can be directly called from Perl, Java, ActiveX, NET
Cost	Free as it is BSD licensed.	Each toolbox is purchased separately.
Resources needed	It needs less memory.	More RAM is required.

According to the above table, Python is best suited for this project because in addition to being fast and free, it is simple and allows for code readability and ease performance of OpenCV bindings while requiring far less lines of code.

Chapter 3: Implementation

3.1 Face Recognition Module

The following tools were used to implement the face recognition module:

- Open Source Computer Vision Library (OpenCV) which contains programming functions for real-time computer vision with machine-learning algorithms used to train the image dataset
- Haar feature-based cascade classifier for face detection which comes with OpenCV
- LBPH (Local Binary Pattern Histograms) algorithm that also comes with OpenCV
- Raspberry Pi Camera (Due to COVID-19 which affected procurement of most components, in-built Lenovo ThinkPad L560 camera was used to take faces for training and testing)
- Python programming language due to its simplicity and code readability with few lines of code to do complex task and full compatibility with OpenCV (Python 3.6.10)
- Numpy which is an optimized library for numerical operations, mainly to handle arrays from and to OpenCV library since images are translated into matrices for easy manipulation
- Anaconda 3 with Spyder: development environment for writing python programs

The face recognition module in this project comprises of three major tasks: detecting and collecting face samples, training the recognizer, and recognizing the face in real time. Each of these stages is fully described below with code snippets added.

3.1.1 Face Detection and Data Gathering

For a face to be detected, the object, a person in this case, has to be detected first. The object detection algorithm was implemented using Haar Feature-based Cascade Classifiers. Paul et al proposed an object detection approach that classifies images based on the value of simple features. In their machine learning based approach, a cascade function is trained from many positive and negative images which, in turn, is used to detect objects in other images. Initially, the algorithm is fed with a lot of positive images, images with faces, and negative images, images without faces, to train the classifier. Features are then extracted from the algorithm using Haar features, which are just like convolution kernel, like the ones shown in the image below. Each feature is a single value found by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.

One of the two reasons they decided to go with features ahead of pixels was the fact that feature-based system operates much faster than pixel-based system. The three kinds of features are used. The two-rectangle feature whose value is the difference between the sum of the pixels within two rectangular regions, the three-rectangle feature whose value is the sum within two outside rectangles subtracted from the sum in a center rectangle, and the four-rectangle feature whose value is the difference between diagonal pairs of rectangles. Their detector's base resolution is 24x24 resulting in over 180,000 rectangle features associated with each image sub-window, a number far larger than the number of pixels.

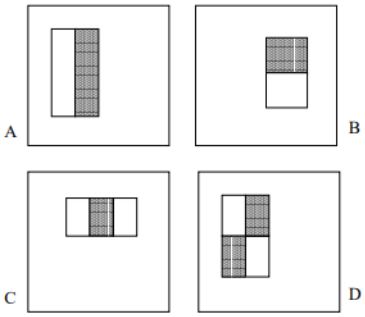

Figure 2. Rectangular features (Haar-like features)

Enormous number of features are calculated from all possible sizes and locations of each kernel whereby the sum of pixels under the white and black rectangles are computed for each feature calculation. Viola et al. [14] introduced a new image representation named the "Integral Image" ensuring a faster computation of features. They argue that most of the features calculated are irrelevant. For illustration, in the image below, the first raw depicts two good features whereby the first feature demonstrates that the region of the eyes is often darker than that of the nose and cheeks while the second demonstrates that the eyes are darker than the nose bridge. From this, applying the same windows on any other place of the image is irrelevant.

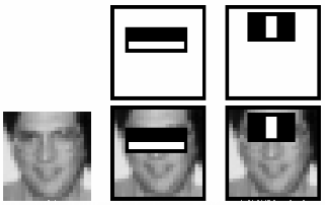

Figure 3. Demonstrating how Haar-like features work on face image

Over 180,000 features are extracted but computing the complete set is prohibitively expensive. Therefore, they believed that "a very small number of these features can be combined to form an effective classifier" hence, the use of "a learning

algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers"[14].

With AdaBoost algorithm, each image is initially assigned equal weight and after each classification, weights of misclassified images are increased. The same process is done over, and over again as new error rates and weights arise until the required accuracy or error rate is reached or desired number of features are found. Final classifier is a weighted sum of these weak classifiers. Weak classifier simply because it alone cannot classify the image, however combined with others forms a strong classifier. The paper reveals that even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features, a huge reduction from over 180,000 features to 6,000 features. However, even with 6000 features, it is still time-consuming and inefficient to apply these on a single image to check if it is a face or not. In response to that, they introduced a method called "Cascade of Classifiers" which combines increasingly more complex classifiers in a cascade enabling background regions of the image to be discarded while more computation focuses on promising object-like regions. "Cascades provides statistical guarantees that discarded regions are unlikely to contain the object of interest [14]." Here, rather than applying all the 6000 features on a window, features are grouped into different stages of classifiers and apply one-by-one. Usually, first few stages will contain very a smaller number of features. If a window fails the first stage, it is immediately discarded and remaining features on it are not considered. Upon passing the first stage, the second stage of features is applied, and the process goes on. Thus, the window which passes all stages is a face region. Their detector had 6000 features grouped into 38 stages with 1, 10, 25, 25 and 50 features in first five stages. According to the authors, on average, 10 features of 6000 are evaluated per sub-window. Their system results into detection rates comparable to the best previous systems and, used in real-time applications, the detector

runs at 15 frames per second without resorting to image differencing or skin color detection.

Face detection of the face recognition system presented in this paper relies on Haar-cascade Detection implemented with the help of Python's OpenCV library. OpenCV contains many pre-trained classifiers for face, eyes, smile and more saved as XML files. This project focuses on recognizing faces from a frontal view, thus frontal face Haar classifier is used. The following code's purpose is threefold: it detects face, does some preprocessing such as cropping, RGB to grayscale conversion, and saving the captured face in a specified folder within the project folder. In this project, 150 faces of a single person were used under normal room lighting condition.

```
import numpy as np
import cv2

cascadeClassifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def extract(img):

    grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = cascadeClassifier.detectMultiScale(grayImage, scaleFactor=1.5, minNeighbors=5)

    if faces is ():
        return None

    for (x,y,w,h) in faces:
        crop = img[y:y+h, x:x+w]
    return crop

cap = cv2.VideoCapture(0)
count = 0

while True:
    ret, frame = cap.read()
    if extract(frame) is not None:
        count += 1
        face = cv2.resize(extract(frame),(200,200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        path = 'faceSample/usr'+str(count)+'.jpg'
        cv2.imwrite(path, face)
        name = str(count)
        font = cv2.FONT_HERSHEY_DUPLEX
        color = (0,255,0)
        stroke = 2
        cv2.putText(face, name, (50,50), font, 1, color, stroke)
        cv2.imshow('Face Cropper', face)
    else:
        print("Error: No Face")
```

```
44          pass
45
46      if cv2.waitKey(20) & 0xFF == ord('q') or count==150:
47          break
48
49  print("----------------------------------------")
50  print("Sample collection complete")
51  print("----------------------------------------")
52
53  cap.release()
54  cv2.destroyAllWindows()
55
```

Figure 4. Demonstrating how faces can be detected and captured or collected

From the above code:

- Line 8 and 9 import OpenCV and NumPy
- Line 11 loads the classifier which is stored in the working directory
- At line 13, the function *extract* is defined to take image as an input and return its face, if any, cropped
- Line 16, the classifier function is called with essential parameters namely the input graysale image, scale factor for how much the image is reduced at each scale, and minNeighbors. Fine tuning is done to get the optimum results.
- The *while* loop from line 28 takes a cropped face resulted from *extract* function and does resizing, conversion from to grayscale, and storing a .jpg face to the designated folder in the project

Sammple faces collected from the above code are illustrated in the RESULTS section of this report.

3.1.2 Face Recognition

Now that the face can and has been detected and that 150 faces of a single person have been collected, the next two steps were to train the available faces and apply the trained

faces to recognize a given face in a real-time video stream. Both face training and recognition were done using Local Binary Pattern Histogram (LBPH) algorithm.

3.1.2.1 Local Binary Pattern Histogram (LBPH) Algorithm

LBPH algorithm is one of the feature-based approaches towards face recognition that combines Local Binary Patters (LBP) and Histogram of Oriented Gradients (HOG) descriptor [15]. Designed for analyzing texture of gray-scale image, "LBP is an easy but powerful way to extract and label the pixels of an image." The region of the face is partitioned into small units from which LBP histograms can be derived, extracted, and concatenated into a single feature vector which efficiently represents the face and is used to derive similarities between images [17]. Even though 3D head pose changes, illumination variations, facial expression, occlusion, and ageing affect the performance of any facial recognition algorithm, Peace [17] argues that the algorithm's major advantages are that it is not profound to illumination and that it offers better recognition rates in controlled environments thanks to the LBP operator that mitigates illumination effects.

3.1.2.1.1 How it works

Forming a part of OpenCV, LBPH algorithm is designed in this way:
Given an image of N by M dimension, it is partitioned into units of same height and width say, m by m dimension for each unit of the face. LBP operator is then applied on each unit, usually defined in window of 3 by 3, and is mathematically expressed as

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} 2^p s(i_p - i_c) \qquad (1)$$

In the above expression, (x_c, y_c) is the central pixel with intensity of i_c while i_n is the intensity of the neighbor pixel.

Median pixel value is regarded as a threshold pixel for comparing a pixel to its eight closest pixels with the help of a sign function, s, defined as:

$$s(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad (2)$$

Here, in case the neighbor value is greater than or equal to the central value, it is set to 1, 0 otherwise. As a result, 8 binary values are derived from 8 neighbors which are combined to give 8-bit binary number translated into decimal number referred to as pixel LPB value with the range from 0 to 255. This is done so that arbitrary number of neighbors could be aligned in a circle of variable radius as demonstrated in the figure below.

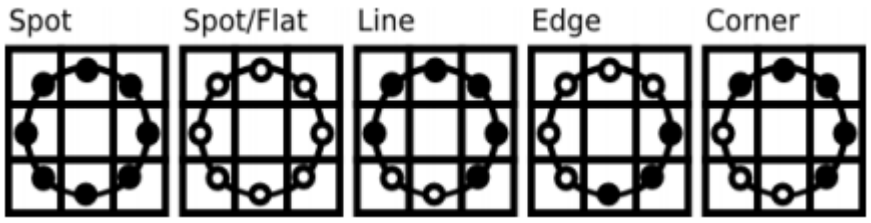

Figure 5. Captured neighbourhoods arranged in a circular fashion

Figure 1: Captured neighborhoods arranged in a circular fashion

Now, for any point (x_c, y_c), to calculate the position of the neighbor (x_p, y_p) such that $p \in P$ the following expressions are used.

$$x_p = x_c + R\cos(\frac{2\pi p}{P}) \qquad (3)$$

$$y_p = y_c - R\sin(\frac{2\pi p}{P}) \qquad (4)$$

In the above expressions, R is the radius of the circle and P is the number of sample points. In case points coordinate on the circle does not correspond to image coordinates, it

is then interpolated with bilinear interpolation. Below is the expression used for interpolating.

$$f(x,y) \approx \begin{bmatrix} 1-x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \quad (5)$$

The LBP operator is proved to be robust against monotonic gray scale transformations which is why the captured images are translated into gray scale with the following line of code:

```
15        grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Figure 6. Code line to convert image or face from BGR to grayscale for further processing

Once the LBP value has been found, histogram of the region is generated by counting the number of similar LBP values in the region, after which histograms of each region are merged to create a single histogram known as the image feature vector. Therefore, having feature vector for both test image and database image, their histograms can be compared and the image with the closest histogram can be returned. Techniques such as Euclidean distance, chi-square, and absolute value among others can be used to make the comparison. For instance, Euclidean distance can be calculated by comparing the features of test image with those found in the dataset where the minimum distance between the test and original image gives the matching rate. $d(a,b) = \sqrt{\sum_{i=1}^{n} |a_i - b_i|^2}$

Finally, the algorithm, which was found to be able to recognize both side and front faces and unaffected by illumination variations, outputs an ID of the image if the test

image is found the database, otherwise, the image will get the ID of “Unknown” to imply that it has not been recognized.

LBPH Algorithm flowchart is presented here below:

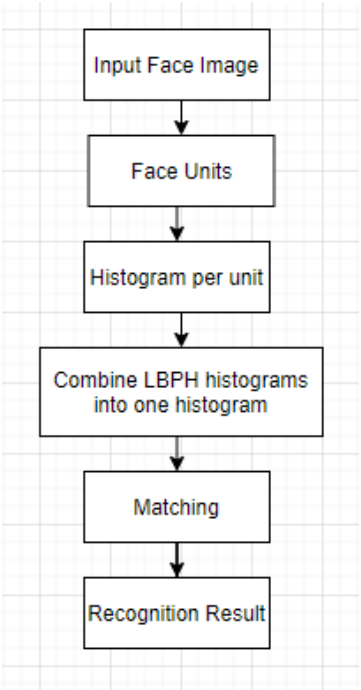

Figure 7. LBPH Algorithm Flowchart

3.1.1 Face Detection and Data Gathering

For simplicity purposes, both training and recognition modules were implemented at a go. The full code is illustrated down below.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from os import listdir
from os.path import isfile, join

pathTrain ='C:/ASHESI UNIVERSITY COLLEGE/FR-Version1/faceSample/'
file = [f for f in listdir(pathTrain) if isfile(join(pathTrain,f))]

trainingData, labels = [], []

for i, files in enumerate(file):
    trainImagePath = pathTrain + file[i]
    images = cv2.imread(trainImagePath, cv2.IMREAD_GRAYSCALE)
    trainingData.append(np.asarray(images, dtype=np.uint8))
    labels.append(i)

labels = np.asarray(labels, dtype=np.int32)

model = cv2.face.LBPHFaceRecognizer_create()

model.train(np.asarray(trainingData), np.asarray(labels))


print("-----------------------------------------------------")
print("Training Complete")
print("-----------------------------------------------------")

cascade_classifier = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

def detect(img, size = 0.5):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = cascade_classifier.detectMultiScale(gray, 1.3, 5)

    if faces is():
        return img,[]
```

```python

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,255), 2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200,200))

    return img,roi

acc = []

cap = cv2.VideoCapture(0)
while True:

    ret, frame = cap.read()

    image, face = detect(frame)

    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)
        res = model.predict(face)

        'Calculating accuracy'
        if res[1] < 100:
            confidence = float("{0:.2f}".format((100*(1-(res[1])/300))))

            acc.append(confidence)
            dis = str(confidence)+'% similar to Alain'

        font = cv2.FONT_HERSHEY_DUPLEX
        name = dis
        color = (250, 120, 255)
        stroke = 2
        cv2.putText(image, name, (100,120), font, 1, color, stroke)

        'Access Control: Door gets unlocked for accuracy of 80% and above'
        if confidence > 80:
            font = cv2.FONT_HERSHEY_DUPLEX
```

```
            name = "Door Unlocked"
            color = (0, 255, 0)
            stroke = 2
            cv2.putText(image, name, (250, 450), font, 1, color, 2)
            cv2.imshow('Face Cropper', image)

        else:
            'Access Control: Door remains locked for accuracy below 80%'
            font = cv2.FONT_HERSHEY_DUPLEX
            name = "Door Locked"
            color = (0, 0, 255)
            stroke = 2
            cv2.putText(image, name, (250, 450), font, 1, color, 2)
            cv2.imshow('Face Cropper', image)

    except:
        font = cv2.FONT_HERSHEY_DUPLEX
        name = "No Face Detected"
        color = (255, 0, 0)
        stroke = 2
        cv2.putText(image, name, (250, 450), font, 1, color, 2)
        cv2.imshow('Face Cropper', image)
        pass
    'Exit by pressing q button on keyboard'
    if cv2.waitKey(20) & 0xFF == ord('q'):
        break

print(acc)
print('Highest Accuracy Generated is ' + str((np.max(acc))))
print("-----------------------------------------------------")

plt.plot(acc)
plt.ylabel('Accuracy')
plt.xlabel('Run Time')
plt.title('Confidence Plot')
plt.show()

cap.release()
cv2.destroyAllWindows()
```

Figure 8. Code for Face Recognition Module excluding Notification Module

Code Description

- Necessary libraries like OS, OpenCV, and NumPy are imported
- To train the model, training data are gotten at line 14 and 15
- Line 19 – 25: training data and labels are generated
- Line 27 and 29 both deal with creating and training LBPH model
- Line 36 loads the classifier for the test image
- Accuracy and decisions pertaining to locking and unlocking as well as SMS notifications are configured in the while loop starting from line 55

- Highest accuracy and graphs are generated between line 108 and 116 and cleanup is made at lines 118 and 119

The flowchart below depicts how the entire system was intended to be run.

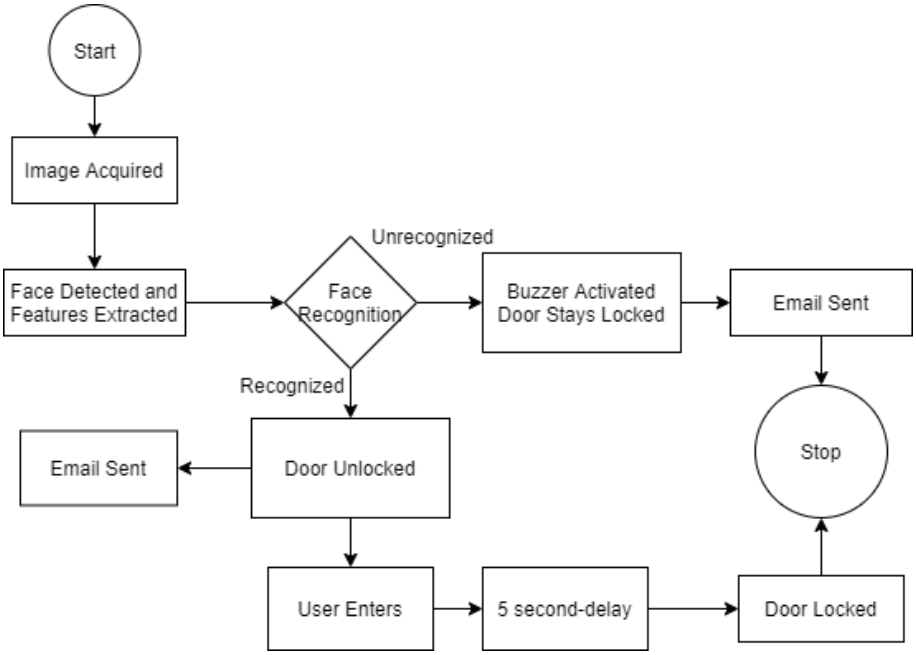

Figure 9. Flowchart of the system

The following flowcharts illustrates how face detection and recognition are performed, respectively.

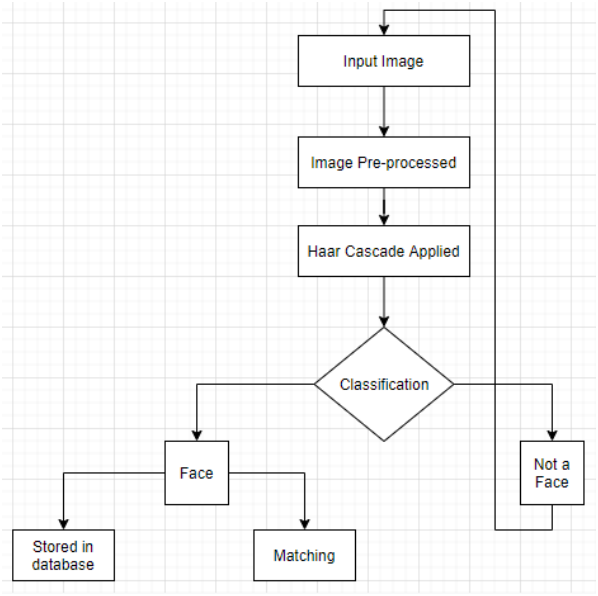

Figure 10. Face Detection Diagram

In this project, the Haar-cascade classifier which operates in the way of staging. The flowchart is shown here below:

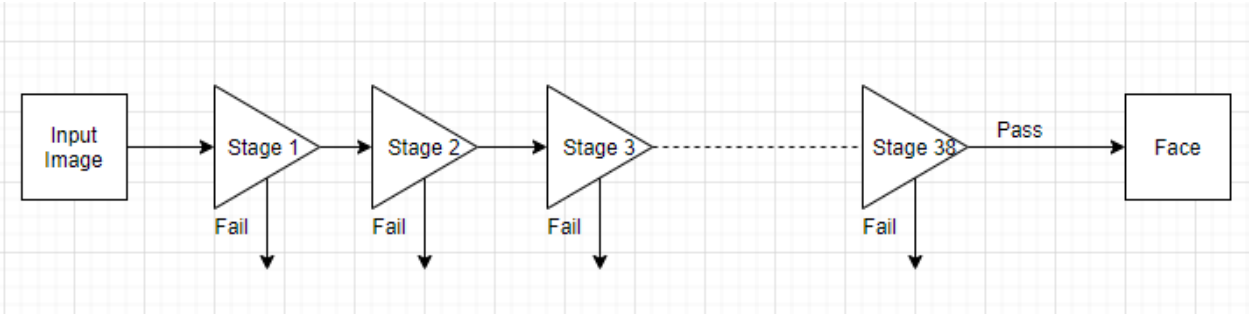

Figure 11. Classification process (Haar-cascade classifier)

A built-in laptop camera was used in place of the Pi camera and the results of running the recognition were observed. One of such results is shown above. The accuracy is not excellent partly because of the dataset used. Lack of variations in illumination conditions, facial expressions, as well as the number of datasets considered for training the model. The working principle is that if an image is recognized beyond a certain threshold, a

confidence level in other words, then the door will automatically become unlocked and the system alerts the system admin via a text message sent directly to his or her smartphone to justify both successful and unsuccessful attempts.

3.2 Buzzer Module

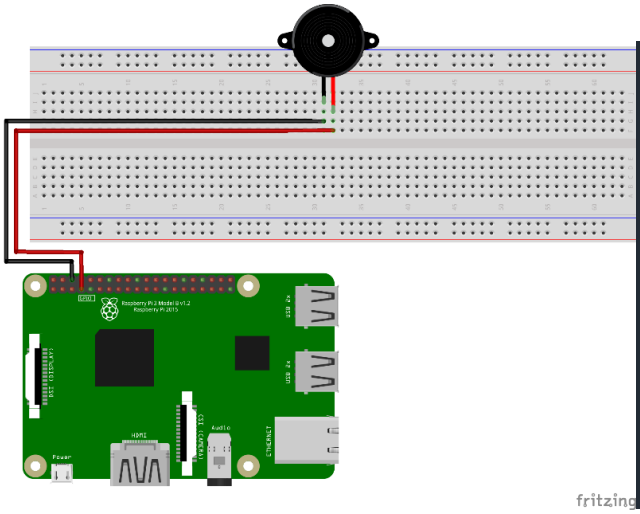

Figure 12. Circuit diagram of

```
8   import time
9   import RPi.GPIO as GPIO
10
11  GPIO_BUZZER = 4
12
13  GPIO.setwarnings(False)
14  GPIO.setmode(GPIO.BOARD)
15  GPIO.setup(GPIO_BUZZER,GPIO.OUT)
16
17  try:
18      while True:
19          GPIO.output(4,0)
20          time.sleep(.2)
21          GPIO.output(4,1)
22          time.sleep(.2)
23  except KeyboardInterrupt:
24      GPIO.cleanup()
25      exit
```

Figure 13. Code for buzzer module

This module is implemented to ensure that the warning is made to unauthorized person trying to gain access to the system. The code above is placed in the main code of face recognition right after line 94 since at that point the image of the person standing in front of the camera is not recognized by the system. As a result, the buzzer will beep non-stop until a manual reset is done by pressing a keyboard key in this case.

3.3 Door Lock Module

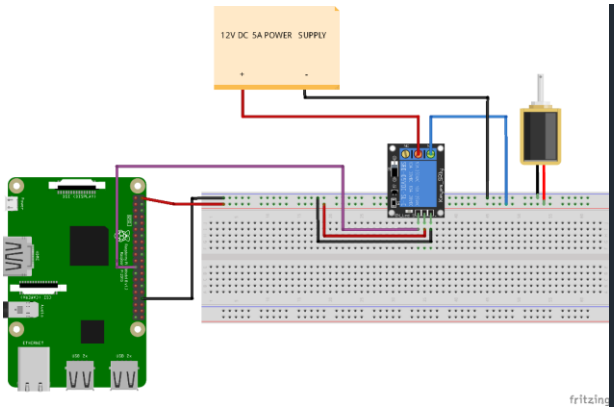

Figure 14. Circuit diagram of locking module

```python
import time
import RPi.GPIO as GPIO

# Setting pin 23 as output
GPIO_RELAY = 23

GPIO.setmode(GPIO.BOARD)
GPIO.setup(GPIO_RELAY, GPIO.OUT)
GPIO.output(GPIO_RELAY, GPIO.LOW)

# Face Recognized?
GPIO.output(GPIO_RELAY, GPIO.HIGH)
print("Door Unlocked")
time.sleep(10) # Door Remains Unlcoked for 10 seconds
GPIO.output(GPIO_RELAY, GPIO.LOW)

# Face Unrecognized?
print("Unidentified => Unauthorized!")
GPIO.output(GPIO_RELAY, GPIO.LOW)

GPIO.cleanup()
```

Figure 15. Code for locking module

For this module, once the person has been recognized from the face recognition module, the above code is called and run to power up the solenoid lock which, in turn, unlock the door for the recognized person to enter. For unrecognized person, the GPIO pin configured to relay is set to LOW, hence the door remains locked and access is denied. Solenoid lock requires 12V DC and 48.4 watts with the calculated current, I, of ${}^{48.4W}/_{12V} = 4.03A$.

3.4 Notification Module

This module is implemented to notify the sysadmin of any successful and unsuccessful attempt to gain access through the door. Below is the python script for sending email notification to the sysadmin. The module was implemented in Python with the help of SMTP (Simple Mail Transfer Protocol). SSMTP, which is a lightweight smtp server used to send email, was installed to allow for the mail transfer. In the script, variable *server* was defined to establish the connection to the server; the sender's and

receiver's emails, the sender's password, the subject, and the body or message were declared. The process uses TLS encryption. This script is always run whenever there is an attempt, be it successful or not, to access the door to keep the sysadmin up to date with the current events with regards to door security. The choice of email notification with SSMTP came because of the fact that sending notification most often requires using web service APIs provided by third-party companies such as Twilio to send email or phone calls. These companies' services are high-priced. The notification module implemented in this project relies solely on Wi-Fi and Raspberry Pi 3 comes with in-built Wi-Fi which makes it easier to work with at a relatively low cost.

```
import smtplib

senderEmail = "accesscontrolnotify@gmail.com"
receiverEmail = "ahkubwayo@gmail.com"

password = "reum jabo tecy tfbe"
subject = 'Access Control Notification'
message = "Attempt to gain access through the door!"
header='To: ' + receiverEmail + '\n' + 'From: ' + senderEmail + '\n' + 'Subject: ' + subject

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login(senderEmail, password)
print("Login successful!")
server.sendmail(senderEmail, receiverEmail, header + '\n\n'+ message)
print("Email has been sent to ", receiverEmail)
```

Figure 16. Code for locking sending email

```
C:\ASHESI UNIVERSITY COLLEGE\Version1-Edit>python notify.py
Login successful!
Email has been sent to  ahkubwayo@gmail.com

C:\ASHESI UNIVERSITY COLLEGE\Version1-Edit>
```

Figure 17. Console showing email login successful

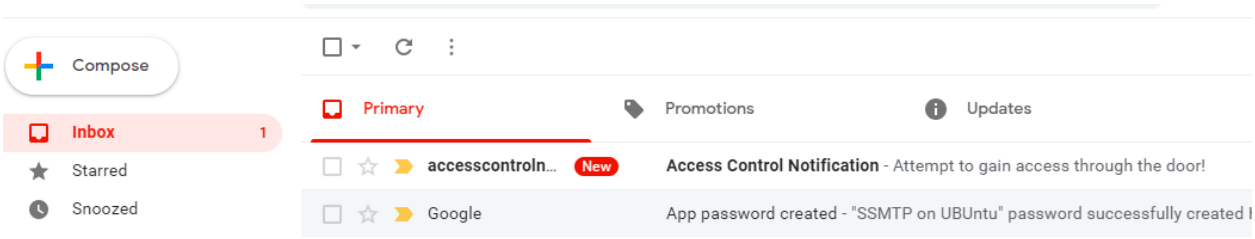

Figure 18. Sample email sent via SMTP

Chapter 4: Results and Discussion

4.1 Face Recognition

Face capture and application of face recognition in real-time were done using a built-in 720p HD front-facing laptop camera. After running the module on Lenovo ThinkPad L560 (Windows 10 64-bit, 8GB RAM), the following results were ascertained. Three tests were conducted, and accuracy(recognition) rates were averaged to determine the accuracy rate of the face recognition implemented. During the testing phase, three test subjects (cases) were considered. A person already defined in the system, a person not stored in the system, and the picture without a face. From the tests, highest accuracy found is 89.2% and the overall face recognition accuracy rate is 88.6%. Refer to table T. #. Full details of the results and their analysis are illustrated below.

Test 1

Highest Accuracy: 88.43%

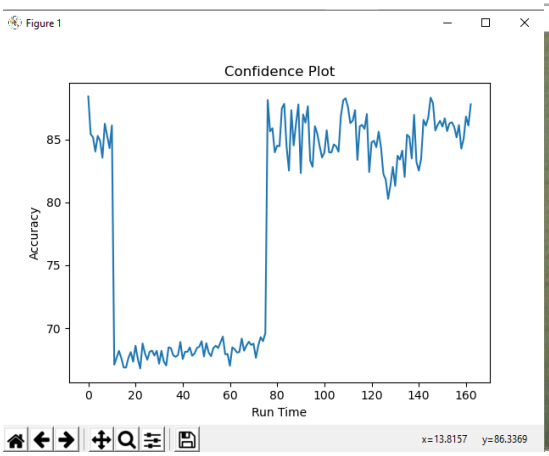

Figure 19. Confidence plot

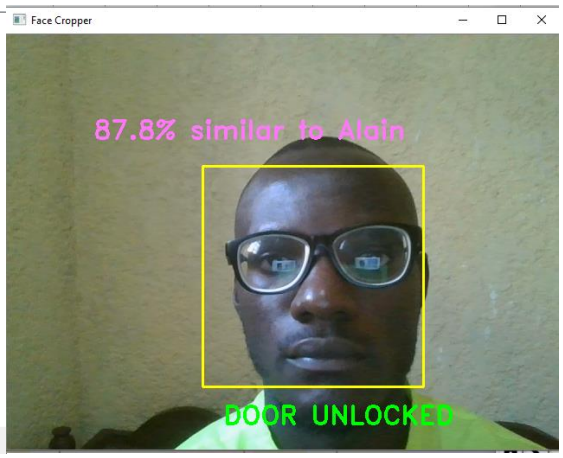

Figure 20. Recognized Face

Test 2

Highest Accuracy: 89.2%

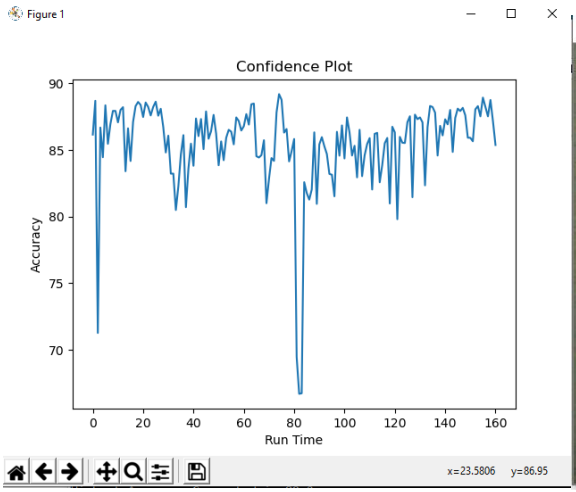

Figure 21. Confidence plot

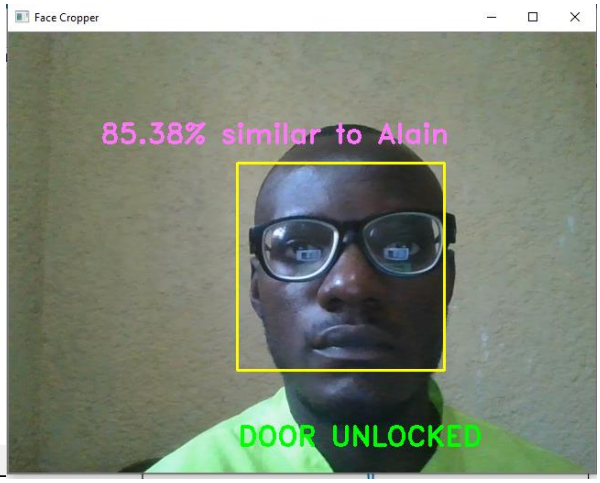

Figure 22. Recognized Face

Test 3

Highest Accuracy: 88.18%

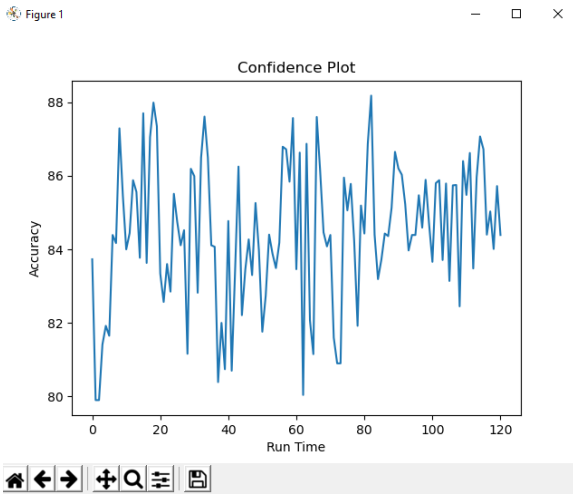

Figure 23. Confidence plot

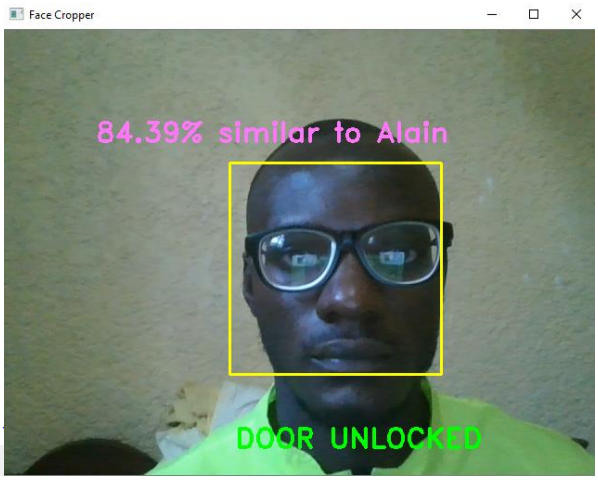

Figure 24. Recognized Face

Table 3. Test results, average accuracy, and highest accuracy

Test 1	88.43%
Test 2	89.2%
Test 3	88.18%
Average Accuracy	**88.6%**
Highest Accuracy	**89.2%**

- System achieved 88.6% recognition rate higher than what 80% achieved by Ibrahim's system
- LBPH algorithm achieves better results than Eigenface
- Factors like datasets and illumination can interfere with the result

Unauthorized Individual:

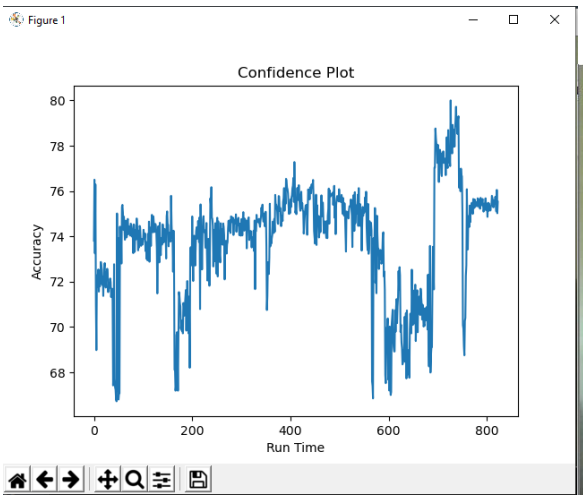

Figure 25. Confidence plot

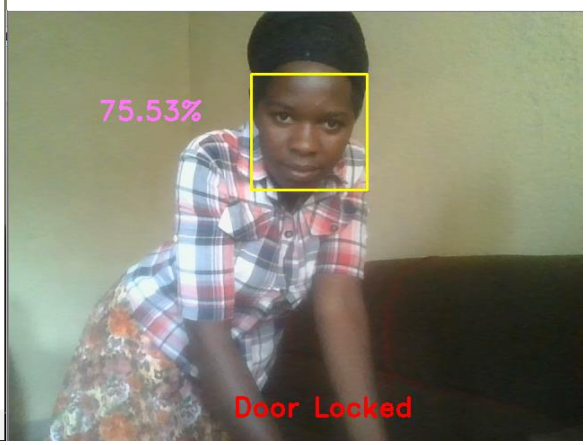

Figure 26. Unrecognized Face

Figure 26. Sample collected face

Figure 26. Sample collected face

4.1.1 Challenges

The accuracy rate found is relatively decent however, it could be improved. According to Joanna [16], challenges facing face recognition include, but not limited to, the image resolutions, varying lightning conditions, pose variations, occlusions, changing facial expression, face ageing, and quality of face datasets, all of which affect the result of the face recognition being implemented. 720p camera does not come closer to most commercial cameras on the market. In addition, the test was done in-door with not that sufficient lighting condition as can be seen in the sample faces show above.

4.1.2 Comparison to face recognition system

In investigating the use of LBPH algorithm for face recognition to find missing people in Zimbabwe, Peace[17] implemented the same system and did two tests: First test considered 20 images with 18 images recognized correctly, leading to a 90% recognition rate. Second test considered 15 images in total with 13 images recognized correctly,

leading to an 86.66% recognition rate. On average, the system achieved the average recognition rate of 67.5% which is quite low compared to 86.6% achieved here in this project, though to achieve it, 150 images of a person were considered. Illumination, as the rate declined with poor lighting conditions, and the hardware, most specifically, the lack of high definition camera with high resolution are the main challenges faced. His recommendation was to implement the system on DSP (digital signal processors) and use hardware devices like Raspberry Pi and due to COVID-19 which disrupted the procurements of components for this project, Raspberry Pi was not procured in time despite being the initial plan.

Chapter 4: Conclusion and Recommendations

This paper presented the design and implementation of smart door security system using face recognition as one of the most secure biometric technology. The project was extended to include an SMTP-based email notification and a buzzer both intended to notify the system owner or sysadmin of the ongoing events with regard to the system. Due to unexpected situation of coronavirus (COVID-19), most components, such as Raspberry Pi 3, that were originally planned to be used to implement this project were not procured which affected the project scope. Hence a laptop built-in camera was used instead of a Pi cam. Face recognition system developed showed decent results however, a lot of improvement and upgrade can be made to further improve the accuracy of the system. Few recommendations include capturing as many datasets as possible whereby each individual can have a lot of pictures under different variations with the likes of pose, smile, lighting conditions, and many more. The face recognition algorithm performed well reaching the recognition rate of 88.6% outperforming some existing systems. Hence, this project can reliably be implemented in real time.

Regarding the system, the following were observed:

- It achieved a very decent recognition rate achieved (88.6%)
- System could send the notifications
- Locking mechanism and buzzer modules designed but not tested due to time constraint and unavailability of materials
- System was run on PC rather than on Raspberry Pi
- Future works could include improving the system to cope with varying illumination conditions, pose variations, and facial expressions
- Recommendation is to increase the number of training images

References

[1] M. Hassaballah and S. Aly, “Face recognition: Challenges, achievements and future directions,” *IET Computer Vision*. 2015.

[2] S. Zeenathunisa, A. Jaya, and M. A. Rabbani, “A biometric approach towards recognizing face in various dark illuminations,” in *Proceedings of International Conference on Electronics Communication and Computing Technologies 2011, ICECCT’11,* 2011.

[3] R. Manjunatha and R. Nagaraja, “Home Security System and Door Access Control Based on Face Recognition,” *Int. Res. J. Eng. Technol.*, 2017.

[4] A. N. Patil, R. B. Ranavare, D. V Ballal, and A. P. P. Kotekar, “Raspberry Pi Based Face Recognition System For Door Unlocking,” *Int. J. Innov. Res. Sci. Eng. vol*, 2016.

[5] I. Gupta, V. Patil, C. Kadam, and S. Dumbre, “Face detection and recognition using Raspberry Pi,” in *WIECON-ECE 2016 - 2016 IEEE International WIE Conference on Electrical and Computer Engineering*, 2017.

[6] H. V. Khodaskar and S. Mane, “Human Face Detection & Recognition Using Raspberry Pi,” 2017.

[7] S. V. Deshmukh and P. D. K. U. A., “Implementation of Human Face Detection System for Door Security using Raspberry Pi,” *IJIREEICE,* 2017.

[8] S. Pravin Kumar and B. Balachander, “Fast and cost efficient face detection system with CNN using raspberry Pi,” *Int. J. Eng. Adv. Technol.*, 2019.

[9] V. Bhutra, H. Kumar, S. Jangid, and L. Solanki, “Door Security using Face Detection and Raspberry Pi,” in *IOP Conference Series: Materials Science and Engineering*, 2018.

[10] A. Najmurrokhman, K. Kusnandar, A. B. Krama, E. C. Djamal, and R. Rahim,

"Development of a secured room access system based on face recognition using Raspberry Pi and Android based smartphone," in *MATEC Web of Conferences,* 2018.

[11] O. Pawar, "Door Lock System using Facial Recognition," *Int. J. Res. Appl. Sci. Eng. Technol.,* 2019.

[12] M. E. Irshad and M. M. Feroz, "International Journal of Computer Science and Mobile Computing Scope of IoT: Performance and Hardware Analysis Between Raspberry Pi-3 And Arduino Uno," *Int. J. Comput. Sci. Mob. Comput.,* 2016.

[13] K. Jain, S. Narang, M. Saxena, and A. Arora, "Comparison of Face Recognition Algorithms Using Opencv for Attendance System," *Int. J. Sci. Res. Publ.,* 2018.

[14] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* 2001.

[15] F. Deeba, A. Ahmed, H. Memon, F. A. Dharejo, and A. Ghaffar, "LBPH-based enhanced real-time face recognition," *Int. J. Adv. Comput. Sci. Appl.,* 2019.

[16] J. I. Olszewska, "Automated Face Recognition: Challenges and Solutions," in *Pattern Recognition - Analysis and Applications*, 2016.

[17] P. Muhambo, "An Investigation on the Use of LBPH Algorithm for Face Recognition to Find Missing People in Zimbabwe", International Journal of Engineering Research & Technology, vol. 7, July 2018.

[18] R. Ibrahim and Z. M. Zin, "Study of automated face recognition system for office door access control application, "in *2011 IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011,* 2011