

FACE RECOGNITION BASED DOOR LOCK USING HAAR CASCADE AND LOCAL BINARY PATTERN HISTOGRAM ON RASPBERRY PI

A Project report submitted in partial fulfillment of the requirements for

the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

P.Adithya (319126512167)

R.SrinivasaRao(319126512176)

Y.V.D.Bhavani (320126512L21)

M.Tarun(319126512165)

Under the guidance of

Mrs.D.Nagamani M.tech(Phd)

Assistant Professor



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, bheemili mandal, visakhapatnam dist.(A.P)

2022-2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

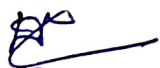
Sangivalasa, Bheemili mandal, Visakhapatnam dist.(A.P)



CERTIFICATE

This is to certify that the project report entitled "FACE RECOGNITION BASED DOOR LOCK USING HAAR CASCADE AND LOCAL BINARY PATTERN HISTOGRAM ON RASPBERRY PI" submitted by P.Adithya (319126512167), R.Srinivasa Rao (319126512176), Y.V.D.Bhavani (320126512L21), M.Tarun (319126512165) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering of Anil Neerukonda Institute of technology and Sciences(A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide



Mrs.D.Nagamani

M.Tech(Phd)

Assistant Professor

Department of E.C.E

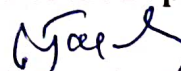
ANITS

Assistant Professor
Department of E.C.E.

Anil Neerukonda

Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

Head of the Department



Dr. B.Jagadeesh

B.E, M.E, Ph.D, FIE, FIETE, MIEEE

Professor & HOD

Department of E.C.E

ANITS

Head of the Department

Department of E C E

Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Mrs.D.Nagamani** , Assistant Professor, Department of Electronics and Communication Engineering, ANITS, for her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. B.Jagadeesh**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

P.Adithya	(319126512167)
R.SrinivasaRao	(319126512176)
Y.V.D.Bhavani	(320126512L21)
M.Tarun	(319126512165)

ABSTRACT

Now a days security is most important in every aspect. In terms of house and office security, the door is crucial. To keep the residence secure, the owner will keep the door locked at all times. To ensure security, a smart Door Lock security system with face recognition using Raspberry pi can be employed. When a person approaches the door, the device photographs their face. Then, to decide if access should be permitted, the facial recognition algorithm compares the image to a database of authorized individuals. If the person has permission, the door lock is opened, allowing them to enter. If the person is not authorized, a photo is sent to the owner's phone so that owner can remotely open or close the door. The proposed system uses Machine learning, Computer vision and Embedded systems to implement the smart door lock.

Keywords:

Embedded Systems, Machine learning, Computer vision

TABLE OF CONTENTS

ACKNOWLEDGEMENT	ii
ABSTRACT	iii
TABLE OF FIGURES	vii
Chapter 1: Introduction	1
Chapter 2: Literature Survey	4
Chapter 3: Haar Cascade classifier	7
3.1. Introduction to Haar cascade classifier:	7
3.3. Using Haar cascades in OpenCV	9
3.5. Implementation	10
3.6. Adaboost Technique	11
3.7. Attentional cascade	13
Chapter 4: Local binary pattern histogram Face Recognition Algorithm	14
4.1. Pre-Requisites	14
4.1.1. Digital Image	14
4.1.2. Pixel	15
4.1.3. Object Detection	16
4.1.4. Face Detection	17

4.3. Parameters	18
4.4. Training the Algorithm	19
4.5. Applying the LBP operation	20
4.6. Extracting the Histograms	22
4.7. Performing the face recognition	23
4.8. LBPH Application Areas	24
4.9. Advantages of LBPH Algorithm	25
Chapter 5: Methodology	26
5.1. Methodology for Hardware setup	28
5.2. Methodology for Software	29
Chapter 6: Software and Hardware Used	31
6.1. Python	31
6.2. VNC Viewer	31
6.3. Libraries	32
6.3.1. Open CV	32
6.3.2. Numpy	33
6.3.3. SYS Module	34
6.3.4. Time Module	35
6.3.5. PIL	36

6.3.6. Pickle Module	37
6.3.7. Telepot	38
6.4. Hardware	39
6.4.1. Raspberry pi 4	39
6.4.2. USB Camera	41
6.4.3. Power suppl	42
6.4.4. Solenoid lock	43
Results	46
Conclusion	48
Future Scope	48
REFERENCES	49

TABLE OF FIGURES

Figure 1: features of haar cascade	11
Figure 2: Digital image	14
Figure 3: Pixel representation	15
Figure 4: object detection using multiple frames	16
Figure 5: Steps of LBPH	18
Figure 6: LBPH operation	20
Figure 7:circular LBPH	22
Figure 8:Histogram Extraction	22
Figure 9:Flow chart of project	26
Figure 10:Sample Data Set for face recognition	28
Figure 11: Block diagram of hardware	39
Figure 12: Block diagram of raspberry pi	39
Figure 13: Raspberry Pi 4B model	41
Figure 14: USB Camera	41
Figure 15: Solenoid Door Lock	43
Figure 16: Resistor	44
Figure 17: TIP Transistor	44
Figure 18: Hardware circuit Diagram	45
Figure 19: Result of known person	46
Figure 20: Result of unknown person	47

ABBREVIATIONS

AI:	Artificial Intelligence
ANN:	Artificial Neural Networks
API:	Application Programming interface
CCTV:	closed circuit television
CNN:	Convolution neural networks
CPU:	Central Processing unit
DWT:	Discrete Wavelet Transform
GPIO:	General Purpose input output
GUI:	Graphical user interface
HOG:	Histogram of oriented gradients
ID:	Identification
IoT:	Internet of Things
JPEG:	Joint Photographic experts group
LBP:	Local binary pattern
LBPH:	Local Binary Pattern Histogram
MATLAB:	Matrix laboratory
NumPy:	Numerical python
opencv:	Open Computer vision
Pandas:	Panel dataframes
PCA:	Principal component analysis

PIL:	Python Imaging Library
PNG:	Portable Network Graphic
RBF:	Radial Basis functions
RFID:	Radio Frequency Identification
RNN:	Recurrent neural network
SoC:	System on chip
SVM:	Support vector machine
USB:	Universal serial bus
VNC:	Virtual network computing
XML:	Extensible Markup Language

Chapter 1: Introduction

INTRODUCTION:

In the modern world, there are several occurrences including robberies and unexpected entry theft. Security consequently became a key component of this lifestyle. People are constantly preoccupied with their daily tasks while simultaneously wanting to ensure the security of their most treasured possessions. Human facilities are expanding in modern society as technology advances. Due to the use of numerous technologies, people's daily lives have significantly improved. However, it also raises security concerns. They sometimes appear to lose track of essential items like keys, wallets, credit cards, etc. They are unable to enter their home or any other location they require without these. A key, a security password, an RFID card, or an ID card are all required for entry to a traditional security system. The drawbacks of these security measures include the possibility that they will be overlooked or stolen by unauthorized parties. The issue with conventional door locks is that nearly anyone can pick them and gain entry to your home. To solve these issues is therefore a huge challenge.

As a result, a better system must be created for more security. People have been employing non-living objects (such as smart cards, plastic cards, pins, tokens, and keys) for access control and authentication in restricted places for a long time. Consequently, there is a potential that someone will misplace their pins, keys, cards, etc., but if face recognition is employed for the door operating system, there is hope for improved security. The features of a person's face, which include their eyes, nose, and other distinctive features, can reveal a variety of emotions. There are two types of biometrics: behavioral traits and physiological traits (facial features, fingerprints, hand and finger shapes, palms, iris, ears, and voice). (gait, signature and keystroke dynamics). Your behavior may occasionally shift as a result of

illness, fear, hunger, etc. Compared to other biometrics, face recognition technology is more secure.

Biometrics and faces are frequently combined to identify people. Security personnel have given face recognition a lot of thought as a result of human activity discovered in a variety of security systems, including forensic, airport, face tracking, criminal detection, etc. When compared to other biometric features like palm prints, finger prints, palm prints, etc., face recognition can be utilised for security-based programmes like crime detection, face tracking, airport security, and legal investigations, among others, without the visitor's awareness. Using a webcam, a face can be captured for facial recognition. They are compared to a saved website photo and are visitor photography. Keep them on the internet and classify them into recognised groups. Keep them on the internet and classify them into recognised groups. With a variety of machine facial recognition restrictions, including changes in light, head shape, facial features, lip gloss, aging, etc., facial biometrics is a hard topic for researchers. Researchers proposed a number of strategies to refute this assertion. Face detection, face recognition, and feature removal are all included in automatic face recognition. The two categories of face detection algorithms are geometric-based elements and image-based templates. The link between a face and a template made up of one or more models is one of the template-based methods for identifying faces. Face templates are made using key component analysis, kernel approaches, and linear discrimination analysis. Specific spatial features and their geometric correlations are analyzed using geometric-based approaches.

Tools with the highest resolution, such as ridge detectors, have been found to be helpful in analyzing the information contained in images and have been beneficial in pattern identification and computer recognition.

The Object Internet Research Center has experienced multi-sector growth and development recently. The connectivity between mobile devices, cars, buildings, and other embedded things, as well as the electrical hardware, software, sensors, actuators, and network connections that enable these objects to gather and share data, is known as the Internet of Things (IoT). Traditional fields including embedded systems, wireless sensor networks,

control systems, and automation systems are brought together by IoT. As a result, the dynamic success of mobile and internet networks is the foundation for the internet of things. People use CCTV in general to secure their homes. Images are going to be stored in the database so that when any suspicious incidence occurs, action can be taken. This method of communication is passive. Therefore, the aim is to create a clever IOT-based facial recognition system that recognizes the face of someone standing close to the door and compares it to uploaded faces that are saved in the database. If a person is found, the door will open to let them in. A message containing an image of the intruder would inform the owner if someone unfamiliar tried to enter.

To create this system, we used a Raspberry Pi, a USB camera that will be placed close to the door to identify an intruder's face, and a solenoid door lock to unlock the entrance.

Chapter 2: Literature Survey

"A Novel Approach for Face Recognition Door Lock System using Haar Cascades and PCA" by R. Vigneshkumar and V. Uma. This paper proposes a face recognition door lock system using the Haar Cascade algorithm for face detection and Principal Component Analysis (PCA) for feature extraction and recognition.

"An Efficient Face Recognition Door Lock System using Hybrid Features and Support Vector Machine" by D. S. Hada and P. K. Soni. This paper proposes a face recognition door lock system using hybrid features extracted from the face image, such as Local Binary Pattern (LBP) and Discrete Wavelet Transform (DWT), and a Support Vector Machine (SVM) for classification.

"Face Recognition Door Lock System Based on Local Gradient Orientations and Deep Learning" by Y. Wang, J. Li, and Y. Zhang. This paper proposes a face recognition door lock system using local gradient orientations (LGOs) as features extracted from the face image and a deep learning-based neural network for recognition.

"A Face Recognition Door Lock System using Local Phase Quantization and Fuzzy Logic" by H. Jiang, Y. Liu, and Y. Zhang. This paper proposes a face recognition door lock system using Local Phase Quantization (LPQ) for feature extraction and Fuzzy Logic for classification.

"Face Recognition Door Lock System using Neural Network and Singular Value Decomposition" by M. A. Rahman and M. M. Hassan. This paper proposes a face recognition door lock system using a neural network and Singular Value Decomposition (SVD) for feature extraction.

"Robust Face Recognition Door Lock System using Gabor Filters and Local Ternary Patterns" by H. Zhang, Y. Zhou, and H. Zhang. This paper proposes a face recognition door lock system using Gabor filters for feature extraction and Local Ternary Patterns (LTPs) for classification.

"A Face Recognition Door Lock System based on Gabor Filters and Fuzzy Inference" by Y. Liu, Y. Zhang, and H. Jiang. This paper proposes a face recognition door lock system using Gabor filters for feature extraction and Fuzzy Inference for classification.

"Face Recognition Door Lock System using Convolutional Neural Network and Eigenfaces" by M. Hasan, M. R. Hasan, and A. H. Bhuiyan. This paper proposes a face recognition door lock system using a Convolutional Neural Network (CNN) for feature extraction and Eigenfaces for recognition.

Face Recognition using Haar Cascade and Local Binary Pattern Histogram in OpenCV

Publisher: IEEE

Aman Sharma; Khushi Shah; Salil Verma

Face recognition using Haar Cascade and Local Binary Pattern Histograms (LBPH) is a popular technique for detecting and recognizing faces.

Haar Cascade is a machine learning-based algorithm that can detect objects in an image by using a set of predefined features. It is often used for face detection, where it identifies the location of faces in an image by analyzing their features, such as the edges, corners, and lines. LBPH is a method for describing the texture of an image by dividing it into small regions and then computing a histogram of the local binary patterns within each region. It is often used for feature extraction in face recognition systems, as it can capture the texture and shape of the face.

In face recognition using Haar Cascade and LBPH, the Haar Cascade algorithm is first used to detect the face in an image. Then, the region containing the face is extracted and processed using LBPH to extract the features of the face. These features are then compared to a database of known faces to recognize the person in the image

The literature survey shows that there are many different approaches to developing a face recognition door lock system, including various feature extraction and classification techniques. Some of the most common methods include Haar cascades, PCA, SVM, deep learning-based neural networks, and Gabor filters. In addition, many researchers have explored the use of hybrid features, such as combining LBP and DWT or LGOs and deep learning. Fuzzy logic and Singular Value Decomposition (SVD) have also been used for classification.

Hence we adopted Haar cascade classifier for feature extraction and face detection and lbph algorithm for recognition. We implemented this technique using the microprocessor – Raspberry pi and other hardware.

Chapter 3: Haar Cascade classifier

3.1. Introduction to Haar cascade classifier:

Haar cascade classifier is a type of object detection method used in computer vision and image processing. It's based on the Haar wavelets and was introduced by Viola and Jones in 2001. The Haar wavelets are mathematical functions that can be used to describe local features of an image, such as edges, lines, and corners.

The Haar cascade classifier works by dividing an image into multiple stages, with each stage consisting of several Haar-like features that are used to classify whether an object is present in a given sub-window of the image. If an object is detected in one of the stages, the algorithm moves to the next stage to perform a more detailed examination of the object. This process is repeated until the final stage is reached and a decision is made on whether the object is present in the image or not.

Haar cascade classifiers have been widely used for face detection in images and videos, but they can also be used for other object detection tasks, such as detecting eyes, noses, and mouths for facial feature extraction, or detecting cars, bicycles, and pedestrians for object tracking in videos.

In summary, the Haar cascade classifier is a powerful tool for object detection in computer vision, and its fast and accurate performance has made it a popular choice for a wide range of applications.

3.2. Pre-Trained Haar cascades:

Pre-trained Haar cascade classifiers are pre-existing object detection models that have already been trained on a large dataset of images and can be used directly for object detection tasks. These pre-trained models are made available to the public and can be used as a starting point for building custom object detection systems.

One of the main advantages of using pre-trained Haar cascade classifiers is that they save a significant amount of time and resources compared to training an object detection model from scratch. Training an object detection model can take days or even weeks, depending on the size of the dataset and the complexity of the model. In contrast, pre-trained models can be used almost immediately after downloading, and can be fine-tuned or adapted for specific tasks with relatively little effort.

Another advantage of pre-trained Haar cascade classifiers is that they have already been trained on large and diverse datasets, which ensures that they have a good level of accuracy and robustness for a wide range of object detection tasks. This makes it possible to use pre-trained models as a drop-in replacement for custom object detection models, which can be particularly useful for prototyping or when there is limited time or resources available.

However, it's worth noting that pre-trained Haar cascade classifiers may not always be the best choice for a particular task, as they may not have been trained on the exact type of data that is being used. For example, pre-trained models trained on images of faces may not work well for detecting objects in videos, as the dynamics and movement in videos can be different from static images. In such cases, it may be necessary to train a custom object detection model from scratch or fine-tune a pre-trained model.

In conclusion, pre-trained Haar cascade classifiers are a useful and efficient tool for object detection in computer vision, and can be used as a starting point for building custom object detection systems. They offer a fast and cost-effective alternative to training object detection models from scratch and have the added advantage of being trained on large and diverse datasets, which ensures that they have a good level of accuracy and robustness for a wide range of object detection tasks."

3.3. Using Haar cascades in OpenCV

The Haar cascade classifier can be used in OpenCV, an open-source computer vision library, to perform object detection tasks. Here's a basic overview of how to use Haar cascade classifiers in OpenCV:

Load the Haar cascade classifier: The first step is to load the Haar cascade classifier into memory. This can be done using the `cv2.CascadeClassifier()` function in OpenCV. The function takes the path to the XML file that contains the Haar cascade classifier as an argument.

Load the input image: The next step is to load the input image that you want to perform object detection on. This can be done using the `cv2.imread()` function in OpenCV.

Convert the image to grayscale: The input image should be converted to grayscale, as the Haar cascade classifier operates on grayscale images. This can be done using the `cv2.cvtColor()` function in OpenCV.

Perform object detection: The next step is to perform object detection on the grayscale image. This can be done using the `detectMultiScale()` method of the `cv2.CascadeClassifier` class in OpenCV. This method takes the grayscale image as an argument and returns the coordinates of the objects detected in the image.

Draw bounding boxes around the objects: The final step is to draw bounding boxes around the objects detected in the image. This can be done using the `cv2.rectangle()` function in OpenCV, which takes the input image, the coordinates of the bounding box, and the color of the bounding box as arguments.

OpenCV provides pre-trained classifiers for detecting faces, eyes, smiles, etc., which are stored as XML files in the "opencv/data/haarcascades/" folder. One can create a face and eye detector using OpenCV by utilizing the "detectMultiScale" function with the following parameters:

- "image": A matrix of type CV_8U that represents the image in which objects are to be detected.

- "objects": A vector of rectangles, where each rectangle contains the detected object. Note that the rectangles may partially extend beyond the boundaries of the original image.
- "scaleFactor": A parameter that determines how much the image size is reduced at each image scale during the detection process.
- "minNeighbors": A parameter that specifies the minimum number of neighbors a candidate rectangle must have in order to be retained as a valid detection.
- "flags": A parameter with similar functionality as in the legacy "cvHaarDetectObjects" function for old cascade classifiers, but not used for new cascade classifiers.
- "minSize": The minimum size of the object that can be detected. Objects smaller than this size will be ignored.
- "maxSize": The maximum size of the object that can be detected. Objects larger than this size will be ignored. If "maxSize" is set to be equal to "minSize", the detection will be performed at a single scale.

3.5. Implementation

Haar-feature choice Dark and light sections make up a Haar-like characteristic. By comparing the difference between the sum of the intensities of the bright and dark zones, it yields a single value. In order to identify an object, valuable elements must be extracted. Viola and Jones' suggested characteristics are:

Building Integral Images In the integral image, each pixel's left and rightmost neighbours are added together to get its value. Integral Images considerably shorten the time required to perform the operation of extracting Haar-like features, which requires computing the

difference between dark and light rectangular patches.

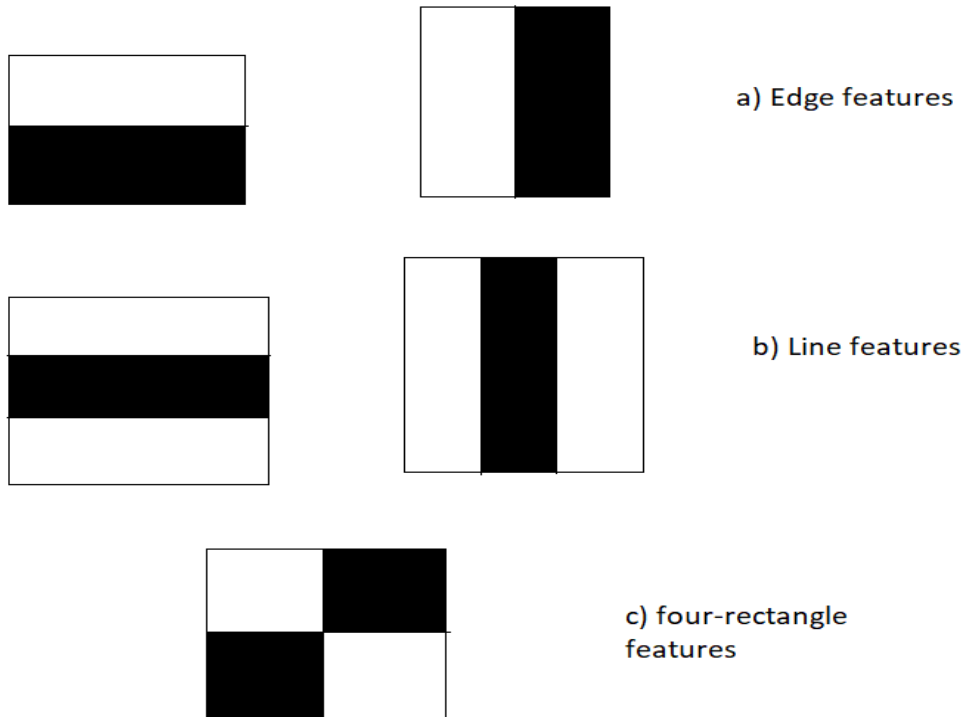


Figure 1: features of haar cascade

3.6. Adaboost Technique

There is a collection of characteristics that would accurately depict specific facial features, such as the lips, the bridge connecting the two eyes, or the brows. However, this was not the feature set's sole focus at first. About 180,000 of these made up the feature set, which was then reduced to 6000.

The majority of these features won't fit the facial features well or won't be relevant to them because they are too random to be of any use. Therefore, they need a feature selection technique to pick a smaller subset of characteristics from the large set, which would not only pick features that performed better than the rest but would also remove the unnecessary ones.

They employed a technique known as AdaBoost, It contained each of these 180,000 features separately on the photos to produce Weak Learners. Some of them produced low error rates because they distinguished between the Positive and Negative Images more well than the some didn't, while others did. These weak learners are created so that they would incorrectly classify a minimum amount of photos. They are capable of doing better than a simple guess. Their final collection of features was condensed using this method to a total of 6000 features. Adaboost, short for Adaptive Boosting, is a machine learning technique used in object detection algorithms such as Haar Cascade face detection. Adaboost combines multiple weak classifiers to form a strong classifier that can detect objects with high accuracy. In Haar Cascade face detection, Adaboost is used to train a classifier that can detect faces in images. The algorithm works by training a set of weak classifiers, which are simple classifiers that can only make binary decisions on whether an input belongs to a certain class or not.

During training, Adaboost assigns a weight to each training sample based on its importance in correctly classifying the target class. The algorithm then trains each weak classifier on a subset of the training data, with more emphasis given to the samples that were misclassified by the previous weak classifiers. This adaptive weighting of the training data allows Adaboost to focus on the samples that are most difficult to classify, improving its accuracy.

Once all the weak classifiers have been trained, Adaboost combines them into a strong classifier by assigning weights to each weak classifier based on their performance. The final classifier is a weighted combination of the weak classifiers, with more weight given to the more accurate ones.

In Haar Cascade face detection, the strong classifier generated by Adaboost is used to scan an image and detect faces by comparing different features such as the presence of eyes, nose, and mouth. The Haar-like features are used to detect the facial features based on the brightness of adjacent pixels.

3.7. Attentional cascade

The attentional cascade is a type of object detection framework that combines the principles of cascade classifiers and attention mechanisms. It is a two-stage approach where the first stage, known as the "attentional stage," quickly filters out negative regions from the input image, reducing the computational load of the second stage, known as the "classifier stage," which performs more detailed object detection. In the attentional cascade, the attentional stage uses a simple and fast feature extraction process to rapidly discard regions of the image that are unlikely to contain the target object. This is achieved by applying a set of simple, low-complexity features to each region of the image and using an attention mechanism to determine the regions that are most likely to contain the object of interest. The classifier stage is typically composed of more complex and computationally expensive classifiers, such as machine learning algorithms like Support Vector Machines (SVMs) or Convolutional Neural Networks (CNNs). The attentional cascade framework allows for faster and more efficient object detection compared to traditional cascade classifiers, as it quickly eliminates non-relevant regions from further processing, reducing computational overhead. It is particularly useful in real-time or resource-constrained applications where efficient object detection is crucial. The attentional cascade approach has been applied to various computer vision tasks, including face detection, pedestrian detection, and object recognition, and has shown promising results in terms of accuracy and efficiency.

Chapter 4: Local binary pattern histogram Face Recognition Algorithm

4.1. Pre-Requisites

The pre-requisites for LBPH algorithm are :

4.1.1. Digital Image :

A digital image is a representation of a picture or graphic that is stored in a digital format. In other words, it is a picture that is stored as a series of electronic data in a computer or other digital device.

A digital image is made up of a grid of tiny picture elements, called pixels, which are arranged in rows and columns. Each pixel contains a value that represents the colour and intensity of the light at that point in the image. Digital images can be created using digital cameras, scanners, or other image capture devices, and can be edited, processed, and manipulated using various software tools.

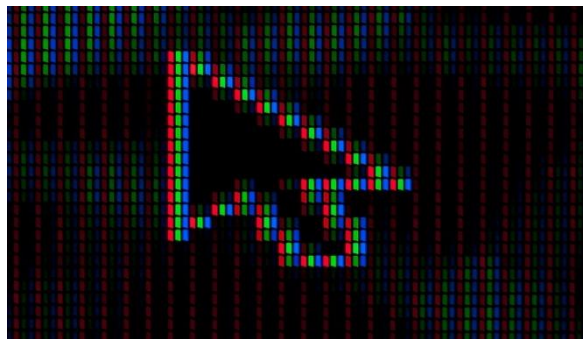


Figure 2: Digital image

4.1.2. Pixel:

“A pixel is the smallest unit of visual information in the digital world. Every digital photo, illustration, video, and game is constructed from pixels, which tend to be perfectly round or square”.

Each pixel contains a value that represents the color and intensity of light at that point in the image. In a color image, each pixel is composed of three color channels, typically red, green, and blue (RGB). The combination of these channels produces a wide range of colors and shades.

Pixels are a fundamental concept in digital imaging and are used in various applications, such as digital photography, computer graphics, and video. They can be manipulated and edited using various software tools to modify the appearance of the image, such as changing colours, adding effects, or adjusting brightness and contrast.

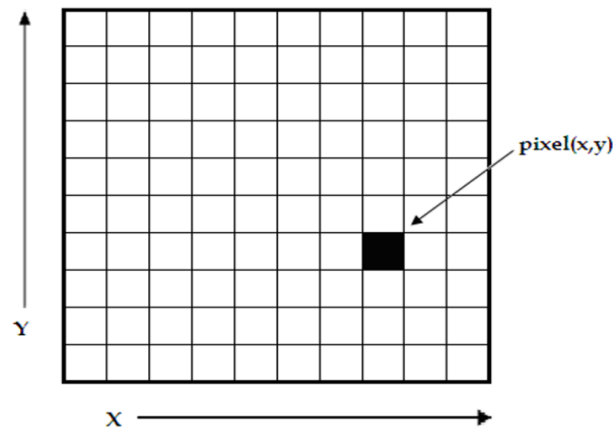


Figure 3: Pixel representation

For grey-scale images, each 8-bit pixel has an intensity value between **0–255**, in such a way that **0** represents the **black pixel**, and **255** represents **white one**.

4.1.3. Object Detection:

Object detection is a computer vision task that involves automatically identifying and localizing objects of interest within an image or video. It is a fundamental and widely used technique in various applications, such as autonomous driving, surveillance, robotics, image retrieval, and augmented reality.

Object detection typically involves two main steps: object localization and object classification. Object localization refers to determining the spatial coordinates (e.g., bounding box) of the object within the image, while object classification involves assigning a predefined label or category to the object (e.g., person, car, or dog).

Object detection has numerous applications, ranging from video surveillance and autonomous vehicles to healthcare and retail. It continues to be an active area of research, with advancements in deep learning, sensor technology, and computational power driving improvements in accuracy, speed, and robustness of object detection algorithms

- Track and segment a target object
 - Annotated by a user in the first frame

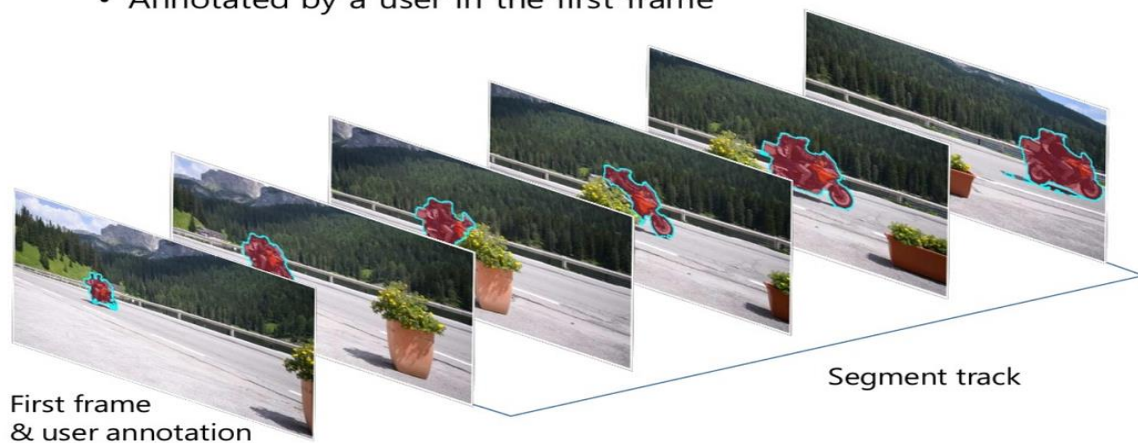


Figure 4: object detection using multiple frames

4.1.4. Face Detection:

Face detection is a technology that allows computer systems to identify and locate human faces in images or video footage. It is a subset of computer vision and image processing, and it involves the use of algorithms and machine learning models to analyze visual data and recognize facial features.

The process of face detection involves identifying regions of an image or video frame that are likely to contain a face, and then analyzing those regions in more detail to confirm the presence of a face. The algorithm then uses various features such as the shape of the face, the position of the eyes, nose, and mouth, and other distinguishing characteristics to identify the individual.

Face detection technology has a wide range of applications, including security and surveillance, social media, entertainment, and biometrics. It is used in security systems to identify individuals in public places or monitor employee attendance, in social media platforms to automatically tag individuals in photos, in entertainment industry for special effects, and in biometric authentication systems to verify the identity of an individual.

4.2. Introduction to LBPH:

Local Binary Patterns Histogram (LBPH) is a popular computer vision and image processing algorithm used for texture analysis and face recognition. It was introduced by Timo Ojala, Matti Pietikäinen, and Topi Mäenpää in 1994.

The LBPH algorithm works by extracting local binary patterns from an image or a region of interest (ROI) within an image. Local binary patterns are simple texture descriptors that capture the relationship between a central pixel and its surrounding pixels. The basic idea is to compare the intensity value of the central pixel with its neighboring pixels and encode the results into binary patterns, typically using a thresholding operation. These binary

patterns are then concatenated to form a histogram, which serves as a feature vector representing the texture of the image or ROI.

Steps:

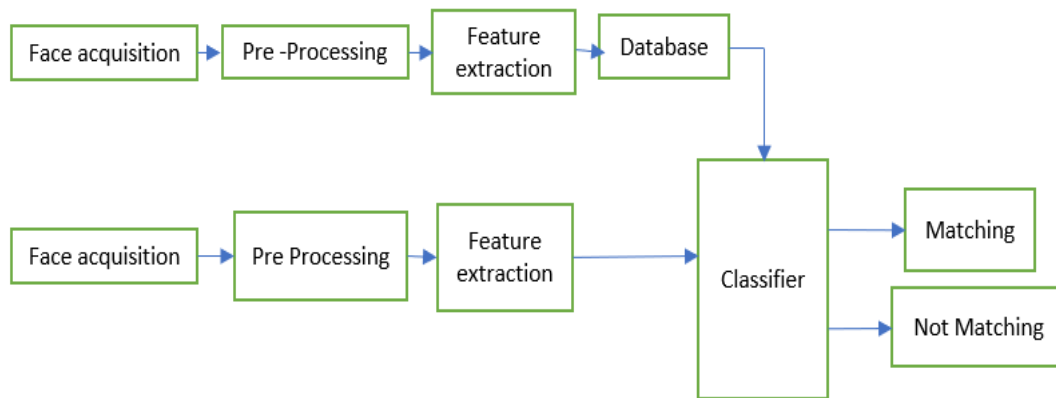


Figure 5: Steps of LBPH

4.3. Parameters:

The Local Binary Patterns Histogram (LBPH) algorithm has several parameters that can be adjusted to fine-tune its performance. Here are the key parameters of the LBPH algorithm:

Radius: Radius refers to the radius of the circular neighborhood around the central pixel for computing the local binary patterns. It determines the size of the neighborhood used to extract texture information. A larger radius includes more neighboring pixels in the pattern computation, capturing broader context but may also introduce more noise.

Number of Points: Number of Points defines the number of sampling points evenly distributed around the circumference of the circle with the specified radius. These points are used to compare intensity values with the central pixel to compute the binary patterns. A

higher number of points provides a more detailed representation of the texture but may also increase computational complexity.

Threshold: Threshold is a value used to binarize the local binary patterns into binary codes. Pixels with intensity values above the threshold are set to 1, while those below the threshold are set to 0. The choice of threshold can affect the discriminative power of the extracted features, and it may need to be experimentally tuned based on the specific application and dataset.

Grid Size: Grid Size refers to the number of cells used to divide the image or ROI for computing the histograms. LBPH can be applied to multiple cells or sub-regions within an image, and the histograms of these cells are concatenated to form the final feature vector. Grid Size determines the granularity of the spatial information captured by the histograms, with a smaller grid size capturing finer details but may increase computational cost.

4.4. Training the Algorithm:

Data Collection: Collect a dataset of images containing faces for training. This dataset should be diverse and representative of the real-world scenarios where the face recognition system will be deployed.

Preprocessing: Preprocess the images, such as resizing, cropping, and normalizing, to ensure consistent image sizes and pixel values across the dataset. It's important to preprocess the images in a way that preserves the local patterns of the faces that LBPH can capture effectively.

Feature Extraction: Apply the LBPH algorithm to each image in the dataset to extract local binary patterns from predefined regions of interest (ROIs), such as the facial region. The LBPH algorithm computes histograms of the binary patterns for each ROI, which serve as the feature vectors representing the texture information of the faces.

Labeling: Assign labels or identities to the extracted features, typically corresponding to the individuals whose faces are present in the images. This creates a labeled dataset that pairs the extracted features with their corresponding identities.

Classifier Training: Train a classifier, such as a support vector machine (SVM) or a neural network, using the labeled dataset of features as input and the corresponding labels as output. The classifier learns to map the extracted features to the corresponding identities based on the labeled dataset.

Evaluation: Evaluate the trained classifier using a separate validation or test dataset to assess its performance, such as accuracy, precision, recall, or F1 score. Adjust the parameters of LBPH, as well as the classifier, if needed, to optimize the performance of the face recognition system.

4.5. Applying the LBP operation:

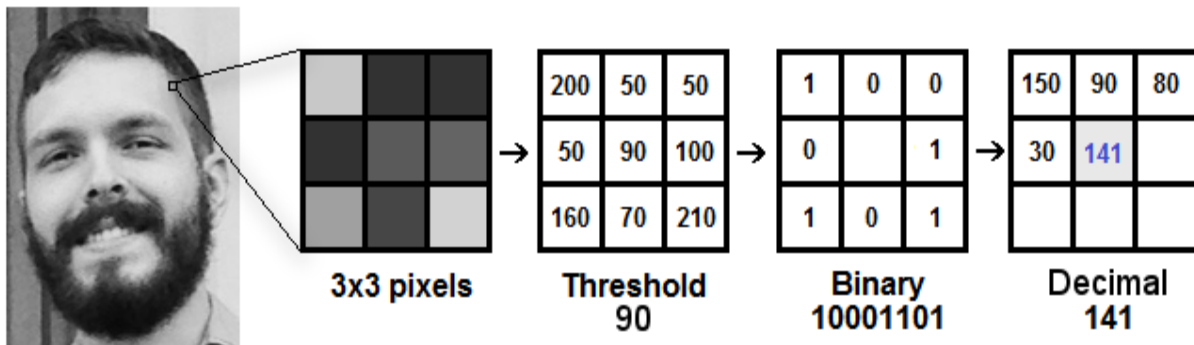


Figure 6: LBPH operation

The Local Binary Patterns Histogram (LBPH) algorithm applies the Local Binary Patterns (LBP) operation on an image by following these steps:

Image Preparation: The input image is loaded and converted to grayscale to ensure that the LBP operates on a single channel image.

Neighborhood Definition: The LBPH algorithm allows you to define the neighborhood around each pixel for which you want to compute the LBP. This neighborhood is typically defined as a circular or rectangular region centered around each pixel, with a certain radius and a number of points (P) evenly spaced along the circumference of the circle or along the sides of the rectangle.

LBP Computation: For each pixel in the image, the LBP operation is applied as follows: a. The neighborhood of the current pixel, based on the radius and number of points defined in the previous step, is selected. b. The intensity value of the central pixel is compared with the intensity values of its neighbors in the selected neighborhood. c. For each neighbor, if its intensity value is greater than or equal to the intensity value of the central pixel, a binary value of 1 is assigned, otherwise 0. d. The binary values of the neighbors are concatenated in a clockwise or counter-clockwise order to form a binary pattern. e. The binary pattern is converted to a decimal value to obtain the LBP value for the current pixel.

Image Representation: The resulting LBP values for all pixels in the image form a new grayscale image, where each pixel represents the LBP value of the corresponding pixel in the original image. This new image is called the LBP image and represents the local texture information of the input image.

Histogram Calculation: The LBPH algorithm further processes the LBP image by calculating a histogram of the LBP values. The histogram bins correspond to the possible LBP values, and the bin counts represent the frequency of each LBP value in the image. This histogram can be used as a texture descriptor to represent the texture information of the image.

Post-processing (Optional): The LBPH algorithm may also include optional post-processing steps, such as histogram normalization, histogram equalization, or other operations, depending on the specific application or requirements.

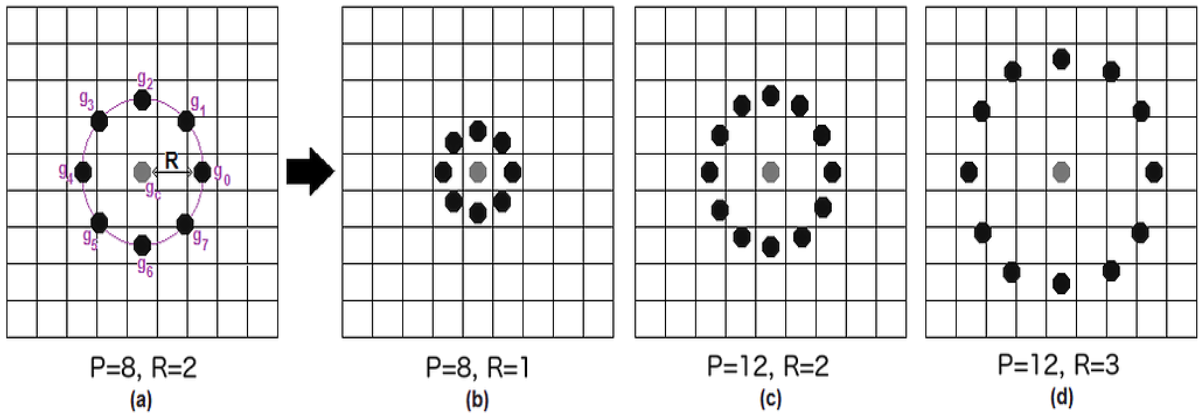


Figure 7:circular LBPH

4.6. Extracting the Histograms:

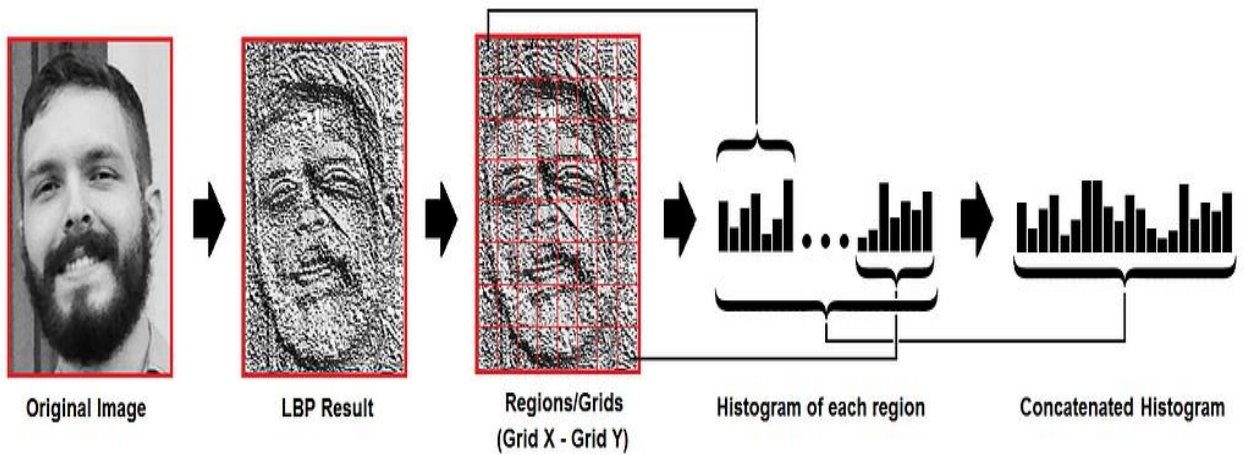


Figure 8:Histogram Extraction

Histogram is extracted in the LBPH algorithm in following steps:

LBP Computation: The LBPH algorithm computes the LBP values for each pixel in the input image, as explained in the previous responses. The LBP values represent the local texture information of the image.

Binning: The LBP values are typically represented as integer values ranging from 0 to a maximum value, depending on the number of points (P) in the neighborhood. These integer LBP values are used to populate histogram bins.

Histogram Calculation: A histogram is then computed by counting the frequency of occurrence of each LBP value in the image. Each LBP value corresponds to a bin in the histogram, and the count represents the number of times that LBP value occurs in the image.

Histogram Representation: The resulting histogram is a vector, where each bin represents a possible LBP value, and the bin count represents the frequency of occurrence of that LBP value in the image. This histogram can be used as a texture descriptor that captures the distribution of LBP values in the image, which represents the texture information of the image.

Histogram Normalization (Optional): Optionally, the histogram can be normalized to account for variations in image intensity or image size. Common normalization techniques include L1 or L2 normalization, where the bin counts are divided by the total count of LBP values or the square root of the sum of squares of the bin counts, respectively. Normalization helps to make the histogram more robust to variations in image intensity and size, and it can improve the performance of the LBPH algorithm in different lighting conditions or image scales.

4.7. Performing the face recognition:

Feature Representation: The histograms of LBP values from different facial regions are concatenated or combined to form a single feature vector that represents the entire face. This feature vector serves as a unique representation of the face's texture features.

The distance between two histograms can be calculated using a variety of methods, such as the Euclidean distance, chi-square, absolute value, etc. Based on the following formula, One can apply the well-known Euclidean distance in this example:

$$D = \sqrt{\sum_{i=1}^n (hist1i - hist2i)^2}$$

Model Training: The extracted feature vectors from the training set are used to train a face recognition model, such as a classifier or a similarity metric. The model learns to map the input feature vectors to their corresponding labels or identities, capturing the patterns in the feature space that differentiate different faces.

Model Testing: The trained model is then evaluated on the testing set to measure its performance in recognizing unseen faces. The feature vectors from the testing set are extracted using the same LBPH algorithm and fed into the trained model for prediction. The predicted labels or identities are compared with the ground truth labels to compute the recognition accuracy or other performance metrics.

Face Recognition: Once the model is trained and tested, it can be used for face recognition on new, unlabeled images. The LBPH algorithm is applied to extract the LBP features from the input face image, and the trained model is used to predict the identity or label of the person in the image based on the extracted features.

4.8. LBPH Application Areas:

The LBPH method can be used for face recognition in the following contexts:

Texture analysis is useful for both research and practical purposes like medical imaging. This has facilitated straightforward image processing through the texture segmentation of images, which has significantly advanced analysis.

employed in biometrics, such as the recognition of palm prints, fingerprints, irises, gaits, the order of placement of recognition, and in the presence of an age rating.

employed in computer vision applications like motion analysis. Computer systems can read and interpret information on images thanks to LBPH technology.

4.9. Advantages of LBPH Algorithm:

The LBP operator is resistant to monotonic grey scale modifications, making it one of the best texture descriptors.

Fisher Faces still views lighting fluctuations as a good quality even though it just prevents a person's features from becoming dominant. However, since light variation is not a component of the actual face, it is not a useful characteristic to extract. Fisher faces require a bigger storage space for face data and more processing time during recognition. While the eigenfaces and fisher faces methods examine the dataset as a whole, LBPH analyses each image separately.

The Local Binary Pattern Histogram (LBPH) algorithm is a popular and effective method for face recognition and detection in computer vision. It is a simple yet powerful algorithm that works by extracting local patterns from an image and using them to represent and classify faces.

LBPH has several advantages, including its simplicity, efficiency, and robustness to variations in lighting and facial expressions. It is also relatively easy to implement and has been used successfully in a wide range of applications, including security, surveillance, and biometric authentication.

However, LBPH is not without its limitations. It may struggle to accurately identify faces in images with complex backgrounds or when faces are partially occluded. Additionally, it may not perform as well as more advanced algorithms in scenarios with large datasets or high levels of variability.

Chapter 5: Methodology

The Haar cascade classifier is used to determine whether or not a single face has been spotted after preprocessing such as image resizing and cropping. The Haar features operating as inputs are edge, line, and center surround. The image is tested using these cascade features. The characteristics of Haar are broken down into multiple stages. If all of the processes are completed successfully, it is said that a face has been spotted and is being compared to the images previously kept in the Raspberry Pi database.

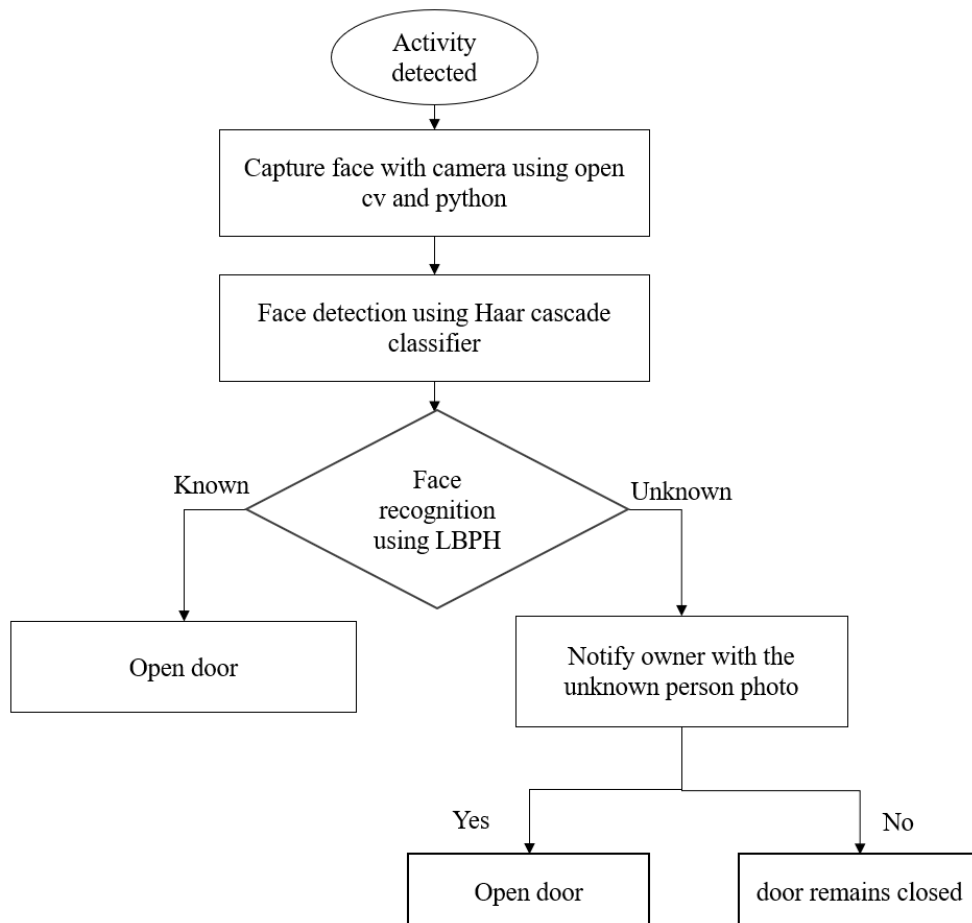


Figure 9:Flow chart of project

The flow chart contains Haarcascade and LBPH algorithms

Haar cascade

A machine learning object detection algorithm called Haar cascade is used to locate things in an image or video. Object detection is a branch of computer science that deals with finding instances of things in pictures and movies. It is related to computer vision, image processing, and deep learning. The classifier is trained using a large number of both positive and negative images in the Haar Cascade technique, which is based on machine learning.

LBPH

There are 5 steps involved in algorithm.

1) Parameters:- 1) Radius 2) Neighbour 3) Grid X 4) Grid Y.

2) Training the algorithm :

The local binary pattern histogram(LBPH) algorithm is a simple solution on face recognition , which can recognize both front face and side face.

3)LBP Operation (intermediate image using sliding window operations)grayscale->pixels or matrix or grid->threshold value->binary->decimal placed at center pixel

4)Extracting Histogram:-Using grid X,Y divide into multiple grids(0~255) based on pixel intensity.

5)Performing Face Recognition:- Checking the input image with dataset, comparing threshold value and confidence value i.e., distance between two histograms using euclid distance, chi square, absolute value.

5.1. Methodology for Hardware setup:

Hardware setup: You will need a Raspberry Pi with a camera module, a solenoid door lock to control the lock, and a power source. You will also need to physically attach the camera and solenoid door lock to the door.

Install necessary software: Install the OpenCV library and any necessary dependencies on the Raspberry Pi. This will enable you to perform face detection and recognition.

Capture faces: Capture images of faces that you want to recognize and store them in a database. You can use the Raspberry Pi camera to capture images.

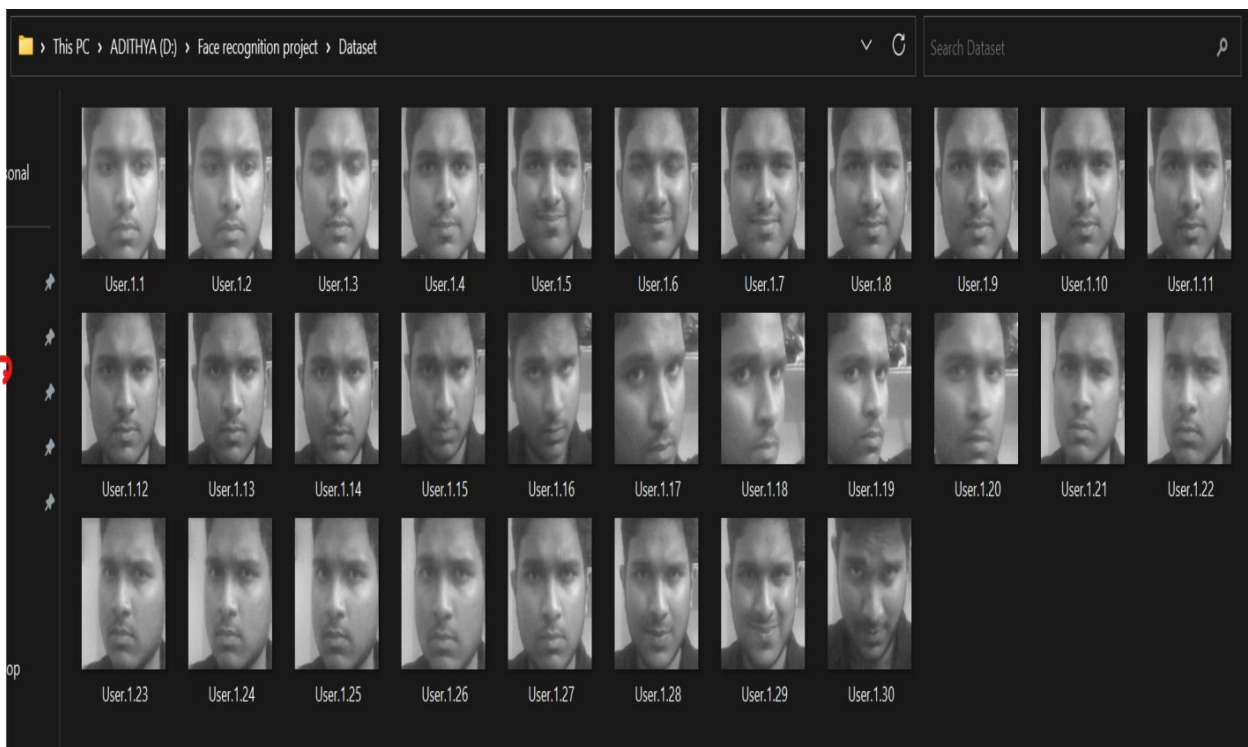


Figure 10:Sample Data Set for face recognition

Train a face recognition model: Use the captured images to train a face recognition model. This involves extracting features from each image and using those features to create a model that can identify faces.

Write code to control the lock: Write Python code that uses the Raspberry Pi GPIO pins to control the solenoid door lock. This code should be able to lock and unlock the door.

Write code to perform face recognition: Write Python code that uses the face recognition model to identify faces in real-time. When a face is identified, the code should send a signal to the solenoid door lock to unlock the door.

Integrate all the code: Combine the code for controlling the solenoid door lock and performing face recognition into a single script. This script should run automatically when the Raspberry Pi is turned on.

Test the system: Test the face recognition door lock by attempting to unlock the door with both recognized and unrecognized faces.

Improve the system: Analyse the performance of the system and make any necessary improvements to increase the accuracy of the face recognition model and the reliability of the solenoid door lock.

5.2. Methodology for Software :

Collect a dataset: Gather a dataset of face images for training and testing purposes. The dataset should contain a diverse range of face images with different angles, lighting conditions, and expressions.

Prepare the dataset: Convert the images to grayscale and crop them to include only the face region. This will help to reduce the amount of noise in the images and improve the accuracy of the face recognition algorithm.

Train the Haar cascades: Use OpenCV to train the Haar cascades to detect the face region in an image. This involves using a machine learning algorithm to learn the features that distinguish the face from the background. The resulting model can be used to detect faces in new images.

Extract LBP features: Use the LBP algorithm to extract features from the face region of the images. LBP is a texture descriptor that captures the local structure of an image. The LBP features are computed by comparing the intensity of each pixel with its neighbours.

Train an LBP histogram classifier: Use the extracted LBP features to train a classifier that can distinguish between different faces. This involves creating a histogram of the LBP features for each image and using these histograms to train a machine learning algorithm. The resulting model can be used to identify faces in new images.

Test the face recognition system: Use the trained model to recognize faces in new images. This involves detecting the face region using the Haar cascades and extracting the LBP features. The resulting features are then compared to the features of the known faces in the database using the LBP histogram classifier.

Evaluate the performance: Evaluate the performance of the face recognition system using metrics such as accuracy, precision, and recall. The performance can be improved by adjusting the parameters of the Haar cascades and LBP algorithm, and by increasing the size and diversity of the dataset.

Chapter 6: Software and Hardware Used

6.1.Python:

Among its many uses are web development, data analysis, artificial intelligence, machine learning, and scientific computing. Python is a high-level, interpreted programming language. Since its initial release in 1991, it has grown to rank among the most widely used programming languages worldwide.

One of the key features of Python is its simplicity and ease of use. It has a clean, readable syntax that is easy to learn and understand, making it an ideal language for beginners. At the same time, it is also a powerful language that can be used to build complex applications.

Python is an interpreted language, which means that it does not need to be compiled before it can be run. This makes it fast and efficient to develop and test code, as well as to modify and iterate on code quickly. Python also has a large and active community of developers who contribute to the language and its ecosystem of libraries and tools.

Some of the most popular libraries and frameworks in Python include NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow, and Django. These libraries provide a wide range of functionality for data analysis, visualization, machine learning, and web development, making Python a versatile language that can be used for many different applications.

Overall, Python is a powerful and versatile language that is widely used and supported by a large community of developers. Its simplicity and ease of use make it an ideal language for beginners, while its versatility and power make it a popular choice for professional developers and data scientists alike.

6.2.VNC Viewer:

VNC Viewer is a software application that allows you to remotely access and control another computer using the VNC (Virtual Network Computing) protocol. It is commonly used for remote technical support, remote administration, and remote working.

With VNC Viewer, you can connect to another computer over the internet or a local network and view and control its desktop, just as if you were physically sitting in front of the computer. This allows you to access files, run applications, and troubleshoot issues from a remote location, without having to physically be in the same location as the computer.

VNC Viewer works by sending and receiving screen updates over the network, allowing you to see and interact with the remote computer in real-time. It also supports file transfer and clipboard sharing, allowing you to transfer files and text between the local and remote computers.

VNC Viewer is compatible with a wide range of operating systems, including Windows, macOS, Linux, and Unix. It is also available as a mobile app for iOS and Android devices, allowing you to access and control remote computers from your smartphone or tablet.

Overall, VNC Viewer is a powerful and flexible tool for remote access and control of computers. Whether you need to provide technical support, remotely administer a system, or work from a remote location, VNC Viewer can help you stay connected and productive.

6.3.Libraries:

6.3.1. Open CV:

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library designed to help developers create applications that can process visual data such as images and videos. It was first released in 2000 and has since become one of the most widely used computer vision libraries in the world.

OpenCV provides a wide range of tools and functions for processing visual data, including image and video capture, image and video processing, object detection and tracking, machine learning, and deep learning. It is written in C++ and has interfaces for several programming languages, including Python, Java, and MATLAB. Some of the key features and capabilities of OpenCV include: Image and video capture and processing: OpenCV

provides tools for capturing and processing images and videos from cameras, files, and other sources.

Object detection and tracking: OpenCV includes several algorithms for detecting and tracking objects in images and videos, including Haar cascades, HOG (Histogram of Oriented Gradients), and Deep Neural Networks.

Machine learning and deep learning: OpenCV provides support for machine learning and deep learning algorithms, including SVM (Support Vector Machines), K-Nearest Neighbours, Random Forests, and deep neural networks like CNNs (Convolutional Neural Networks) and RNNs (Recurrent Neural Networks).

GUI and visualization: OpenCV includes tools for creating graphical user interfaces (GUIs) and visualizing results, such as drawing shapes and annotations on images and videos.

Overall, OpenCV is a powerful and flexible library for processing visual data and creating computer vision and machine learning applications. It is widely used in industries such as robotics, automotive, healthcare, and entertainment, as well as in research and education.

6.3.2. Numpy:

NumPy is a Python library for numerical computing that provides a powerful array processing capability. It is one of the fundamental libraries for scientific computing in Python and is widely used in a variety of fields, including data science, machine learning, engineering, and finance.

NumPy provides an array object that is similar to a list or a sequence in Python, but with added features for mathematical operations. These arrays are n-dimensional and can be indexed and sliced like regular lists, but they support vectorized operations, which can perform operations on entire arrays at once. This makes them more efficient and faster than regular Python lists, especially for large data sets.

NumPy also includes functions for linear algebra, Fourier transforms, and random number generation, among others. It also integrates well with other Python libraries, including

Matplotlib, which is used for data visualization, and Pandas, which is used for data manipulation.

Some of the key features and benefits of using NumPy include:

Efficient array processing: NumPy provides a fast and efficient way to perform numerical computations on large datasets.

Vectorized operations: NumPy supports vectorized operations, which can perform operations on entire arrays at once, making them more efficient and faster than regular Python lists.

Mathematical functions: NumPy includes a range of mathematical functions for linear algebra, Fourier transforms, and random number generation.

Integration with other Python libraries: NumPy integrates well with other Python libraries, including Matplotlib for data visualization and Pandas for data manipulation.

Overall, NumPy is a powerful and essential library for numerical computing in Python. Its efficient array processing capabilities and vectorized operations make it ideal for handling large datasets and performing complex mathematical computations.

6.3.3. SYS Module:

"SYS" is a built-in module in Python that provides access to some variables and functions that interact directly with the Python interpreter. It is a standard library module and comes installed with Python, so you don't need to install any additional packages to use it.

Some of the key features and capabilities of the "sys" module include:

System-related functions: The "sys" module provides functions that interact with the system on which the Python interpreter is running. For example, "sys.exit()" can be used to exit a Python script, and "sys.argv" provides access to the command-line arguments passed to the script.

Runtime-related variables: The "sys" module provides access to some variables that are related to the runtime environment of the Python interpreter. For example, "sys.version" provides the version of Python currently running, and "sys.path" provides the list of directories that Python will search for modules.

Memory management: The "sys" module provides functions that can be used to manage memory in Python. For example, "sys.getsizeof()" can be used to determine the size of an object in memory, and "sys.getrefcount()" can be used to determine the number of references to an object.

Standard input and output: The "sys" module provides access to the standard input, output, and error streams of the Python interpreter. For example, "sys.stdin" can be used to read input from the user, and "sys.stdout" can be used to print output to the console.

Overall, the "sys" module is a powerful tool that can be used to interact with the Python interpreter and manage various aspects of the Python runtime environment. It is an essential module for many advanced Python scripts and applications.

6.3.4. Time Module:

The "time" module in Python provides functions that deal with various aspects of time. It is a built-in module and comes installed with Python, so you don't need to install any additional packages to use it.

Some of the key features and capabilities of the "time" module include:

Time access and conversions: The "time" module provides functions to access and manipulate time values. For example, "time.time()" returns the current time in seconds since the Epoch (January 1, 1970), and "time.gmtime()" can be used to convert a time value to a struct_time object representing a UTC time. Time formatting: The "time" module provides functions to format time values into strings. For example, "time.strftime()" can be used to format a time value into a string using a specified format string.

Sleep function: The "time" module provides a "time.sleep()" function that can be used to suspend the execution of a script for a specified number of seconds.

Performance measurement: The "time" module provides functions to measure the performance of a script. For example, "time.perf_counter()" can be used to measure the time taken by a block of code, and "time.process_time()" can be used to measure the CPU time used by a process.

Overall, the "time" module is a powerful tool that can be used to deal with various aspects of time in Python. It is especially useful for tasks that involve time-based calculations or performance measurement.

6.3.5. PIL:

The Python Imaging Library (PIL) is a popular open-source library for image processing and manipulation in Python. However, it is no longer actively maintained and has been replaced by the Pillow library, which is a fork of PIL.

Pillow is an enhanced version of PIL that adds new features and functionality, while also maintaining backward compatibility with PIL.

Some of the key features and capabilities of the Pillow library include:

Image manipulation: Pillow provides functions for reading, writing, and manipulating a wide range of image file formats, including JPEG, PNG, GIF, BMP, TIFF, and more.

Image filtering: Pillow provides a range of image filtering functions, including blur, sharpen, edge enhancement, and more.

Image enhancement: Pillow provides functions for enhancing images, including adjusting brightness, contrast, and color balance.

Image transformations: Pillow provides functions for transforming images, including rotation, scaling, and cropping.

Drawing: Pillow provides functions for drawing on images, including drawing lines, rectangles, circles, and text.

Image analysis: Pillow provides functions for analyzing images, including calculating image histograms and extracting color information.

Overall, Pillow is a powerful and versatile library for image processing and manipulation in Python. It is widely used in a variety of applications, including web development, computer vision, and scientific research.

6.3.6. Pickle Module:

The "pickle" module in Python provides a way to serialize and deserialize Python objects. Serialization is the process of converting an object into a stream of bytes that can be saved to a file or sent over a network. Deserialization is the reverse process of creating an object from a stream of bytes.

Some of the key features and capabilities of the "pickle" module include:

Serialization: The "pickle" module provides functions for serializing Python objects into a binary format that can be stored in a file or sent over a network. The "pickle.dump()" function can be used to serialize an object and write it to a file, while the "pickle.dumps()" function can be used to serialize an object and return a string.

Deserialization: The "pickle" module provides functions for deserializing a binary string or file back into a Python object. The "pickle.load()" function can be used to read a serialized object from a file and create a Python object, while the "pickle.loads()" function can be used to create a Python object from a serialized binary string.

Compatibility: The "pickle" module is compatible with a wide range of Python objects, including custom classes and complex data structures like lists, dictionaries, and tuples.

Security: The "pickle" module has some security concerns, as it can execute arbitrary code during the deserialization process. Therefore, it is recommended to only unpickle data from trusted sources.

Overall, the "pickle" module is a powerful tool that can be used to save and load Python objects in a convenient and efficient way. However, it is important to use it carefully and with caution to avoid any security issues.

6.3.7. Telepot:

Telepot is a Python library that provides a simple and easy-to-use interface for building Telegram bots. Telegram is a popular messaging platform that allows users to send and receive messages, photos, videos, and other types of media.

With Telepot, you can create bots that can receive and respond to messages from users, as well as perform a wide range of tasks, including sending messages, photos, videos, and files.

Some of the key features and capabilities of Telepot include:

Bot creation: Telepot makes it easy to create a new bot on Telegram and obtain an API token that can be used to authenticate with the Telegram API.

Message handling: Telepot provides functions for handling incoming messages from users, including text messages, photos, videos, and other types of media.

Message sending: Telepot provides functions for sending messages, photos, videos, and files to users.

Chat management: Telepot provides functions for managing chats with users, including getting information about a user or group chat and adding or removing users from a group chat.

Inline mode: Telepot supports inline mode, which allows users to interact with a bot in a chat without having to start a new conversation.

6.4. Hardware:

Block Diagram

Face Recognition-door Lock

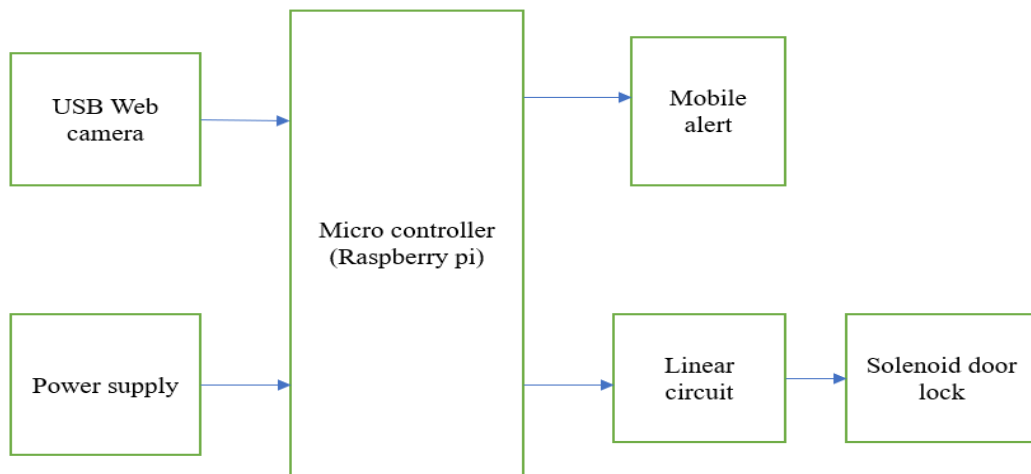


Figure 11: Block diagram of hardware

6.4.1. Raspberry pi 4 :

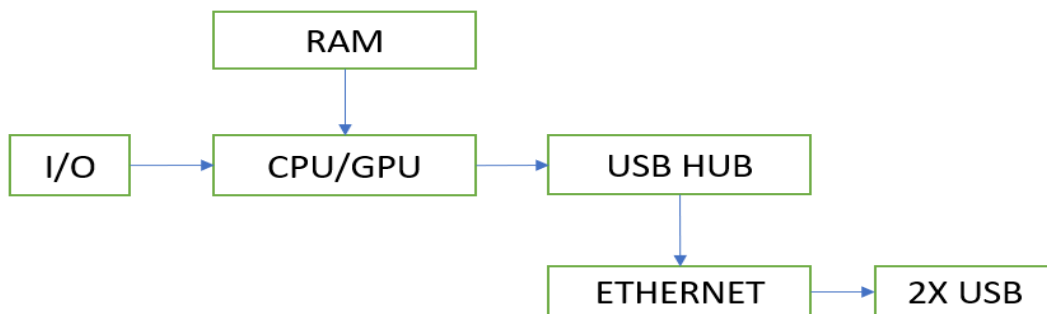


Figure 12: Block diagram of raspberry pi

Raspberry Pi is a series of small, single-board computers (SBCs) that are popular among hobbyists, educators, and developers for a wide range of projects. It was initially developed by the Raspberry Pi Foundation, a UK-based charity organization, with the goal of providing an affordable and accessible platform for learning about computer programming and electronics.

The Raspberry Pi board is typically the size of a credit card and comes with various models and versions, each with its own specifications and features. The most common models are the Raspberry Pi 4, Raspberry Pi 3, and Raspberry Pi Zero, with the Raspberry Pi 4 being the latest and most powerful model.

Here are some key features of the Raspberry Pi:

Hardware:

CPU: Raspberry Pi boards use ARM-based processors, which are energy-efficient and commonly used in mobile devices.

RAM: Raspberry Pi boards typically come with varying amounts of RAM, ranging from 1 GB to 8 GB, depending on the model.

Storage: Raspberry Pi boards usually have a microSD card slot for storage, but they can also connect to external storage devices such as USB drives or network-attached storage (NAS).

Operating System: Raspberry Pi boards can run a variety of operating systems, including the official Raspberry Pi OS (formerly known as Raspbian), which is a Linux-based operating system optimized for the Raspberry Pi.

Programming: Raspberry Pi supports a wide range of programming languages, including Python, C/C++, Java, and many more. This makes it an excellent platform for learning and developing software.



Figure 13: Raspberry Pi 4B model

6.4.2. USB Camera:



Figure 14: USB Camera

A USB web camera, also known as a USB webcam, is a type of digital camera that can be connected to a computer or other device via a USB (Universal Serial Bus) port. It is designed

specifically for capturing video and/or still images for use in video conferencing, online streaming, video recording, or other applications that require a camera to be connected to a computer.

6.4.3. Power supply:

power supplies for Raspberry Pi:

Power Requirements: The Raspberry Pi has specific power requirements, typically 5 volts (V) DC and a recommended current capacity of at least 2.5 amps (A) for most models. However, the power requirements may vary slightly depending on the specific model of Raspberry Pi and any peripherals connected to it, such as USB devices, displays, or cameras.

Connector Type: The Raspberry Pi typically uses a micro-USB or USB-C connector for its power input, depending on the model. The power supply should have the appropriate connector to match the Raspberry Pi

power supplies for solenoid door locks:

Voltage Requirements: Solenoid door locks typically require a specific voltage to operate, which is usually specified by the manufacturer. Common voltage ratings for solenoid door locks can range from 12 volts (V) DC to 24V DC, although other voltages may be used depending on the specific model or application.

Current Requirements: Solenoid door locks also require a specific current or amperage to operate. The current requirements are typically specified by the manufacturer and may vary depending on the size and type of solenoid used. It's important to choose a power supply that can provide the required current to ensure proper operation of the solenoid door lock.

6.4.4. Solenoid lock:



Figure 15: Solenoid Door Lock

A solenoid door lock is an electromechanical device that is used to control access to a door by locking or unlocking it using a solenoid, which is an electrically operated coil of wire. Solenoid door locks are commonly used in a variety of applications, including residential, commercial, and industrial settings, and can provide security and access control for doors.

Here are some key points to know about solenoid door locks:

Operation: A solenoid door lock operates by using an electrical current to energize the solenoid coil, which generates a magnetic field that moves a plunger or bolt, either to engage or disengage the lock mechanism. When the solenoid is energized, the lock is typically unlocked, allowing the door to be opened, and when the solenoid is de-energized, the lock is typically locked, securing the door.

Applications: Solenoid door locks are used in various applications, such as residential and commercial doors, cabinets, gates, safes, and access control systems. They are commonly used in combination with other security measures, such as keypads, card readers, or biometric scanners, to provide secure access control to authorized personnel.

In summary, solenoid door locks are electromechanical devices that are used to control access to doors by locking or unlocking them using a solenoid. They have specific voltage and current requirements, and proper installation, wiring, and quality are crucial for their safe and reliable operation. Solenoid door locks are commonly used in various applications to provide access control and security.

linear components used in door lock are

1.Resistors



Figure 16: Resistor

A resistor is a passive electronic component that restricts the flow of electric current in a circuit. It is one of the most basic and commonly used components in electronic circuits, and it is used to control the amount of current flowing through a circuit, regulate voltage, limit current, provide stability, and dissipate heat

2.Tip Transistor :



Figure 17: TIP Transistor

Transistors are three-terminal electronic devices that are used as amplifiers or switches in electronic circuits. They are fundamental building blocks of modern electronic circuits and are widely used in a wide range of applications, including computers, telecommunications, consumer electronics, and many other electronic devices.

Transistors come in various types, including bipolar junction transistors (BJTs) and field-effect transistors (FETs), each with their own characteristics and usage.

Hardware:

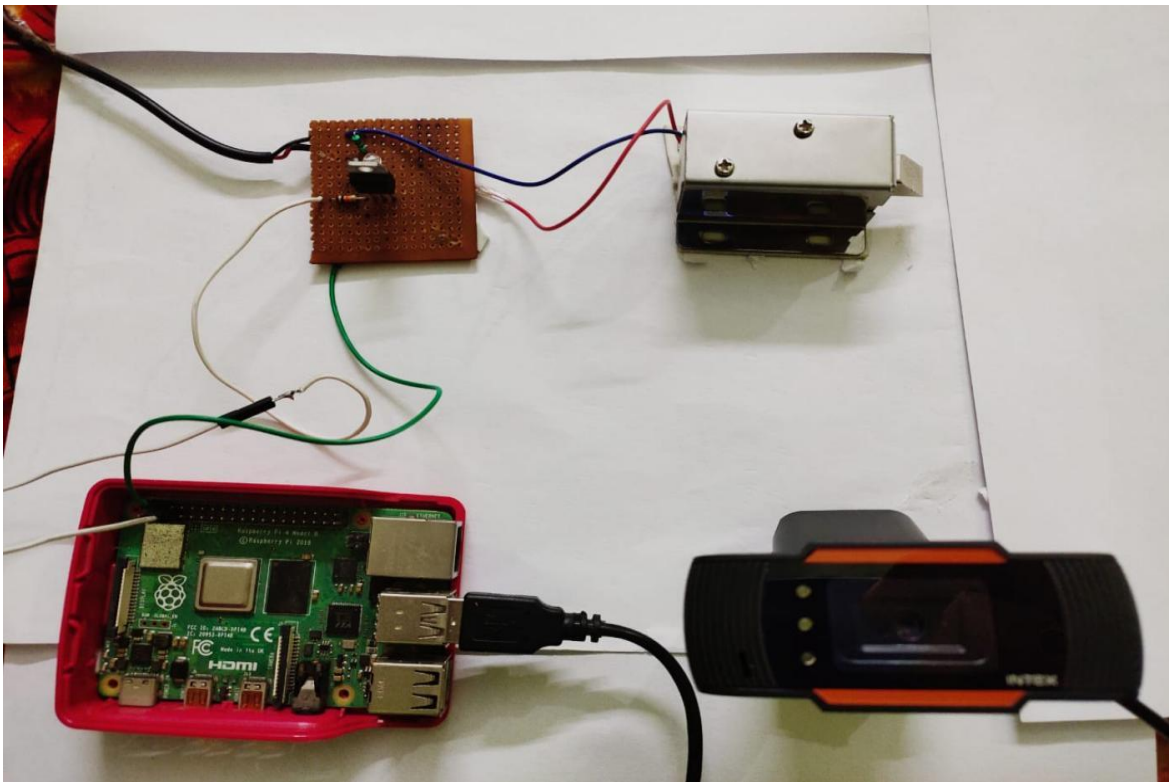
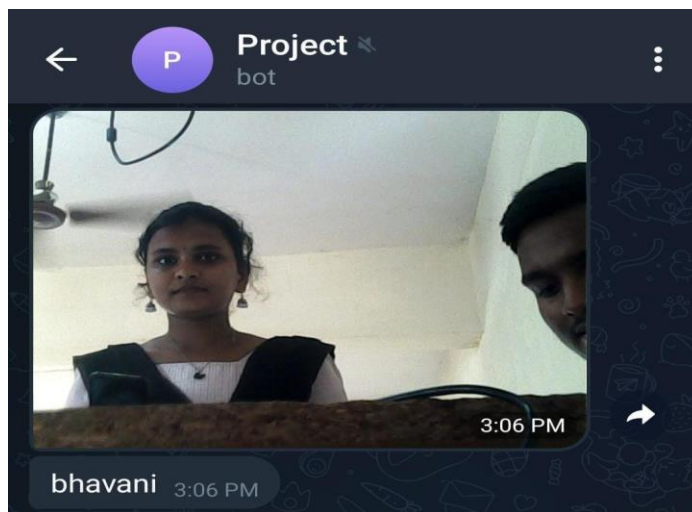


Figure 18: Hardware circuit Diagram

The above circuit and components are required for hardware implementation of smart door lock .

Results

The following are the results obtained.



Person identified as known



Front view of door open



Back view of door open

Figure 19: Result of known person

The above image indicates the recognized face which is sent through telegram to the owner's mobile phone. The door gets opened in this case.

In the other case

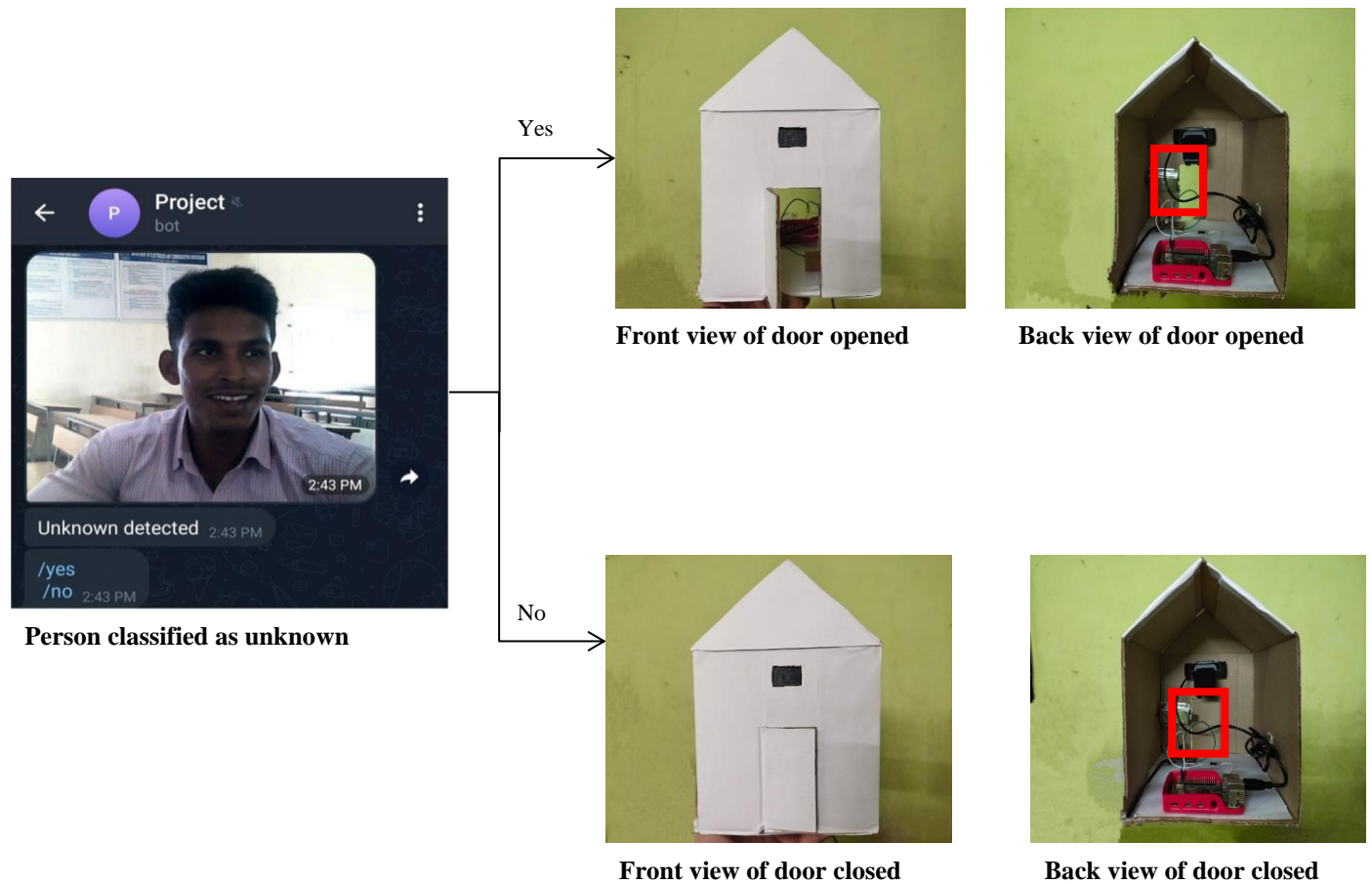


Figure 20: Result of unknown person

The above result indicates that the person is not present in the database and the owner is notified with the unknown person's photo. The owner can remotely open or close the door in this case, using the commands /yes and /no.

Conclusion

In conclusion, The proposed facial recognition door lock system is built, in which an Haar classifier is used for face detection, local binary pattern histogram algorithm for facial recognition, camera to capture the images and raspberry pi for processing these operations. The system allows authorized persons to enter and If the person is not authorized, a photo is sent to the owner's phone so that owner can remotely open or close the door. The system provides a convenient and secure way to control access to buildings and homes, especially in situations where keys or access cards can be easily misplaced or stolen.

The use of Raspberry Pi for this application makes the system cost-effective, easy to install, and maintain. The combination of Haar Cascade and Local Binary Pattern Histogram algorithms ensures accurate and reliable face recognition.

Overall, the face recognition door lock system using Haar Cascade and Local Binary Pattern Histogram on Raspberry Pi has great potential to revolutionize access control systems and security measures in the future.

Future Scope

The accuracy can be further improved by increasing the size of the data set.

In terms of future scope, multi-factor authentication and Integration with smart home systems can be further done.

REFERENCES

1. Yong.T.P, Pranesh.S,Jae-Young.P(2009), Smart digital door lock for the home automation, IEEE, 978-1-4244-4547- 9/09/\$26.00 ©2009 IEEE
2. Ilkyu Ha (2015), Security and usability improvement on a digital door lock system based on Internet of things, International Journal of Security and Its Applications Vol.9, No.8 (2015), pp.45-54
3. Ibshar.I, Wajiha.M.Ali, Sana.G, Sadia.S , Maria.W, Fakhra. A (2017), smart door lock system with automation and security, issn: 1013-5316; coden: sinte 8
4. Swati.P, V. D. Ugale², Vishal. P, Aditya.P, Nikhil.P(2020), Review Paper Based on Smart Locking System For Illegally Parked Vehicle,IJRAR March 2020, Volume 7, Issue 1 ISSN 2348-1269, P- ISSN 2349-5138)
5. Joongjin Kook (2019), Design and Implementation of an OTP-based IoT Digital Door-lock System and Applications, International Journal of Engineering Research and Technology, ISSN 0974-3154, Volume 12, Number 11 (2019), (pp. 1841- 1846)
6. JayantDabhade, AmirushJavare, TusharGhayal, AnkurShelar, “Smart Door Lock System: Improving Home Security using Bluetooth Technology” , International Journal of Computer Applications (0975 – 8887) Volume 160 –No 8, February 2017
7. Ushie James Ogri, DonatusEnangBasseyOkwong, AkaisoEtim, “Design and construction of door locking security system using GSM”, International Journal Of Engineering And Computer ScienceISSN:2319-7242 Volume 2 Issue 7 (July 2013), Page No. 2235-2257
8. Md. Nasimuzzaman Chowdhury, Md. ShibleeNooman, SrijonSarker, “Access Control of Door and Home Security by Raspberry Pi through Internet”, International Journal of Scientific & Engineering Research, Volume 4, Issue11, November-2013 ISSN 2229-5518.

9. M. Sajjad et al., "Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities," *Futur. Gener. Comput. Syst.*, 2017.
10. Patil, A. N., Ranavare, R. B., Ballal, D. V., & Kotekar, A. Raspberry Pi based face recognition system for door unlocking. *International journal of innovative research in science and engineering vol*, (2)
11. Saputra, R., & Surantha, N. (2021). Smart and real-time door lock system for an elderly user based on face recognition. *Bulletin of Electrical Engineering and Informatics*, 10(3), 1345-1355.
12. Yedulapuram, S., Arabelli, R., Mahender, K., & Sidhardha, C. (2020, December). Automatic Door Lock System by Face Recognition. In *IOP Conference Series: Materials Science and Engineering* (Vol. 981, No. 3, p. 032036). IOP Publishing
13. Sourav Roy; Md Nasir Uddin; Md Zahirul Haque; Md Jahidul Kabir, "Design and Implementation of the Smart Door Lock System with Face Recognition Method Using the Linux Platform Raspberry Pi", by *IJCSN - International Journal of Computer Science and Network*, 7(6), December 2018
14. I. Aydin and N. A. Othman, "A new IoT combined face detection of people by using computer Vision for security application," *International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, 2017, pp. 1-6, doi: 10.1109/IDAP.2017.8090171.
15. Sahani, M., Nanda, C., Sahu, A., & Pattnaik, B. (2015). Web-based online embedded door access control and home security system based on face recognition. 2015 *International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, 1-6.