SMART LOCK MANAGEMENT SYSTEM USING AI

EPICS report submitted in partial fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

SK SAMEER	198W1A05B5
K SHANMUKHA	198W1A05B8

Under the Guidance of

Dr. G. ANURADHA

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

V.R SIDDHARTHA ENGINEERING COLLEGE

Autonomous and Approved by AICTE, NAAC A+, NBA Accredited

Affiliated to Jawaharlal Nehru Technological University, Kakinada

Vijayawada 520007

2022

V.R. SIDDHARTHA ENGINEERING

COLLEGE (AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the EPICS Report entitled **"SMART LOCK MANAGEMENT SYSTEM USING AI"** being submitted by **SHAIK SAMEER (198W1A05B5), K SHANMUKHA (198W1A05B8),** partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the Jawaharlal Nehru Technological University, Kakinada is a record of bonafide work carried out under my guidance and supervision.

Dr.G Anuradha
Associate Professor & Guide

Dr. D Rajeswara Rao MTech,Ph.D
Professor & HOD

DECLARATION

We hereby declare that the EPICS project entitled **"SMART LOCK MANAGEMENT SYSTEM USING AI"** submitted for the B.Tech Degree is our original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Vijayawada
Date: 07-04-2022

SHAIK SAMEER	198W1A05B5
K SHANMUKHA	198W1A05B8

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without whom it would ever have come into existence. To them we lay the words of gratitude imprinted with us.

We would like to thank our respected Principal, **Dr. A.V. Ratna Prasad** and **Dr. D. Rajeswara Rao,** Head of the Department, Computer Science and Engineering for their support throughout our Project.

It is our sincere obligation to thank our guide, **Dr.G.Anuradha,** Senior Assistant Professor, Department of Computer Science and Engineering, for her timely valuable guidance and suggestions for this Project.

We owe our acknowledgements to an equally long list of people who helped us in this Project. Finally, we wish to thank all the supporting staff who gave us facility in lab for the completion of this Project.

Place: Vijayawada
Date: 07-04-2022

SHAIK SAMEER 198W1A05B5
K SHANMUKHA 198W1A05B8

TABLE OF CONTENTS

ABSTRACT

Nowadays, we are facing security issues in every aspect. So we have to resolve these issues by using updated technology. In this project, we are using the Face recognition module to capture human images and to compare with stored database images. The most important of feature of any home security system is to detect the people who enter or leave the house. Instead of monitoring that through passwords or pins, unique faces can be made use of as they are one's biometric trait. We aim to create a smart door, which secures the gateway on the basis of who we are. We want to develop this system based on **Raspberry-pi 4**, to make the house only accessible when your face is recognized by the recognition algorithms from **Open CV library** and meanwhile you are allowed in by the house owner, who could monitor entrance remotely. Whenever the person comes in front of the door, it recognizes the face and if it is registered then it unlocks the door, if the face is not registered it will raise an alarm in the mobile and clicks a picture and send it on the registered number. This is how the system works. Face recognition system is broadly used for human identification because of its capacity to measure the facial points and recognize the identity in an unobtrusive way. The application of face recognition systems can be applied to surveillance at home, workplaces, and campuses, accordingly. The problem with existing face recognition systems is that they either rely on the facial key points and landmarks or the face embeddings from FaceNet for the recognition process.

Keywords: Raspberry-pi 4, OPENCV, SSD(Single Shot Detector), FaceNet, local binary pattern(LBP).

1. INTRODUCTION

Biometrics is unique to an individual and is used in many systems that involve security. In the face recognition approach, a given face is compared with the faces stored in the database in order to identify the person. The aim is to search out a face in the database, which has the highest similarity with the given face. This paper deals with the idea of secure locking automation utilizing RPi for door unlocking system to provide essential security to our homes, bank lockers and related control operations and security caution through the GSM module/SMTP Module .It uses an image capturing technique in an embedded system based on raspberry pi server system. RPi (Raspberry pi) controls the video camera for catching it for turning on a relay for door unlocking. The module contains a secured face recognizer for automatic door unlocking. The camera catches the facial picture and compares it with the image which is stored in the database. If the picture is found in the database then the door lock opens otherwise it will produce a SMS(Telegram) or an E-MAIL to owner that an unknown person is trying to gain access. Most of the face recognition systems available today work under controlled environment. Variation in lighting, pose, facial expression, occlusion, ageing etc., are some of the key factors that greatly influence the accuracy and efficiency of face detection and recognition.

Now-a-days door locks can be accessed using systems that are incorporated with fingerprint sensors, passwords, RFIDs, face recognition etc. The system comprises a webcam to detect the faces and a solenoid door lock for unlocking the door. Every users detected by the webcam will be checked for compatibility with the database in the system. If the face is recognized, the password box will appear. The user will enter the password. If the password is correct, the door will open automatically. On the other hand, GSM module/SMTP Package(in python) will send the notification to the owner for the unknown person. The main control circuit on this system is Raspberry Pi. The software used is OpenCV Library &Python

1.1 Basic Concepts

DeNoising: The underlying goal of DeNoising is to estimate the original image by suppressing noise from a noise-contaminated version of the image.

Sharpening: Sharpening an image involves improving the details and visibility

OpenCV: It is a free and open source library which is used for manipulating images and it also provides methods to perform tasks like face detection, object tracking, landmark detection etc.

Raspberry PI-4: Raspberry Pi is a series of small single-board computers (SBCs) The original model became more popular than anticipated, selling outside its target market for uses such as robotics. It is widely used in many areas, such as for weather monitoring, because of its low cost, modularity, and open design. It is typically used by computer and electronic hobbyists, due to its adoption of HDMI and USB devices.

Convolutional Neural Network: A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.

Single Shot Detector -Multi Box: The tasks of object localization and classification are done in a *single forward pass* of the network. the name of a technique for bounding box regression developed by Szegedy et al.

1.1 Motivation

Visualizing the future of Home Automation by implementing an Advanced Lock Management System in a Multiple access Environment.

1.2 Village visited: Chinna Avutapalle, Gannavaram, Vijayawada, Krishna District, Andhra Pradesh

1.3 Photo with the client: The figure 1.2 depicts the photo with client.

Figure 1.1 Photo with client

1.4 Problem Statement

Build, Design and implement an intuitive and Advanced IoT/AI based Lock Management System

(LMS)
Which aims at improving upon current home automation solutions whilst simultaneously providing high -end security and reliability to its users.

1.5 Objectives

- To provide a KEY-LESS solution for lock management of doors which will help take the Automation process one step ahead
- Able to control the Locking mechanism of multiple doors by a single admin by a single command execution.
- To implement a robust fool-proof idea of face recognition by utilizing AI Techniques

1.6 Advantages

1. The Trust Factor
2. Time Saving
3. Providing high security
4. Cost Effective
5. Face Recognition
6. Authentic Product
7. Improves Career Opportunities-promotion

1.7 Applications

1. Can be implemented for Home/ Office use.
2. Provide security for lockers storing valuable items
3. Advancing the present RFID meth

2. LITERATURE SURVEY

2.1 Image Quality Assessment for Fake Biometric Detection: Application to Iris, Fingerprint, and Face Recognition [1]

Methodology:
Animation studios render 3D scenes using a technique called path tracing which enables them to create high quality photorealistic frames. Path tracing involves shooting 1000's of rays into a pixel randomly (Monte Carlo) which will then hit the objects in the scene and, based on the reflective property of the object, these rays reflect or refract or get absorbed. The colors returned by these rays are averaged to determine the color of the pixel. This process is repeated for all the pixels. Due to the computational complexity it might take 8-16 hours to render a single frame. We implemented a neural network-based solution to reduce the time it takes to render a frame to less than a second using a generative adversarial network (GAN), once the network is trained. The main idea behind this proposed method is to render the image using a much smaller number of samples per pixel than is normal for path tracing (e.g., 1, 4, or 8 samples instead of, say, 32,000 samples) and then pass the noisy, incompletely rendered image to our network, which is capable of generating a high-quality photorealistic image.

Advantages:

- This framework can achieve better denoising performance than the state-of-the-art denoising methods.

Disadvantages:

- Initial Analysation of the cause of blurriness is required.

2.2 Three Triangle Method for Face Recognition using Dlib and OpenCV [2]

Methodology:

This study shows how useful and feasible in facial recognition a triangular feature of the face using the left and right Lateral Canthus, Nasal Bridge, and Apex of the Nose. These features are sometimes overlooked by traditional facial recognition programs. This paper explores a different approach using a triangle method for facial recognition utilizing DLIB and OpenCV to create the machine learning model. It has been found that the approach can be viable only with a certain distance of the camera to the face. However, the supervised learning of the segmenter has two critical drawbacks: Its objective function is sub-optimal and its training requires a large amount of annotation efforts. Thus, by adopting the REINFORCE algorithm, we train the segmenter so as to optimize the overall performance, i.e., we minimize the edit distance of final recognition results. Experimental results have shown that the proposed method significantly improves the performance for multilingual scripts and large character set languages without using character boundary labels.

Advantages:

- With a 92% accuracy of the machine learning model, the research has concluded that the approach is feasible

Disadvantages:

This model requires a lot of annotations for training the segmenter

2.3 Intelligent Secure Smart Locking System using Face Biometrics[3]

Methodology:

The rapid growth of technology in the modern society has raised many questions on the terms like security and privacy. Due to the evolution in the technology and industrialization the terms like security and privacy has become imperative for a common person. Authentication is a key factor which helps for the identification of authorized people and helps in eradicating fraudulent activities, robberies, and many other social crimes. Most of the crimes are due to the vulnerabilities in the door locking systems which can be easily accessible by the outsiders. Though there are solutions like smart doorbells and video streaming, which have limitations like heavy cost, complex and have loopholes in the security issues. To diminish the limitations and to enhance the security Smart door unlock systems using face recognition is proposed. The proposed system consists of a camera sensor popularly known as esp32-cam for storing the pictures of persons and for live streaming. The proposed system recognizes the face of the person standing in front of the door with the help AI-Thinker in the esp32-cam. The face of the person is compared with the faces of the authorized persons which are stored in the SD card of esp32-cam. If the person is an authorized person then the door gets unlocked which can be achieved with the hardware component solenoid lock. If the person is an unauthorized person then the door will be locked.

Advantages:

- The proposed system helps in adapting from traditional mechanical lock methods to enhanced security methods. It also helps in case of losing keys and helpful for disabled persons with easier access.
- Better accuracy than contour based recognition method.

Disadvantages:

- This method only works for single style license plates.

2.4 Deep Residual Learning for Image Recognition[4]

Methodology:

 Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.The experimental results indicate that the method using normalized correlation coefficient matching is the best choice, demonstrating a high accuracy of 95%, and the average running time of 14 milliseconds.

Advantages:

This method demonstrated a high accuracy of 95%, and the average running time of 14 milliseconds.

Disadvantages:

The template should be proper without any markings and it should not be wrinkled for proper matching.

2.5 Bluetooth Door Lock System Based on Smart Mobile Device.[5]

Methodology:
With1 the advancement of mobile Internet technology, smart home control systems are developing rapidly. The system platform design of this article is based on the Android platform. The smart phone is used as a client to receive signals through Bluetooth, and use a certain infrared protocol to send to household appliances that require action, and realize intelligent control and security protection of home appliances. The smart phone, a client receives signals through Bluetooth, sends signals which based on infrared protocol to household appliances so that we can realize intelligent control and security protection of home appliances. The results show that: The system hardware and equipment is simple, the cost is low, the system is reliable and easy to expand. With all of this, we can come to a conclusion that the hardware and equipment of the system are simple, spending little money and the system is reliable and easy to expand.

Advantages:

1. Provides an easy connection to the external Bluetooth module by using the mobile's Bluetooth technology
2. Easily able to control the Arduino Board using the established Bluetooth connection.

Disadvantages:
If the Captcha contains too many strike over lines, then both models fail to determine the actual text

2.6 Automatic Recognition of Traffic Signs Based on Visual Inspection.[6]

Methodology:

The automatic recognition of traffic signs is essential to autonomous driving, assisted driving, and driving safety. Currently, convolutional neural network (CNN) is the most popular deep learning algorithm in traffic sign recognition. However, CNN cannot capture the poses, perspectives, and directions of the image, nor accurately recognize traffic signs from different perspectives. To solve the problem, the authors presented an automatic recognition algorithm for traffic signs based on visual inspection. For the accuracy of visual inspection, a region of interest (ROI) extraction method was designed through content analysis and key information recognition. Besides, a Histogram of Oriented Gradients (HOG) method was developed for image detection to prevent projection distortion. Furthermore, a traffic sign recognition learning architecture was created based on CapsNet, which relies on neurons to represent target parameters like dynamic routing, path pose and direction, and effectively capture the traffic sign information from different angles or directions. Finally, our model was compared with several baseline methods through experiments on LISA (Laboratory for Intelligent and Safe Automobiles) traffic sign dataset. The model performance was measured by mean average precision (MAP), time, memory, floating point operations per second (FLOPS), and parameter number. The results show that our model consumed shorter time yet better recognition performance than baseline methods, including CNN, support vector machine (SVM), and region-based fully convolutional network (R-FCN) ResNet 101.

Advantages:

2.4.1 This model consumed shorter time yet better recognition performance than baseline methods, including CNN, support vector machine.

Disadvantages:

This method only works for single style license plates

2.7 Two-Step CNN Framework for Text Line Recognition in Camera-Captured Images[7]

Methodology:

They introduce a “on the device” text line recognition framework that is designed for mobile or embedded systems. We consider per-character segmentation as a language-independent problem and individual character recognition as a language-dependent one. Thus, the proposed solution is based on two separate artificial neural networks (ANN) and dynamic programming instead of employing image processing methods for the segmentation step or end-to-end ANN. To satisfy the tight constraints on memory size imposed by embedded systems and to avoid overfitting, we employ ANNs with a small number of trainable parameters. The primary purpose of our framework is the recognition of low-quality images of identity documents with complex backgrounds and a variety of languages and fonts. We demonstrate that our solution shows high recognition accuracy on natural datasets even being trained on purely synthetic data. We use MIDV-500 and Census 1961 Project datasets for text line recognition. The proposed method considerably surpasses the algorithmic method implemented in Tesseract 3.05, the LSTM method (Tesseract 4.00), and unpublished method used in the ABBYY FineReader 15 system. Also, our framework is faster than other comparable solutions. We show the language-independence of our segmenter with the experiment with Cyrillic, Armenian, and Chinese text lines.

Advantages:
This Framework demonstrates the powerful capabilities of employing the FCNs for text line segmentation.

It uses extremely light-weight ANNs for camera‑captured image recognition

Disadvantages:
This method only works for single style license plates
If the Captcha contains too many strike over lines, then both models fail to determine the actual text

3. ANALYSIS AND DESIGN

This section explains about hardware and Software requirements required for this project.

A functional requirement document defines the functionality of a system or one of its subsystems. It also depends upon the type of software, expected users and the type of system where the software is used.
Functional user requirements may be high-level statements of what the system should do but functional system requirements should also describe clearly about the system services in detail.

3.1 HARDWARE REQUIREMENTS

Raspberry pi 4 (Buster OS install)
Solenoid Lock
Pi Camera ver 2.0 Passive IR Sensor Capacitor (Batterey)
LED (optional, only for user notification) Jumper Wires
Relay Module 2-channel

3.2 SOFTWARE REQUIREMENTS

Python 3.9
TensorFlow 2.2.0

Python is a general-purpose, versatile, and powerful programming language. It's a great first language because it's concise and easy to read. Whatever you want to do, Python can do it. From web development to machine learning to data science, Python is the language for you.. It is simple, yet powerful. Python is easy to write, and simple to understand. This behavior makes it intuitive

NumPy

NumPy is a basic level external library in Python used for complex mathematical operations. NumPy overcomes slower executions with the use of multidimensional array objects. It has built-in functions for manipulating arrays. We can convert different algorithms to can into functions for applying on arrays. 28 NumPy has applications that are not only limited to itself. It is a very diverse library and has a wide range of applications in other sectors. NumPy can be put to use along with Data Science, Data Analysis and Machine Learning. It is also a base for other python libraries. These libraries use the functionalities in NumPy to increase their capabilities. Arrays in NumPy are objects.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

TENSORFLOW

TensorFlow is a Python library for fast numerical computing created and released by Google.
It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.
TensorFlow is an open source library for fast numerical computing

KERAS

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

Visual Studio Code

A standalone source code editor that runs on Windows, macOS, and Linux. The top pick for JavaScript and web developers, with extensions to support just about any programming language. It provides a wide variety of Extensions to be used with any programming languages. You can configure live servers, Manage Databases, Create File, Folders all at one place without frequently moving out of the working window

3.1 Functional Requirements

1. Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

2. PIL:

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool**.**

3. Sklearn:

Scikit-learn (formerly scikits. learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project

4. Pandas:

Pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulatingnumerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Non-Functional Requirements

Processor: i3 and above

RAM: 4GB and above

Operating System: Any operating system

3.2 Design Diagrams

Use case diagram:

An effective use case diagram can help your team discuss and represent:

1. Scenarios in which your system or application interacts with people, organize, or external systems.
2. Goals that your system or application helps those entities achieve
3. The scope of your system.

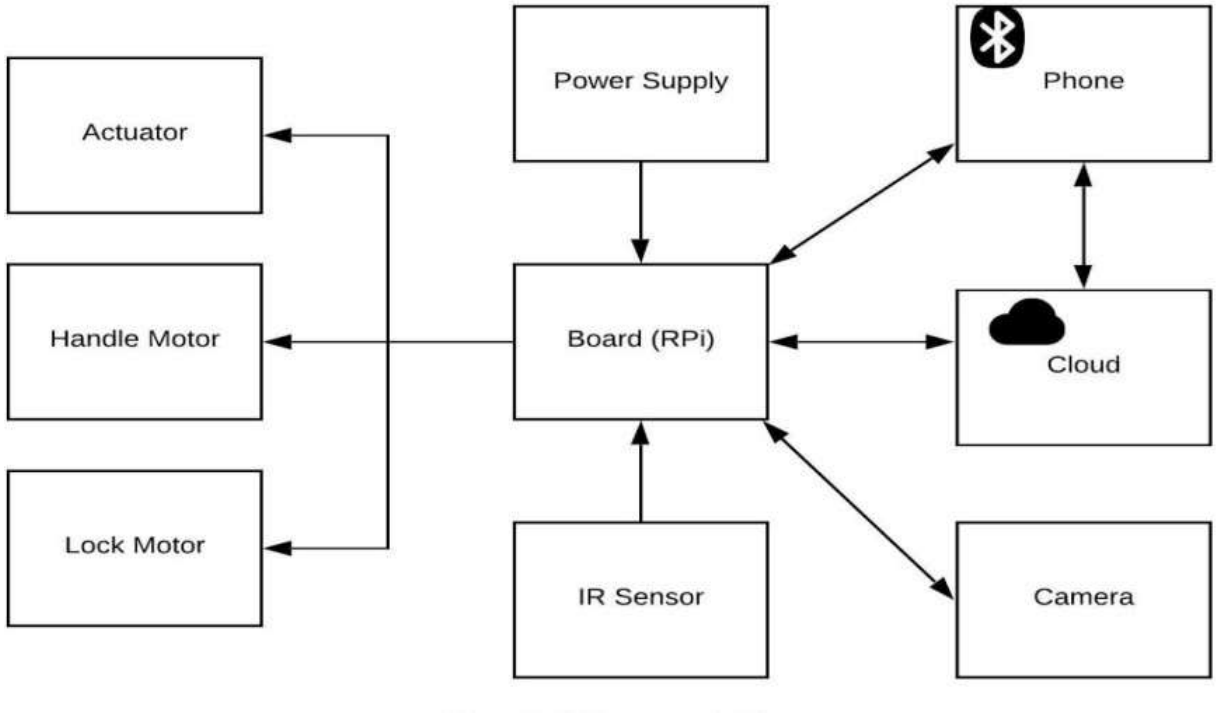

Figure 3.1 Use case diagram

4. PROPOSED SYSTEM DIAGRAM

4.1 Process Flow Diagram

The figure 4.1 depicts the process flow diagram

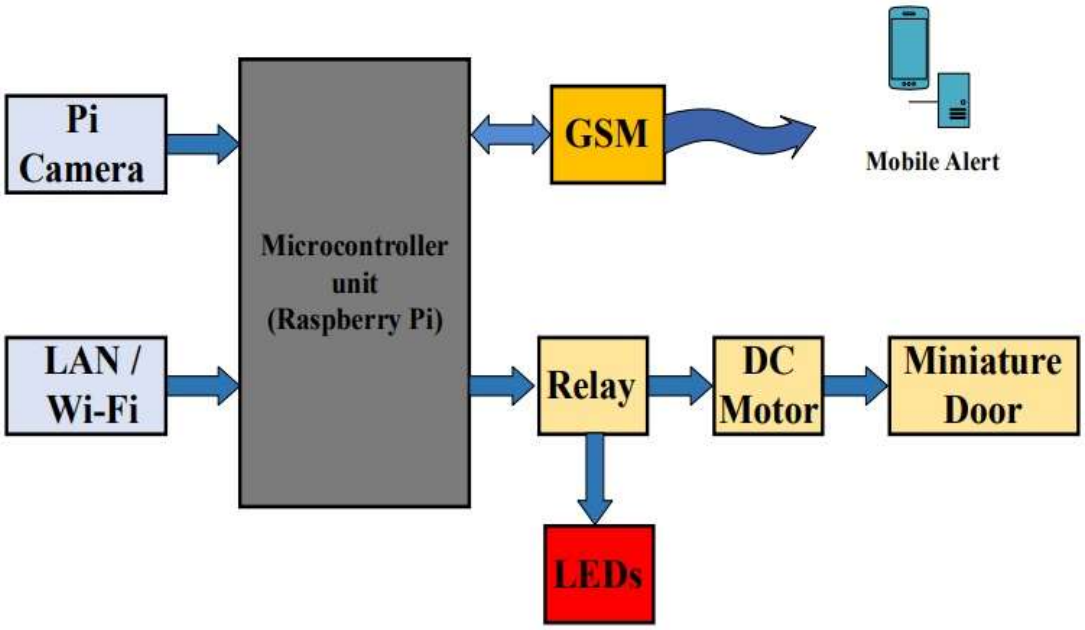

Figure 4.1 Process Flow diagram

4.2 METHODOLOGY

In this system, we are using Local Binary Pattern Histogram (LBPH) for face recognition. Open CV is an open source computer vision library that has three built-in face recognition algorithms, (i)Haar-Cascade Classifier, (ii) SSD-MultiBox, (iii) Local Binary Pattern Histogram (LBPH). Compared with the two algorithms, the LBPH can not only recognize the front face, but also recognize the side face, which is more flexible. Therefore, the face recognition algorithms used here is Local Binary Pattern Histogram (LBPH). It is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. And then it converts the binary number into a decimal number, and that decimal number is the new value of the centrepixel. At first, we have to save images by using data sets and after that, we will train that faces to LBPH algorithm then it stores into the database. At first, it converts colour images to gray scale images and then it converts into pixels for detecting this will divide the image into various pieces then it stores the values of each pixel. If pixels are less than it will be represented as 0 and pixels which are high will be 1 then it will be arranged in 3 × 3 matrix format for recognizing the images on screen compared to database stored images. The proposed works are as follows:

- Interfacing of camera to capture live face images.
- Create a database of authorized person if they exist.
- Capturing current image, save it and compare with the database image.
- If detected false individual, send the image captured to the owner via email.
- If Successful recognition, Unlock the door by controlling electronic lock.
- The project can also be used for surveillance. For instance, it can capture the images of unidentified individuals and store it which can later be used to determine the impostors who tried to gain illegitimate access.

4.3. ARCHITECTURE

Our project has a simplified architecture where a single Raspberry pi is connected to couple of sensors like IR. When a person is detected as real by using the Liveness Detector, the program then transfers to detect facial recognition of the person. The face live-ness detector uses Haar-Cascade classifier and CNN
;whilst the face recognizer uses SSD to extract facial embeddings into the '.pickle' file.

Figure 4.2 Face Recognition Process Flow

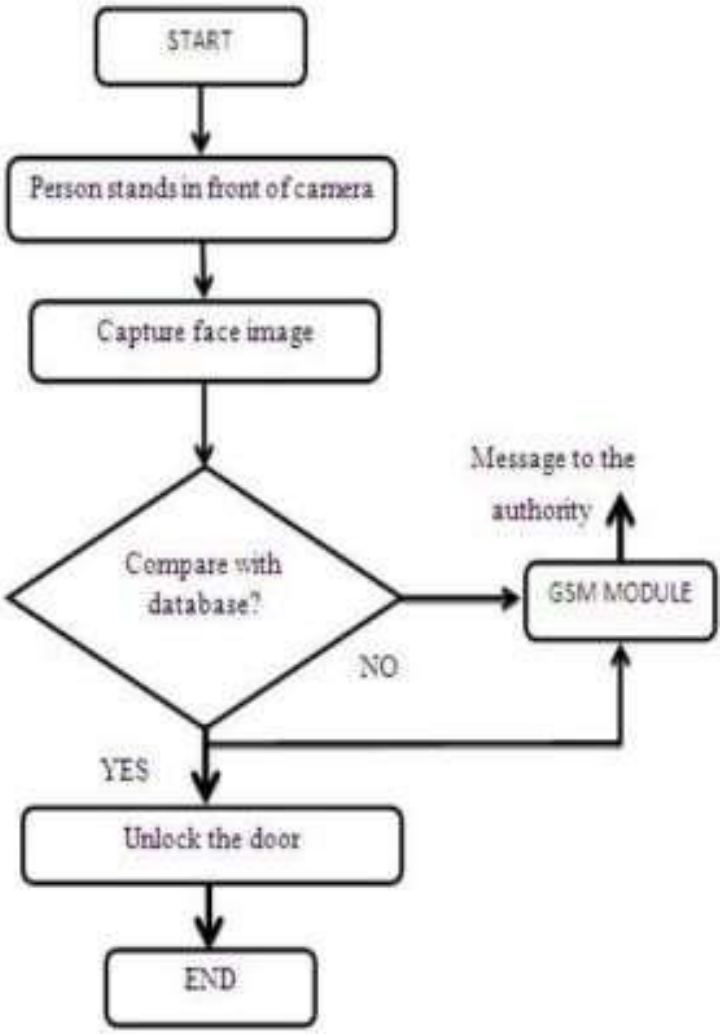

Figure 4.3 Face Process diagram

5.1 Algorithm for face detection

- Import all the required packages.

- First, we need to load the necessary XML classifiers and load input images (or video) in grayscale mode.

- After converting the image into grayscale, we can do the image manipulation where the image can be resized, cropped, blurred, and sharpen if required. The next step is image segmentation; identify the multiple objects in the single image, so the classifier quickly detects the objects and faces in the picture.

- The haar-Like feature algorithm is used to find the location of the human faces in frame or image. All the Human faces have some common universal properties of faces like the eye region is darker than it's neighbor's pixels and nose region is more bright than the eye region.

- In this step, we extract the features from the image, with the help of edge detection, line detection, and center detection. Then provide the coordinate of x, y, w, h, which makes a rectangle box in the picture to show the location of the face. It can make a rectangle box in the desired area where it detects the face.
-

2. **Algorithm for unlocking door**
    - First,when the face is successfully detected,THE GPIO pin is set to high, which triggers the RELAY module connected to the electronic lock
    - The lock is then opened for a few seconds (wait_TIME()=20), after which the door gets locked again

Recognize.py

```
import argparse
import imutils
import pickle import
cv2 import os
import time
    import subprocess
    import logging


    logging.basicConfig(filename='faceunlock.log', level=logging.INFO, filemode='w', format='%(asctime)s
    - %(levelname)s - %(message)s')

    logging.info('Starting program')

    recognized = False

    videoobj=cv2.VideoCapture(0)
    capturing=True
    while(capturing):
       ret,frame=videoobj.read()
       cv2.imshow('abcd',frame)
       cv2.imwrite('/home/pi/FaceUnlock2/images/Image.jpg',frame)
       capturing=False
    videoobj.release()
    cv2.destroyAllWindows()


    # construct the argument parser and parse the arguments ap
    = argparse.ArgumentParser()
    ap.add_argument("-i", "--image", required=True,
          help="path to input image")
    ap.add_argument("-d", "--detector", required=True, help="path
          to OpenCV's deep learning face detector")
    ap.add_argument("-m", "--embedding-model", required=True, help="path
          to OpenCV's deep learning face embedding model")
    ap.add_argument("-r", "--recognizer", required=True,
          help="path to model trained to recognize faces")
    ap.add_argument("-l", "--le", required=True,
          help="path to label encoder")
    ap.add_argument("-c", "--confidence", type=float, default=0.9,
          help="minimum probability to filter weak detections")
    args = vars(ap.parse_args())

    # load our serialized face detector from disk
    print("[INFO] loading face detector...")
    protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
    modelPath = os.path.sep.join([args["detector"],
          "res10_300x300_ssd_iter_140000.caffemodel"]) detector
    = cv2.dnn.readNetFromCaffe(protoPath, modelPath)
```



```
    # load our serialized face embedding model from disk
    print("[INFO] loading face recognizer...")
```

```
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])

# load the actual face recognition model along with the label encoder
recognizer = pickle.loads(open(args["recognizer"], "rb").read())
le = pickle.loads(open(args["le"], "rb").read())


# load the image, resize it to have a width of 600 pixels (while
# maintaining the aspect ratio), and then grab the image dimensions
while True:
    image = cv2.imread(args["image"]) image
    = imutils.resize(image, width=600) (h, w) =
    image.shape[:2]

    # construct a blob from the image
    imageBlob = cv2.dnn.blobFromImage(
            cv2.resize(image, (300, 300)), 1.0, (300, 300),
            (104.0, 177.0, 123.0), swapRB=False, crop=False)

    # apply OpenCV's deep learning-based face detector to localize #
    faces in the input image
    detector.setInput(imageBlob)
    detections = detector.forward()

    # loop over the detections
    for i in range(0, detections.shape[2]):
            # extract the confidence (i.e., probability) associated with the #
            prediction
            confidence = detections[0, 0, i, 2]

            # filter out weak detections
            if confidence > args["confidence"]: #recognizes likely face
                    # compute the (x, y)-coordinates of the bounding box for the #
                    face
                    logging.info('Face detected')
                    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                    (startX, startY, endX, endY) = box.astype("int")

                    # extract the face ROI
                    face = image[startY:endY, startX:endX]
                    (fH, fW) = face.shape[:2]

                    # ensure the face width and height are sufficiently large if
                    fW < 20 or fH < 20:
                            continue

                    # construct a blob for the face ROI, then pass the blob
                    # through our face embedding model to obtain the 128-d #
                    quantification of the face
                    faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255, (96, 96),
                            (0, 0, 0), swapRB=True, crop=False)
                    embedder.setInput(faceBlob)
                    vec = embedder.forward()
```

```
            # perform classification to recognize the face
            preds = recognizer.predict_proba(vec)[0]
            j = np.argmax(preds)
            proba = preds[j]
            name = le.classes_[j]
            if str(name) == "sam" and proba > 0.90: #recognized face and 90% probability it's my face
               recognized = True
               print("True")
               print(str(proba)) #changing from confidence to proba
               percent_confidence = round(proba * 100,1) logging.info('Sam
               detected '+str(percent_confidence))
               speech = "I am " + str(percent_confidence) + "% sure you are Sam"
               speech2 = "Welcome home"
               print(speech+"\n"+speech2)
               time.sleep(10)
            else:
               recognized = False print("Unrecognized")

            # draw the bounding box of the face along with the associated #
            probability


            text = "{}: {:.2f}%".format(name, proba * 100)
            y = startY - 10 if startY - 10 > 10 else startY + 10 cv2.rectangle(image,
            (startX, startY), (endX, endY),
                  (0, 0, 255), 2)
            cv2.putText(image, text, (startX, y), cv2.FONT_HERSHEY_SIMPLEX,
                  0.45, (0, 0, 255), 2)

            cv2.imwrite("CV2_Result.png",image);
            cv2.imshow("Image", image)
            cv2.waitKey(0)


   videoobj=cv2.VideoCapture(0)
   capturing=True
   while(capturing):
      ret,frame=videoobj.read()
      cv2.imwrite('/home/pi/FaceUnlock2/images/Image.jpg',frame) capturing=False
   videoobj.release()
   cv2.destroyAllWindows() #print("new
   capture")
```

Train.py

```
from sklearn.preprocessing import LabelEncoder from
sklearn.svm import SVC
```

```
import
argpars
e import
pickle

# construct the argument parser and parse the
arguments ap = argparse.ArgumentParser()
ap.add_argument("-e", "--embeddings",
        required=True, help="path to
        serialized db of facial embeddings")
ap.add_argument("-r", "--recognizer",
        required=True, help="path to output model
        trained to recognize faces")
ap.add_argument("-l", "--le",
        required=True,
        help="path to output
        label encoder")
args = vars(ap.parse_args())

# load the face embeddings
print("[INFO] loading face embeddings...")
data = pickle.loads(open(args["embeddings"], "rb").read())

# encode the labels
print("[INFO]
encoding labels...") le
= LabelEncoder()
labels = le.fit_transform(data["names"])

# train the model used to accept the 128-d embeddings of
the face and # then produce the actual face recognition
print("[INFO] training model...")
recognizer = SVC(C=1.0, kernel="linear", probability=True)
recognizer.fit(data["embeddings"], labels)

# write the actual face recognition
model to disk f =
open(args["recognizer"], "wb")
f.write(pickle.dumps(recognizer))
f.close()

# write the label
encoder to disk f =
open(args["le"],
"wb")
f.write(pickle.dum
ps(le)) f.close()
```

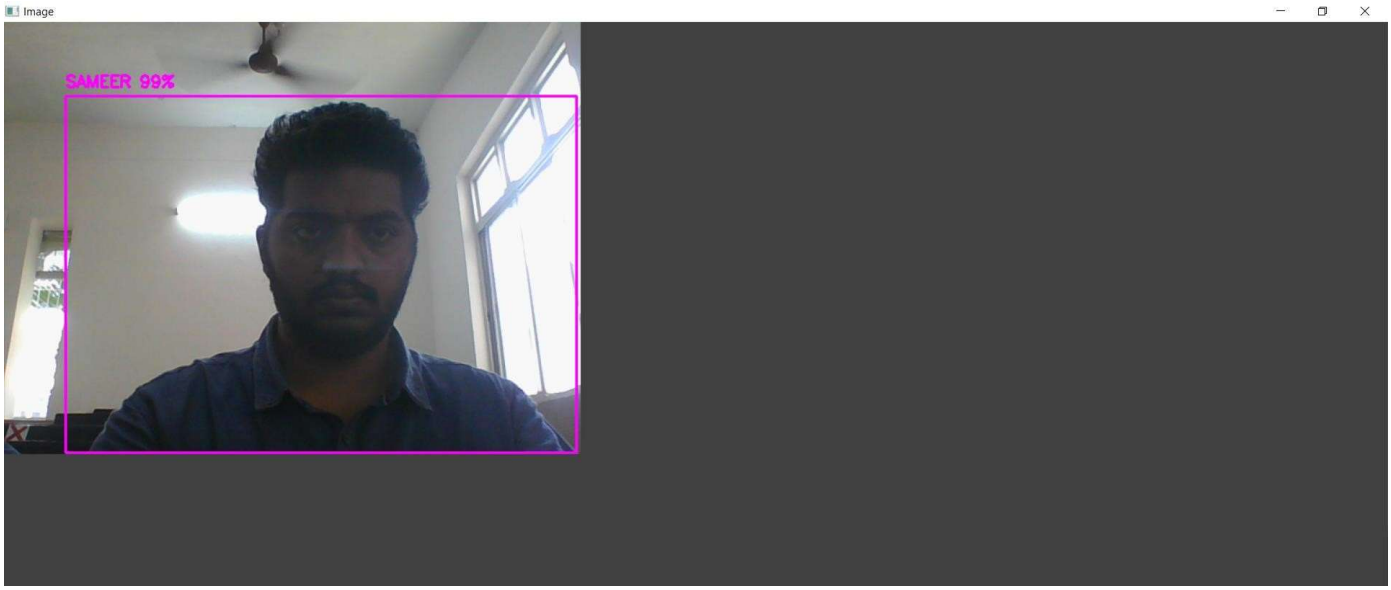

Figure 5.1 Result as recognized as authorized person

Figure 5.2 Result as recognized as authorized person

5 . CONCLUSION & FUTURE WORK

The proposed platform takes the advantage of the computer vision techniques in order to create a trustable certification verification system. As a proof of concept, we presented a prototype implementation of the system platform which is based on the openCV platform. The proposed system platform addresses a globally unified viewpoint for students and organizations. An application program with ability to upload to the database like Amazon aws or Azure can be implemented to allow seamless cloud access, rather than offline dataset access. An app is built for the owner to allow overall access to the Lock Management System which can unlock the door remotely within the app.

References

- [1]. X. Luo, J. Li and L. Zhen, “Design and implementation of a card reader based on build-in camera”, International Conference on Pattern Recognition, vo13, 2004, pp. 417-420.
- [2]. P.Jasmine Lois Ebenezer and S.Cephas, “Detecting Forged Scan of Certificates using GLCM and SVD Algorithms”, IJCRT, Vo19, 2021, pp.1-10.
- [3]. url: https://pyimagesearch.com/2020/05/25/tesseract-ocr-text-localization-and-detection , last accessed on: 16/02/2022.
- [4]. S. D. Connell and A. K. Jain, "Template-based online character recognition", *Pattern Recognition.,* vol. 34, no. 1, pp. l-14, Jan. 2001.
- [5] S. Du, M. Ibrahim, M. Shehata and W. Badawy, "Automatic License Plate Recognition (ALPR): A State-of-the-Art Review," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, no. 2, pp. 311-325, Feb. 2013, doi: 10.1109/TCSVT.2012.2203741
- [6] Shin, W., Yeom, H. G., Lee, G. H., Yun, J. P., Jeong, S. H., Lee, J. H., ... & Kim, B. C. (2021). Deep learning-based prediction of necessity for orthognathic surgery of skeletal malocclusion using cephalogram in Korean individuals. *BMC Oral Health*, *21*(1), 1-7.
- [7]Yang, J., Ling, X., Lu, Y., Wei, M., & Ding, G. (2001). Cephalometric image analysis and measurement for orthognathic surgery. *Medical and Biological Engineering and Computing*, *39*(3), 279-284.