CRIMINAL IDENTIFICATION AND ALERTING SYSTEM USING FACE DETECTION

TEAM MEMBERS:
P.SRINIVAS – 16BEC1016
M.MANIDEEP – 16BEC1032
K.HEMANTH KUMAR – 16BEC1297

GUIDED BY : Dr . CHITRA K

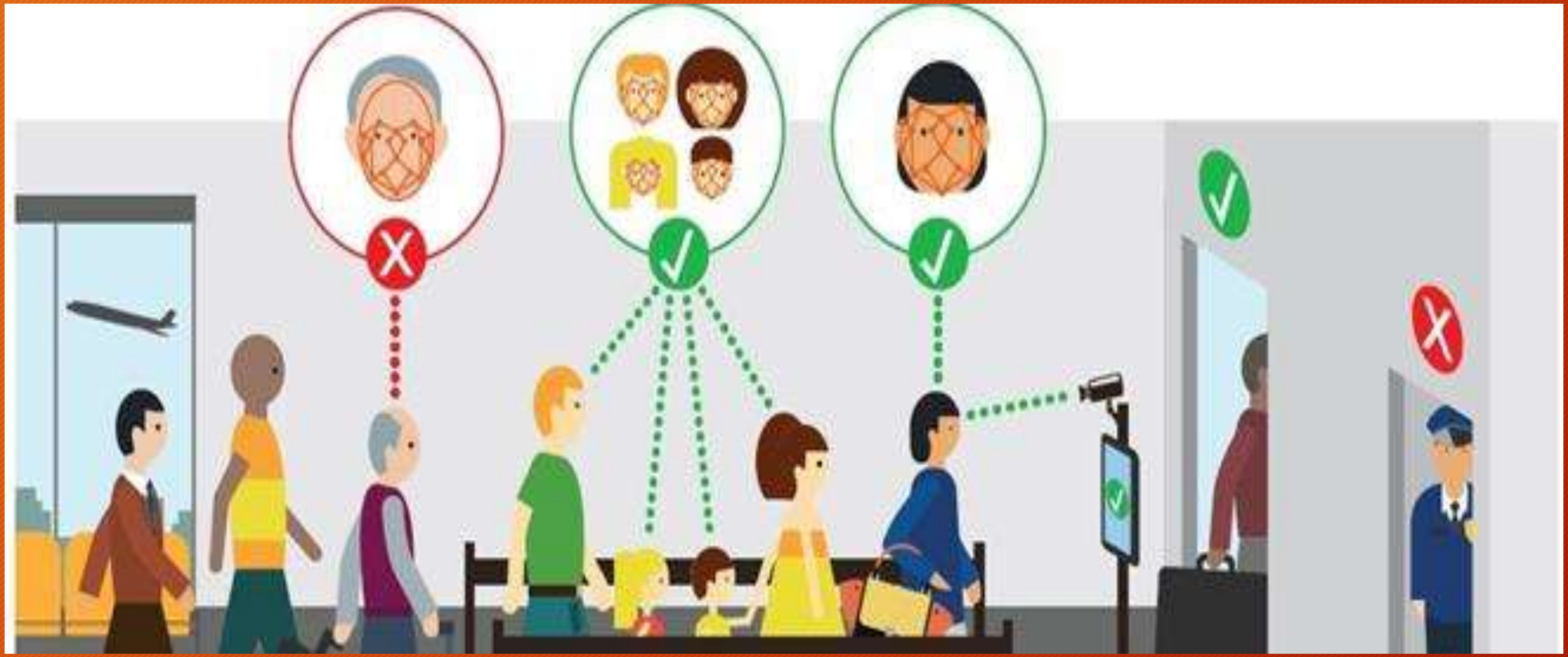
INTRODUCTION

- In this project, an automated facial recognition system for criminal database is proposed.
- We have used machine learning technique to implement this system.
- This system will be able to detect face and recognize face automatically in real time. Its database containing all information about criminals makes it easier to catch the criminals.
- If the sensor identifies the criminal, it alerts the authorities so that we can prevent a crime before it takes place.
- Such systems will be very effective for Defense Services, Bank Services, Public places, Airport service.

MOTIVATION

- Introduction of technologies have made significant changes in many daily activities
- Machine learning has a lot of importance and applications because they are able to independently adapt to the new exposed data. One of the major applications of machine learning is Face detection system.
- Our idea is to make use of this face detection system for social causes and decrease crime rate.
- In this project, we have proposed an effective criminal detection system using face detection.
- The authorities can breathe a sigh of relief with the face recognition system.

Face recognition at public places



Literature Survey

Title	Author	Algorithms used	Date of publish	Journal
Face Detection and Recognition using Raspberry Pi	Ishita Gupta, Varsha Patil, Chaitali Kadam, Shreya Dumbre	• Haar Cascade • PCA(Principle Component Analysis) • Eigen Face Approach	19-21 December 2016, AISSMS, Pune, India	2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)
Face recognition based door unlocking system using Raspberry Pi	*Thulluri Krishna Vamsi, Kanchana Charan Sai, Vijayalakshmi M*	• Eigen Faces • Fisher Face • LBPH(Local Binary Pattern Histogram)	2019	International Journal of Advance Research, Ideas and Innovations in Technology

Title	Author	Algorithms used	Date of Publish	Journal201
Deep Neural Network for Human Face Recognition	Dr. Priya Guptaa, Nidhi Saxenaa, Meetika Sharmaa, Jagriti Tripathia	• Haar Cascade • Deep neural Networks • Convolutional neural networks	08 January 2018	*Research Association of Modern Education and Computer Science*
Real-Time Implementation Of Face Recognition System	Neel Ramakant Borkar, Sonia Kuwelkar	• Principle Componet Analysys • Linear Descriminant Analysys • EigenVectors • EigenValues	2017	International Conference on Computing Methodologies and Communication

Title	Author	Algorithms used	Date of publish	Journal
Facial Recognition using OpenCV	Shervin EMAMI, Valentin Petrut SUCIU	• Eigenface • Haar Cascade	2012	Journal of Mobile, Embedded and Distributed Systems
Face Detection and Recognition Using OpenCV	Mrs. Madhuram.M, B. Prithvi Kumar, Lakshman Sridhar, Nishanth Prem, Venkatesh Prasad	• AdaBoost • Eigenface using grey scale images • Haar Cascade Classifier	10 Oct 2018	International Research Journal of Engineering and Technology (IRJET)

Title	Author	Algorithms used	Date of publish	Journal
Implementation of Face Recognition Algorithm for Biometrics Based Time Attendance System	Adrian Rhesa Septian Siswanto, Anto Satriyo Nugroho, Maulahikmah Galinium	Eigen face method	April 2016	Center for Information & Communication Technology Agency for the Assessment & Application of Technology
Face Detection for Security Surveillance System	Zhang Jian, Song Wan-juan	Haar cascade algorithm, Ada boost algorithm	December 2018	The 5th International Conference on Computer Science & Education Hefei, China

Title	Author	Algorithms used	Date of publish	Journal
Face recognition Issues and Alternative applications	Waldemar Wójcik, Konrad Gromaszek and Muhtar Junisbekov	PCA, LDA,LRC and Neural Networks.	September 2018	International of advanced research in computer and communication engineering Vol. 7, Issue 14,
Face Recognition Using LBPH Descriptor and Convolution Neural Network	V.Betcy Thanga Shoba1, Dr. I.Shatheesh Sam	LBPH and Nuural Networks	October 2018	International Conference on Intelligent Computing and Control Systems. 2018

- In [1],authors used Haar Cascade algorithm to detect faces and Eigen face Approach for face recognition.
- In [2], journal authors attempted to create a face recognition based door locking system, They used three different methods, Eigen faces, Fisher faces and LBPH Algorithms.
- In [3], authors implemented the face recognition system using deep neural networks and obtained the results.
- In [4], Authors have used Eigen faces and Fisher face method to recognize the faces.

- In [5], authors implemented the face detection using Haar cascade algorithm and face recognition system using Eigen face method and obtained the results.
- In [6], Authors designed a face detection system using OpenCV and has used Ada Boost and Eigen face methods to obtain the results.
- In [7],authors implemented the face recognition algorithm along with the biometric based attendance system in order to increase the security
- In [8],Authors designed a face detection of security surveillance system they have used Haar Cascade algorithm for face detection and AdaBoost algorithm for face recognition

- In [9], The authors have discussed about the various methods of face recognition, their efficiency and their disadvantages. The different methods include PCA, LDA,LRC and Neural Networks
- In [10], authors attempted to recognize faces using local binary pattern histogram approach and for detection convolution neural networks are used.

Hardware and Software Details

- Software
    - Python
    - OpenCV
    - Raspbian

- Hardware
    - Raspberry Pi
    - Mini camera video module

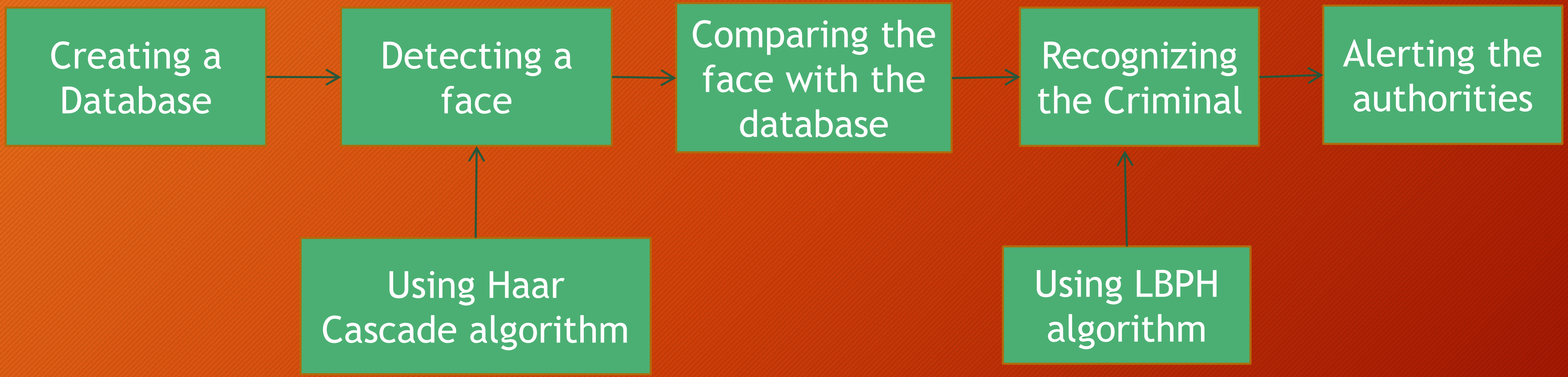
Face recognition for criminal identification.

- In this project, we propose face recognition technique for criminal identification.
- This system requires a camera module, Raspberry pi and a pc.
- Whenever a criminal gets detected, it alerts the authorities by an SMS, Gmail and a buzzer.
- This system can be installed in public areas like Airports, Railway stations, Malls, ATMs, etc.

HOW DOES IT WORK?

- To detect a particular person, we need the image of his face.
- These images can be extracted from CCTV cameras or through the finger prints of the criminal at the crime scene,etc.
- We need to store the images in our database and train the system.
- The more the amount of images, the better will be the efficiency of this system.
- Once the criminal gets detected it immediately sends the SMS to the authorities.

BLOCK DIAGRAM



Proposed methods for Face Recognition

- LBPH (Local Binary Pattern Histogram)

LBPH ALGORITHM

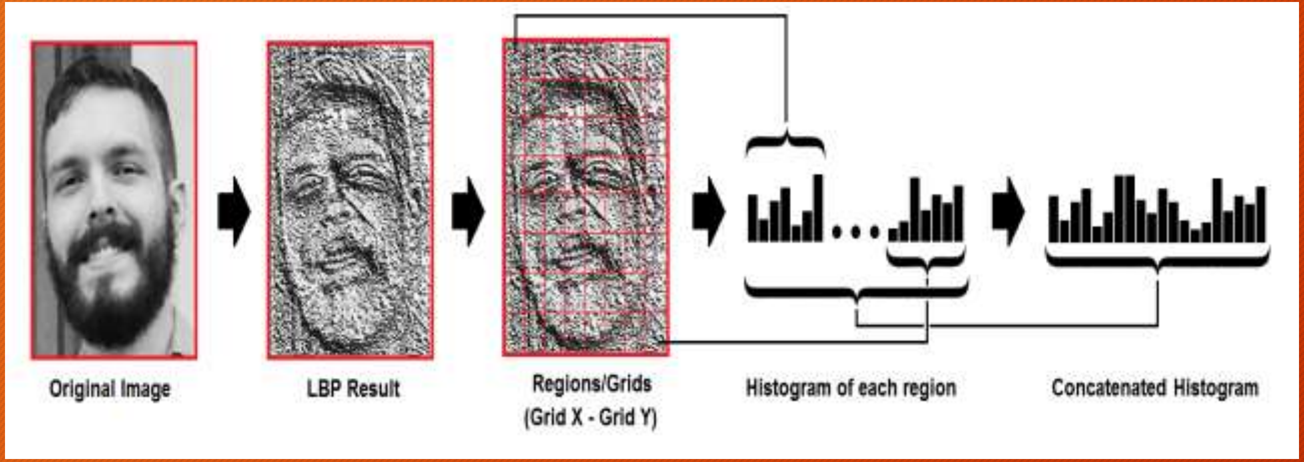
- **Local Binary Pattern** (LBP) is an efficient operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

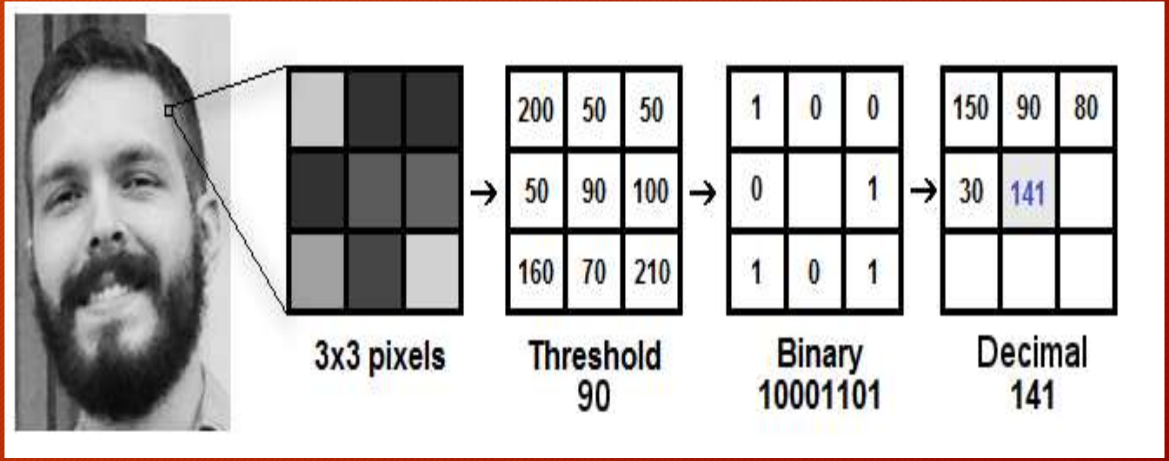
- HOW DOES IT WORK ?
- **Parameters**: the LBPH uses 4 parameters:
- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.

- **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

- **Training the Algorithm:** First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.
- **Applying the LBP operation:** The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

•**Extracting the Histograms:** Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image:

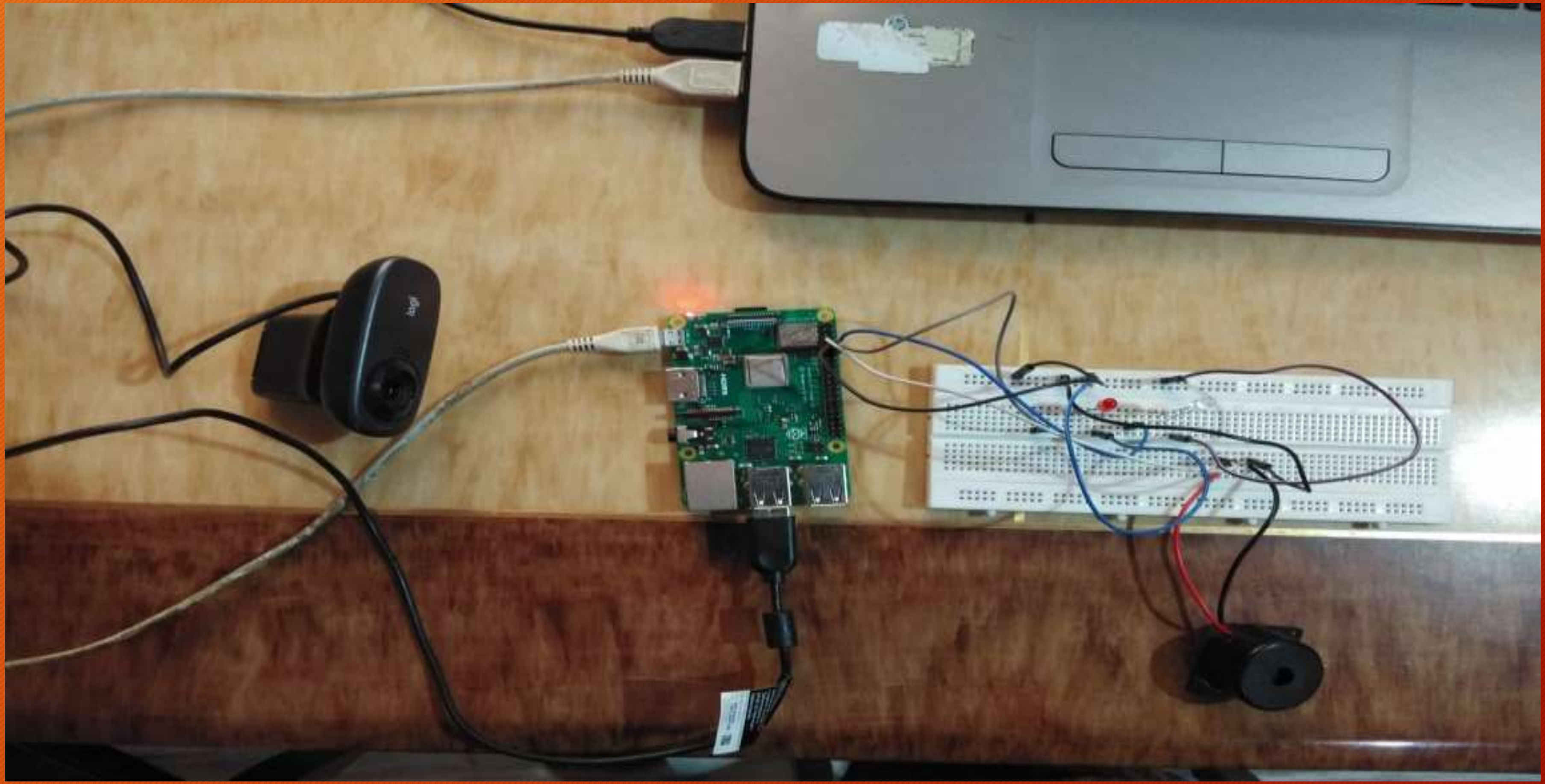




- **Performing the face recognition:** In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.
- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.

- **Conclusions**
- LBPH is one of the easiest face recognition algorithms.
- It can represent local features in the images.
- It is possible to get great results (mainly in a controlled environment).
- It is robust against monotonic gray scale transformations.
- LBPH ALGORITHM.docx

HARDWARE SETUP



Codes And Standards Used

- All the codes in this project are written in Python.
- We have used Haar cascade algorithm for Face detection and Local Binary Pattern Histogram for Face recognition.
- We have used Raspberry Pi to interface the webcam, other hardware components.
- We used a HD webcam, where it can detect any face over 55 degree and up to a 3m distance.

Comparision with the existing techniques

- Security is the important aspect of our lives.
- There are different methods of identifying the criminals and the techniques have been updating on daily basis.
- In most of the cases, we use CCTV cameras, Manual checks on roads or in public areas, etc. By using the face recognition, we can easily spot a person and it makes easier for the authorities.
- The algorithm we used is LBPH, unlike other algorithms it can detect the faces even when the intensity of the light is different. So, we preferred it for using in the public areas.

Constraints and Tradeoffs

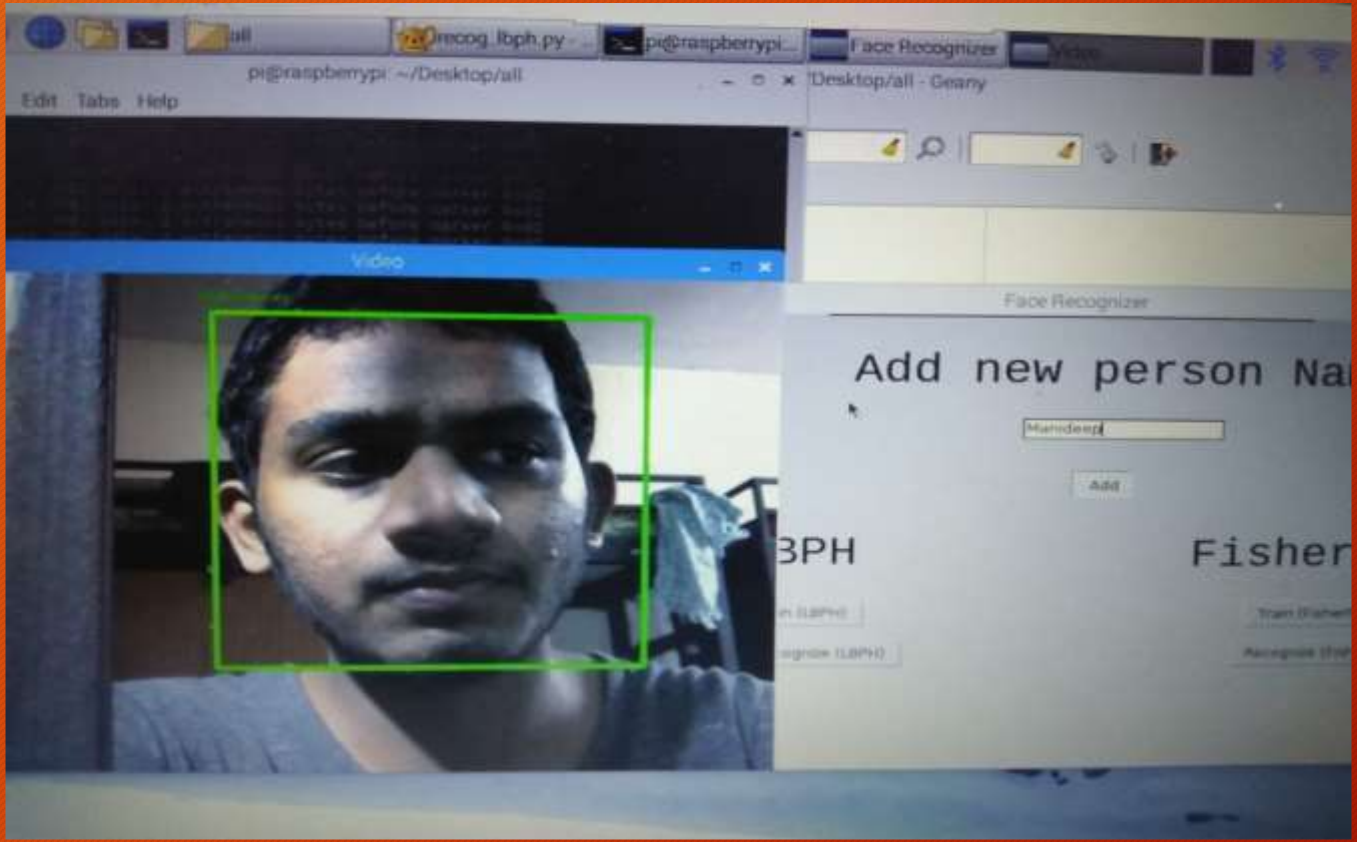
- As this project is based on face detection, The criminal might not get detected if he escapes the system by covering his face by any means.
- The camera we used can detect over 55 degrees and 3m distance. we can use a bigger camera to cover a larger distance.
- We can further improve this by using more high resolution camera so that the face is detected more accurately which can also be used for security surveillance.

Cost Analysis

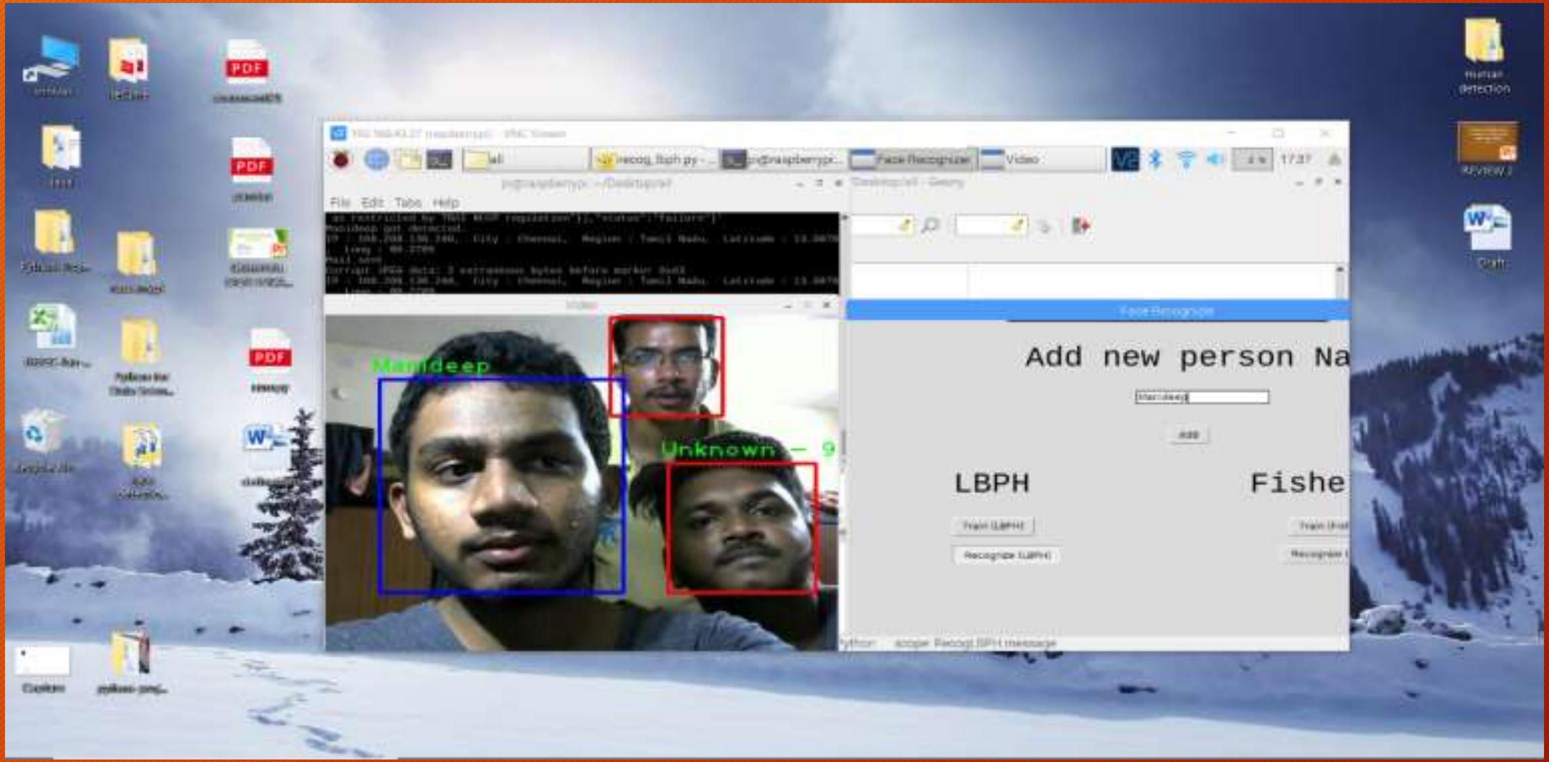
Components	Cost
Raspberry pi	3200
Web Camera	1700
Jump Wires	75
Bread board ,Buzzer And LED’s	200
Total	5175

Demonstration Of The System

- Adding Dataset:

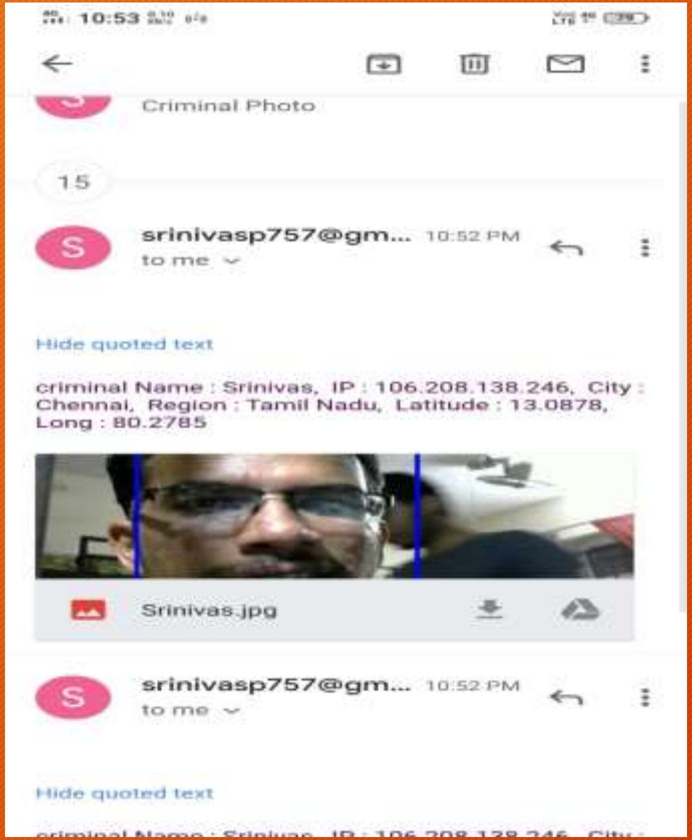


- Recognition of known and unknown faces:



```
as restricted by TRAI NCCP regulation"}],"status":"failure"}
Manideep got detected.
IP : 106.208.136.240,  City : Chennai,  Region : Tamil Nadu,  Latitude : 13.0878
, Long : 80.2785
Mail sent
Corrupt JPEG data: 3 extraneous bytes before marker 0xd3
IP : 106.208.136.240,  City : Chennai,  Region : Tamil Nadu,  Latitude : 13.0878
, Long : 80.2785
```

- Message and Mail Notifications:



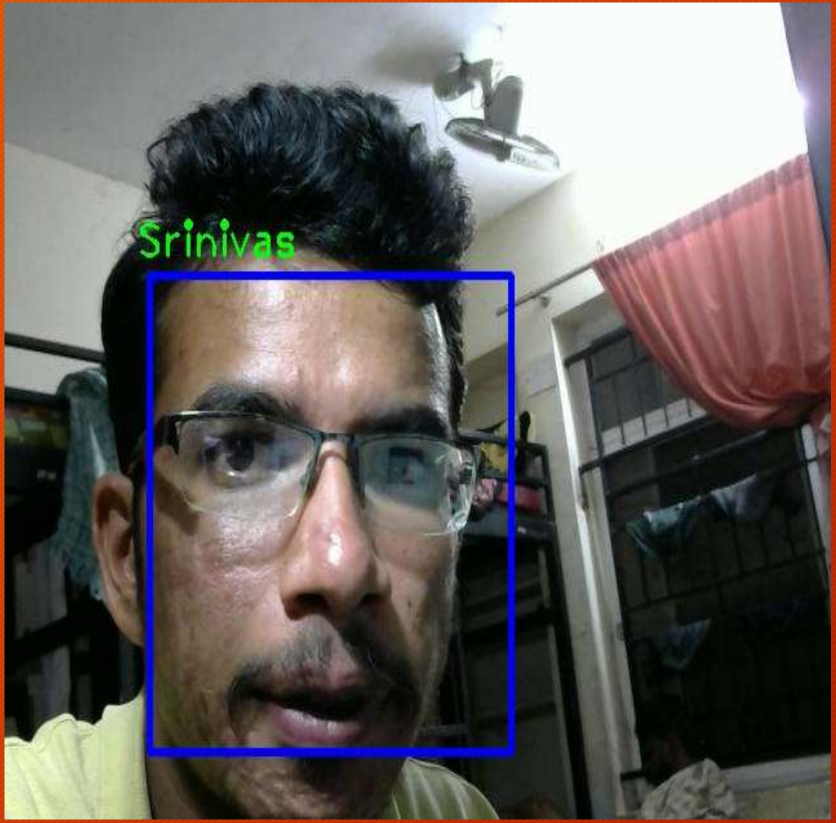
srinivasp757@gm... 10:52 PM
to me

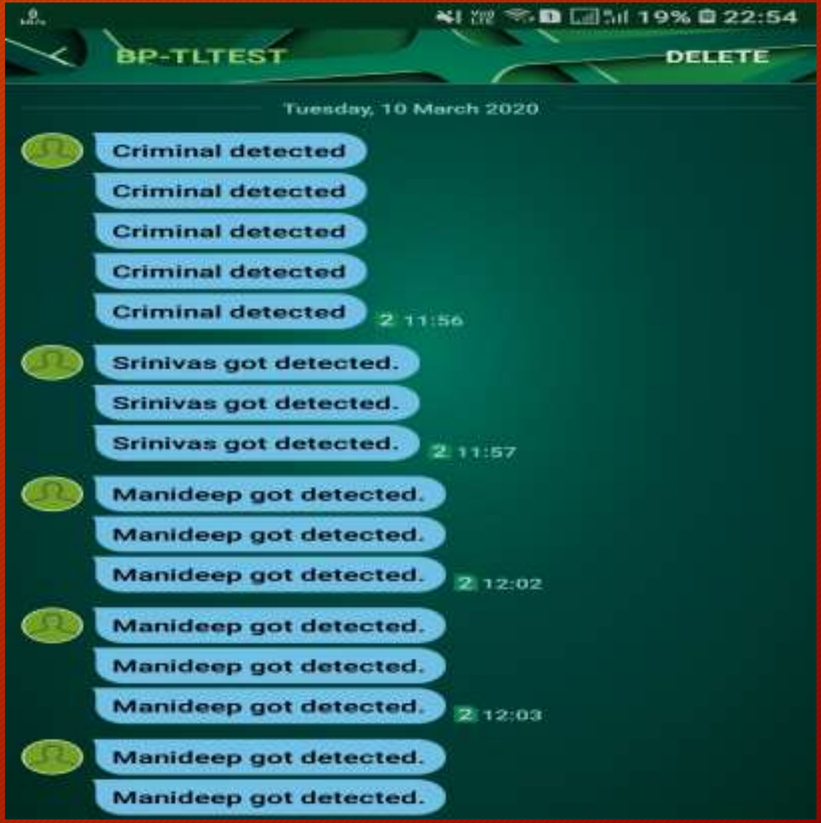
Hide quoted text

criminal Name : Srinivas, IP : 106.208.138.246, City : Chennai, Region : Tamil Nadu, Latitude : 13.0878, Long : 80.2785

srinivasp757@gm... 10:52 PM
to me

Hide quoted text





Tuesday, 10 March 2020

Criminal detected
Criminal detected
Criminal detected
Criminal detected
Criminal detected 2 11:56
Srinivas got detected.
Srinivas got detected.
Srinivas got detected. 2 11:57
Manideep got detected.
Manideep got detected.
Manideep got detected. 2 12:02
Manideep got detected.
Manideep got detected.
Manideep got detected. 2 12:03
Manideep got detected.
Manideep got detected.

Discussion Of The Results

- We are able to recognize the criminals successfully.
- We can successfully detect and recognize the faces in the range of 3m distance.
- The message which contains the name of the criminal is been sent to the authorities including the place and location where the criminal has been detected.

Applications

- In Airport security system
- Prevent Retail crime (can be installed in shopping malls to detect shop lifters)
- To protect stadiums, schools etc from threats.
- Validate Identity in ATMs.
- Control Access to sensitive areas.

References

- [1] Dr. Priya Guptaa, Nidhi Saxenaa, Meetika Sharmaa, Jagriti Tripathia.“Deep Neural Network for Human Face Recognition” Published Online January 2018 in MECS
- [2] Liton Chandra Paul , Abdulla Al Sumam.’ Face Recognition Using Principal Component Analysis Method. Volume 1, Issue 9, November 2012(IJARCET)
- [3]” Md Zahirul Haque; 4 Md Jahidul Kabir. Sourav Roy; 2 Md Nasir Uddin; “Design and Implementation of the Smart Door Lock System with Face Recognition Method using the Linux Platform Raspberry Pi by 3,Volume 7, Issue 6, December 2018(IJCSN)
- [4] Neel Ramakant Borkar, Sonia Kuwelkar” Real-Time Implementation Of Face Recognition System” . Published inICCMC 2017
- [5] Ishita Gupta, Varsha Patil, Chaitali Kadam, Shreya Dumbre. Face Detection and Recognition using Raspberry Pi” (WIECON-ECE), 19-21 December 2016, AISSMS, Pune, India
- [6] Thulluri Krishna Vamsi, Kanchana Charan Sai, Vijayalakshmi M”.Face recognition based door unlocking system using Raspberry Pi (Volume 5, Issue 2) IJARIIT 2019
- [7] Swathi .V, Steven Fernandes. Raspberry Pi Based Human Face Detection. Vol. 4, Issue 9, September 2015 in IJARCCE

CRIMINAL IDENTIFICATION AND ALERTING SYSTEM USING FACE RECOGNITION

A PROJECT REPORT

Submitted in partial fulfillment of the
requirement for the award of the
Degree of

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

by

SRINIVAS.P – 16BEC1016
M.MANIDEEP – 16BEC1032
K.HEMANTH KUMAR – 16BEC1297
Under the Guidance of

Dr.CHITRA.K

SCHOOL OF ELECTRONICS ENGINEERING
VELLORE INSTITUTE OF TECHNOLOGY
CHENNAI - 600127

May 2020

CERTIFICATE

This is to certify that the Project work titled "CRIMINAL IDENTIFICATION AND ALERTING SYSTEM USING FACE RECOGNITION" that is being submitted by Srinivas.P (16BEC1016), M.Manideep (16BEC1032), Hemanth Kumar Kadiyala(16BEC1297) is in partial fulfillment of the requirements for the award of Bachelor of Technology in Electronics and Communication Engineering, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

[illegible]

Dr.Chitra.K
Guide

The Project Report is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR
B. Tech. (ECE)

DEAN
School of Electronics Engineering

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, Dr.Chitra.K, Associate Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to Dr. Sivasubramanian A, Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Programme Chair Dr. Vetrivelan.P and Co-Chair Prof. Reena Monica P for their support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

P. Srinivas	[illegible]	[illegible]
SRINIVAS. P (16BEC1016)	**M.MANIDEEP (16BEC1032)**	**K. HEMANTH KUMAR (16BEC1297)**

ABSTRACT

Introduction of technologies have made significant changes in our daily activities. One of such is Machine learning. One of the significant applications of machine learning is Face detection system. In this project we have proposed an effective criminal identification and alerting system using face recognition it consists of Raspberry pi connected to a video camera module. This system can be installed in private and public places like airports, malls, ATMs etc. to restrict criminals to get inside.

The system consists of a criminal database, and if the pictures captured by the camera module whenever a face is detected, matches with the pictures in the database, the authorities are alerted. Our system implements the “Haar Cascade algorithm” for detecting face. “Local Binary Pattern Histogram” (LBPH) technique for identifying the faces. The programming for this system is done in Python.

LIST OF FIGURES

TABLES

CHAPTER 1
INTRODUCTION

From the past years, the people have used many different methods for catching the criminals and identifying them like CCTV cameras, finger prints etc. Now, we introduce the face recognition for catching the criminals face. It is a biometric technique which is very accurate and detects the criminal with a high precision. The technique uses the characteristics of the face of an individual to identify the person. The criminal's face can be identified using an image or a video automatically.

The comparison of selected facial features is done with the help of photos in a data base. The face recognition can be installed in a system to authenticate or identify a particular individual so, this can be installed in various areas for security purposes. The face recognition is installed in the biometrics system for verification, authentication, identification and authorization. Many companies use this technique in their CCTV cameras etc. For example, Facebook uses this technique in order to create a social media account for the users in it's website. In several countries government creates criminal data for the face recognition to identify the suspected criminals.

In India, Generally, the criminals are detected using the finger prints and CCTV cameras to know who has committed the crime. But, after knowing who is the culprit, catching him is also a tough task for the police. The easiest method of catching them is by getting the information about where they are present. For that we can use face recognition system. In this project we came up with a face identification system for a criminals in which the detection of a criminal is performed through matching the faces.

1.1 Objectives

The following are the objectives of this project:

- To design and implement an efficient criminal identification system to catch criminals.

- To implement the system with the raspberry pi and camera module to identify criminals and alert respective authorities.
- Once the captured faces match with the faces present in our database, we are alerted by an LED and a buzzer and the message that contains the name of the particular criminal is sent to the authorities.

1.2 Background and Related Work

In [1], authors used Haar Cascade algorithm to detect faces and Eigen face Approach for face recognition.

In [2], journal authors attempted to create a face recognition based door locking system, They used three different methods, Eigen faces, Fisher faces and LBPH Algorithms.

In [3], authors implemented the face recognition algorithm along with the biometric based attendance system in order to increase the security .

In [4], authors designed a face detection of security surveillance system they have used Haar Cascade algorithm for face detection and AdaBoost algorithm for face recognition

In [5], authors implemented the face recognition system using deep neural networks and obtained the results.

In [6], authors have used Eigen faces and Fisher face method to recognize the faces.

In [7], authors implemented the face detection using haar cascade algorithm and face recognition system using eigen face method and obtained the results.

In [8], authors designed a face detection system using OpenCV and has used AdaBoost and Eigenface methods to obtain the results.

In [9], the authors have discussed about the various methods of face recognition, their efficiency and their disadvantages. The different methods include PCA, LDA,LRC and Neural Networks

In [10], authors attempted to recognize faces using local binary pattern histogram approach and for detection convolution neural networks are used.

Table 1.1 is our literature survey. It shows different journals and the algorithms used by them for face recognition .

Table 1.1 Literature Survey

Sr. No	Name of article	Name of journal	Author
1.	Face Detection and recognition using Raspberry pi	2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)	Ishita Gupta, VarshaPatil, Chaitali Kadam, Shreya Dumbre
2.	Face recognition based door unlocking system using Raspberry pi	International journal of advance research, Ideas and innovations in technology	Thulluri KrishnaVamsi, Kanchana Charan Sai, Vijayalakshmi M
3.	Implementation of Face Recognition Algorithm for Biometrics Based Time Attendance System	Center for Information & Communication Technology Agency for the Assessment & Application of Technology	Adrian Rhesa Septian Siswanto, Anto Satriyo Nugroho, Maulahikmah Galinium
4.	Face Detection for Security Surveillance System	The 5th International Conference on Computer Science & Education Hefei, China	Zhang Jian, Song Wan-juan
5.	Deep Neural Network for Human Face Recognition	Research Association of Modern Education and Computer Science	Dr.Priya Guptaa, Nidhi Saxenaa, Meetika Sharmaa, Jagriti Tripathia
6.	Real-Time Implementation Of Face Recognition System	International Conference on Computing Methodologies and Communication	Neel Ramakant Borkar, Sonia Kuwelkar
7.	Facial Recognition using OpenCV	Journal of Mobile, Embedded and Distributed Systems	Shervin EMAMI, Valentin Petrut SUCIU

8	Face Detection and Recognition Using OpenCV	International Research Journal of Engineering and Technology (IRJET)	Mrs.Madhuram.M, B. Prithvi Kumar, Lakshman Sridhar, NishanthPrem, Venkatesh Prasad
9	Face recognition Issues and Alternative applications	International of advanced research in computer and communication engineering Vol. 7, Issue 14, September 2018	Waldemar Wójcik, Konrad Gromaszek and Muhtar Junisbekov
10	Face Recognition Using LBPH Descriptor and Convolution Neural Network	International Conference on Intelligent Computing and Control Systems. 2018	V.BetcyThangaShoba1, Dr. I.Shatheesh Sam

CHAPTER 2

FACE RECOGNITION SYSTEM

Here we discuss about the methodology, standards, design and constraints of face recognition system.

2.1 Methodology

Facial recognition is widely used for human identification because of its capability to measure and subsequently identify people. It is mainly used for security purposes. In this project, we propose an automated face recognition system for criminal database .The system detects the face and recognize it simultaneuously in real time. It’s database contains all information about criminals. Such systems will be very effective for defense services, sank services, public places, airport service.

We use Raspberry pi to control camera module to take the photos of faces. Open CV or Python library can detect one or more persons from the given images in the scene by the help of the database of positive images and negative images. The input images are compared with the images present in database. In our case if the face is matched, a message is sent to the respective authorities.

The Raspberry pi in the face recognition system makes the system less weight, smaller and also decrease the , so it is more convenient when compared to the PC-based face recognition system. Also the system sends a security alert message to the authorized person including the photo of the criminal. Fig 2.1 is a block diagram representing the steps of face recognition.

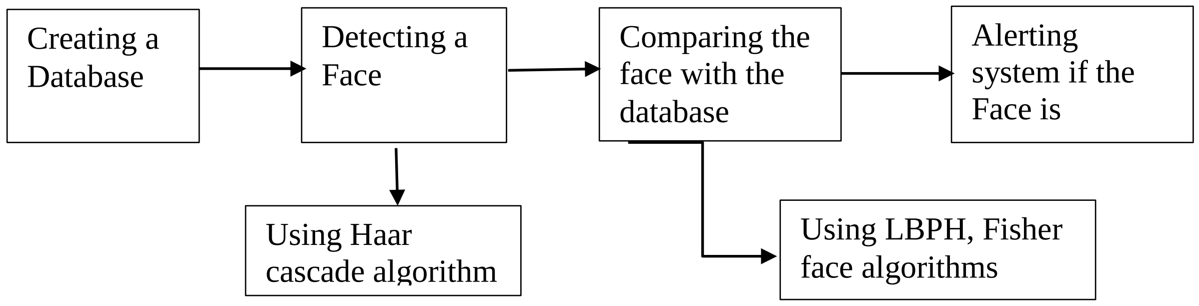

Creating a Database
Detecting a Face
Comparing the face with the database
Alerting system if the Face is
Using Haar cascade algorithm
Using LBPH, Fisher face algorithms

Fig 2.1 block diagram of face recognition process

Technology used:

- **Image processing**: This is a technique that is used for capturing the images and recognizing them by comparing the images with the images in database.
- **Embedded system design**: This technique is used in the module for which the hardware, software and many other functional operators are combined to perform the operation.
- **Wireless communication:** It is used for the communication through mobile network and Gmail .

2.1.1 BLOCK DIAGRAM

Fig 2.2 is the block diagram of the connection of hardware components.

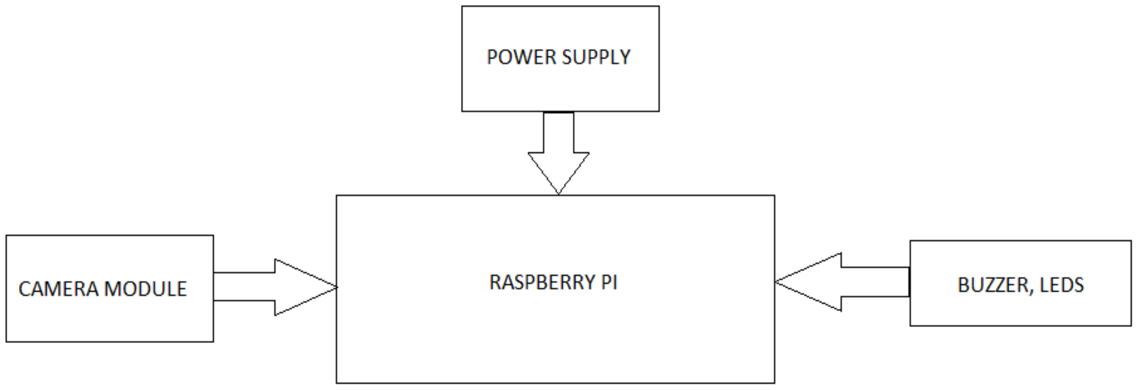


Fig 2.2 Block Diagram

2.2 Description of Tools:

In this section, the tools to implement and evaluate face detection and recognition using OpenCV are detailed.

2.2.1 Software Required

OpenCV 4.1.2, Python 3, VNC Viewer

PYTHON

It is a high-level programming language which can be used for all purposes. It is designed by “Guido van Rossum” and is initially released to the market in the year 1991. It’s design style emphasizes the readability of the code and it also uses the significant white space. It suggests the constructs which enables easy programming on small and large scales.

Python is an interactive, interpreted and object oriented script. It is designed to be highly readable. It is easier than other languages as other languages use punctuations where as python uses frequently used English words and also it has fewer syntactical constructions than the other languages such as C, C++, JAVA etc.

Python has an automatic memory management and dynamic type stream.

OPENCV

It is an open Source Computer Vision Library,which is a machine-learning software library primarily geared towards real time computer vision, developed by Intel. It focuses majorly on the real time image processing. OpenCV is built to produce a typical environment for the applications that work on computer vision and also to increase the employment of machine perception within the products. OpenCV, Being a BSD-licensed product makes it easier for many businesses to use and modify the code

To incorporate a comprehensive set of the classic and state of the art computer vision and machine learning techniques, it’s library contains quite 2500 optimised algorithms. Such algorithms will be used to detect and recognize the faces, identify the objects, classify the human actions in the videos, track moving objects, track camera movements, produce 3D point clouds from stereo cameras extract 3D models of objects , find similar images from the database, stitch images together to provide a high resolution image of a complete scene, remove red eyes from the images taken by flash, identify scenery and establish the markers to overlay it with the augmented reality, follow the eye movements etc

Open CV has interfaces of many programming languages like Python, Java, etc and also supports many operating systems. It’s library was first written using C-Programming and this interface helps it by making it accessible to the platforms such as digital signal processors. The wrappers for few programing languages are built to increase the adoption by a larger audience. However, since version 2.0, OpenCV had included both its traditional C interface also just like the new C++ interface. This interface scales back the amount of lines of code necessary to code up vision functionality and also to scale back common programming errors like memory leaks (by allocating and deallocating information) which is able to arise when using OpenCV in C. Mainly the developments and algorithms in OpenCV are now developed using the C++ interface.

Unfortunately, it's far more tough to provide wrappers in alternative languages to C++ code as hostile C code; therefore the different language wrappers are usually lacking variety of the newer OpenCV 2.0 features. It is used mainly in general vision applications and utilises the profits of “MMX” and “SSE” instructions. Fully featured CUDA and OpenCV interfaces are being actively developed straight away. approximately about 500 programs and almost 10 times as many functions which support these algorithms are present.

VNC VIEWER

It is a graphical desktop sharing software which makes use of the “Remote Frame Buffer protocol” (RFB) to access other pc. VNC viewer sends keyboard and the mouse applications from a pc to a different computer displaying the graphical user updates back within different directions within the network.

It is independent of the platform we are using in. It contains users, servers to several GUI operated OS and to Java programming language. Many users may interface to VNC servers in the identical time. Major applications of this technology includes the assistance and helping to access the files on any computer from personal pc or any other computer. Fig.2.3 shows the VNC Viewer.

Fig.2.3 VNC Viewer

2.2.2 Hardware Required

PC preferably running windows, Raspberry Pi 3, USB web camera, jump wires. The various connections required are as given below:

WEBCAM:

It is a camera which is connected to a computer. It can capture images or videos. By the use of software, It can send it’s captured video to the Internet in the real time. Nowadays, most of the laptops are embedded with a web camera. They can also be connected to the computer using a USB or a fire wire port. The webcam does not contain a built in storage like digital camera. It uses the computer’s hard drive as it can be connected to pc. The below fig 2.4 represents the camera used for this system.

Fig 2.4 Webcam

RASPBERRY PI:

It is the old standard for affordable uniboard computing, it has been powering everything including from robots to the latest smart home devices. The Raspberry Pi 3 Model B is that the most recent version of the $35 Raspberry Pi computer. The Pi is not the same as your common machine, in its cheapest form it does not have a case, and it is simply an electronic board in credit card size of the kind you'll find inside a PC or laptop but much smaller. One thing connected in mind is that the Pi by itself is just a bare board. You will also need a monitor or TV, an influence supply, leads to attach with the monitor typically HDMI, and a mouse and a keyboard.

Once you've plugged altogether the cables, the only way for brand clean users to induce and running on the Raspberry Pi is to download the New Out Of Box Software installer (NOOBS). Once if the download is finished, we have to follow the instructions present and it will walk us through install the OS on the Pi. The installer makes it easier to line up different operating systems, even though an honest choice for the first time users is that the official OS Raspbian although other operating systems. The look and characteristics of Raspbian should be familiar to any pc user. The OS, which is continually being improved, recently had a graphical overhaul, and includes an optimized application, an office suite, programming tools, educational games, and other software. Fig 2.5 represents Rasberry pi.

Fig 2.5 Raspberry pi

2.3 Summary

Therfore, the methods, standards implemented, description of the hardware and software components that are used in this project are discussed in this chapter. The next chapter consists of the process, approach and implementation of criminal identification using the face recognition. It also describes the methodology of the face recognition.

CHAPTER 3
SYSTEM IMPLEMENTION

In this chapter we discuss how criminal identification system is implemented. This chapter explains the Haar cascade algorithm used for detecting the face and local binary pattern histogram algorithm for recognizing the face and also the communication between Raspberry Pi, Buzzer, LEDs and message alerting system. Fig 3.1 shows the hardware setup.

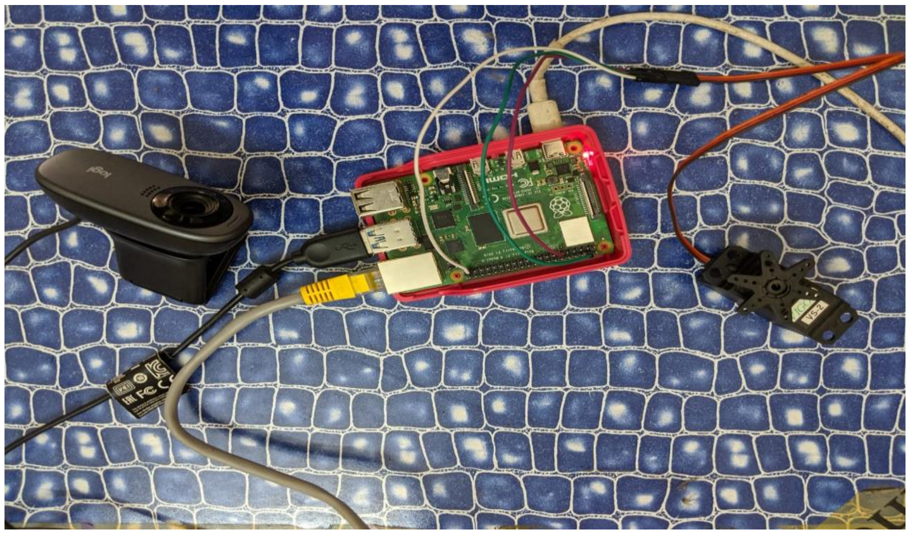


Fig 3.1 Hardware setup.

3.1 Haar Cascade algorithm

In this section, the base algorithm used to detect the face is discussed i.e., Haar Cascade Classifier is discussed. To Detect facial features, like mouth, nose, and eyes the Haar cascades shall be trained. To train these classifiers, this recommends the AdaBoost and Haar feature algorithms. Intel has developed an open source library that helped in making the usage of computer vision related algorithms like Open Computer Vision Library (OpenCV) easy. Face detection technology works using the following stages.

The classifiers are trained with the hundreds of positive sample views of a particular object and arbitrary negative images of the similar size . After training the classifier, it is now ready to detect the required object. To detect a specified object, in a given frame, the searches the entire picture and checks each and every area in search of the classifier. This process is generally used in image processing for object detection. OpenCV contains many pre-trained classifiers for full body, facial characteristics like nose, eyes, ears, etc.These classifiers are present in the xml files. In this project, we are using haarcascade_frontalface_default.xml for facial detection. Fig 3.2 shows line features and edge features used by the Haar cascade algorithm.

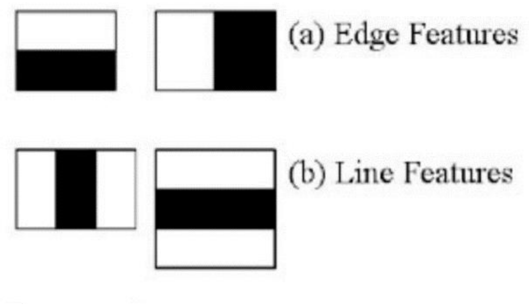


Fig 3.2 Line features and Edge features

- ➢ OpenCV contains a “trainer and detector”. If we have to “train” our own “classifiers” for any objects like bikes, cars etc.
- ➢ In this section we detect the face. “OpenCV” already consists of many pre-trained classifiers for smile, eyes, face etc. These “XML” files will be stored inside the “opencv/data/haarcascades/” folder.

3.1.1 Code Snippet

```
importnumpy as nap
import cv2 as bd
cascaded_frontalface = "haarcascades/haarcascade_frontalface_default.xml"
cascaded_face = bd.CascadeClassifier(cascaded_frontalface)
image_dira = 'person_data'
person_names = sys.argv[1]
path = os.path.join(self.image_dira, self.person_names)
defprocess_image(self, input_image):
framt = bd.flip(input_image, 1)
resizsed_widtth, resised_height = (112, 92)
if n<max_images:
g_image = bd.cvtColour(frame, pd.COLOUR_BRG2GRAY)
g_resised_image = bd.resise(gray1, ((int(round(g_image.shape[1] /
resizing_factor)), int(round(g_image.shape[0] / resizing_factor))))
```

3.2 LBPH algorithm

The recognition of face is done by the LBPH algorithm. It is a very efficient texture operator which takes the pixel of the images. With the help of the pixels that are present beside the particular pixel, the threshold value is calculated. And by comparing the values it is converted into a binary digits i.e. 1s and 0s. So, the characteristics of each picture can be represented by a single value with the help of the LBPH algorithm. The below fig 3.3 represents the features of LBPH, we can see the threshold values calculated in each pixel.

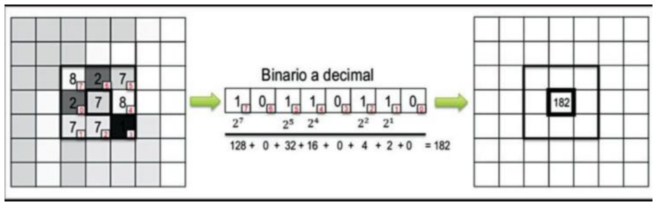


Fig 3.3 LBPH Features

3.2.1 Code Snippet

```
recognizer = cv2.face.createLBPHFaceRecognizer()
```

3.2.2 LBPH parameters:

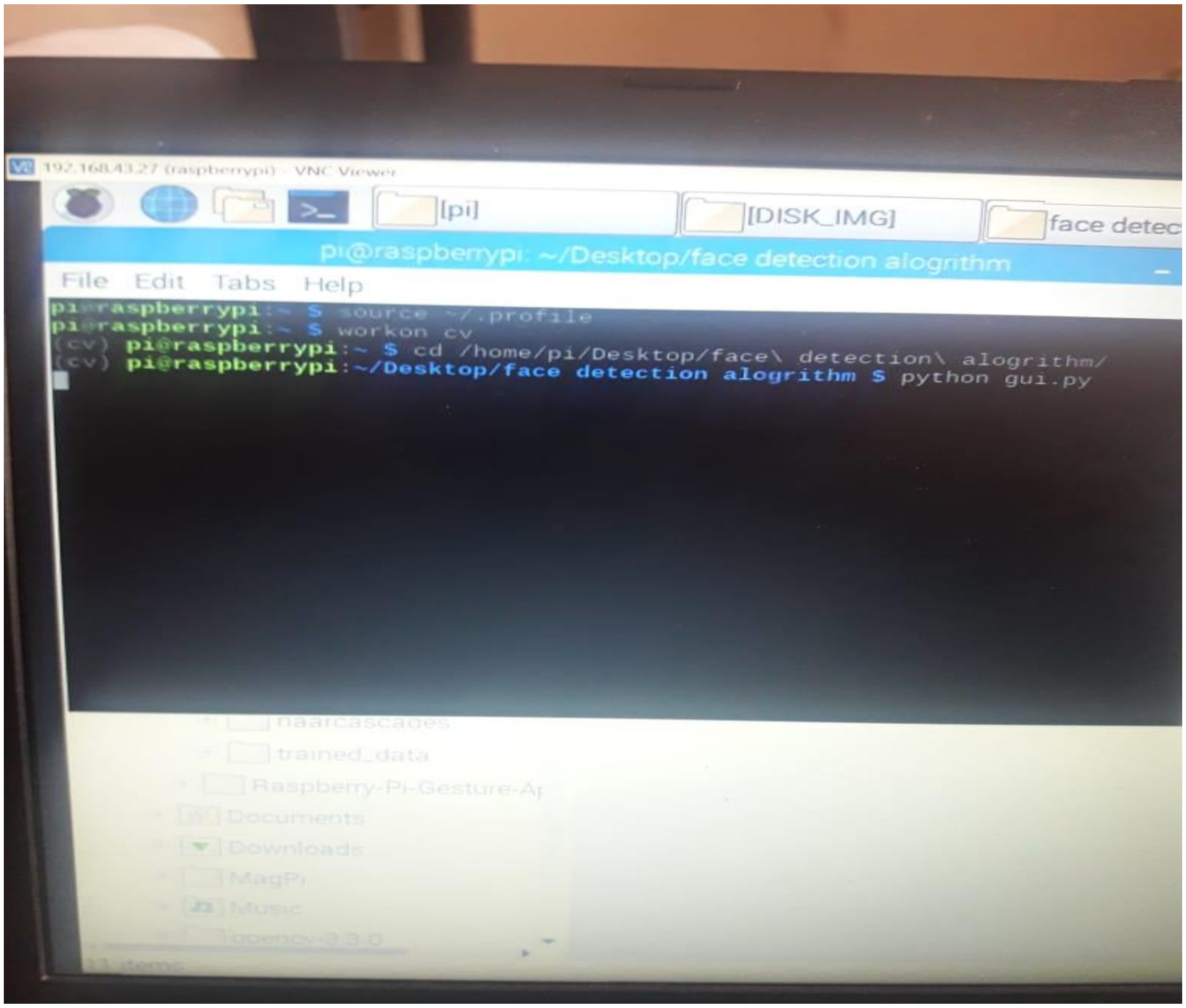
LBPH has various parameters like radius, neighbouring pixels, Grid X and Y. the various steps involed to recognize the face in this algorithm are:

1) Creating a dataset of the images we want to recognize.
2) Getting the images of faces
3) Training the algorithm and extracting the features.
4) Classification of each part of image by calculating the threshold.
5) Input the image we want to recognize.
6) Check if it matches with the database or not.

The LBP operator is used in every area in the image. It is defined in the window of 3x3.

3.3 Design Approach

The webcam sends the video input and is processed with the help of OpenCV library. The OpenCV checks for the face in the video frame, If it is detected, the processing and the coordinates of the face are provided by the OpenCV library. The processing sketch can confirm wherever the face is found within the “frame, relative to the center of the frame, and send this information through a serial affiliation” to the Raspberry Pi. The Raspberry Pi can “use the data from the processing sketch” and compares the given data with the existing pre trained images which are stored in the database. If the face is present in the data base we get a buzzer akert and also the picture of the criminal is sent to the authorities including the location where he is detected. The number of images requires for training can be decided by the user. We can use as many images as we want, the more number of images the bestter will be the prediction of the criminal. In this project we have used over a 20 images. The Fig 3.4 represents the commands in Raspberry pi.



```
pi@raspberrypi:~ $ source ~/.profile
pi@raspberrypi:~ $ workon cv
(cv) pi@raspberrypi:~ $ cd /home/pi/Desktop/face\ detection\ alogrithm/
(cv) pi@raspberrypi:~/Desktop/face detection alogrithm $ python gui.py
```

Fig 3.4 Implementation commands in Raspberry pi

3.3.1 Testing Camera Module

After we import the Open CV in our Raspberry Pi, to check if the camera module is working or not, we have to follow a simple procedure using the following code.

3.3.1.1 Code Snippet

```
importnumpy as nap
importcv2as bd
capp = bd.VideoCap(0)
capp.set(3,480)
capp.set(4,360)
while(1):
    rets, framt = capp.read()
    frames = bd.flip(framt, -1)
    g = bd.cvtColor(framt, cv2.COLOUR_BRG2GRAY)

    bd.imshow(' Framet', framt)
    bd.imshow('Gray_scale_image ', g)

if bd.waitKey(1) & 0xFF == ord('d'):
        break
capp.releease()
bd.destroyAllWindows()
```

3.3.2 Creating Dataset

The important task is creating a data set. So we have to train the algorithm with the dataset. We need to give the number of positive and negative images in the dataset. In out case, we have to store the images of the faces of the criminals we need to detect.

"Rapid Object Detection employing a Boosted Cascade of easy Features" in the year of 2001. It is a "machine learning based algorithm" in where a cascade function is trained from a number of positive images and also the negative images. It will then detect objects in other images. In the face detection, firstly, the algorithm needs a number of positive images (images that contain faces) and also the negative images (images that doesn't contain faces) in order to coach the classifier. Then we want to extract features from it. OpenCV arrives with an extra trainer as a detector. If we don't want to make our own classifier , OpenCV already contains many pre-trained classifiers for face, eyes, smile, etc . Those XML files are often download from haar cascades directory. After detecting the faces are captured so as to make a database of that exact person.

3.3.2.1 Code Snippet

```
import numpy as nap
import cv2 as bd
import sys
import os

frequency = 5
resizing_factor = 4
max_images = 50


class Add_person:
def __init__(self):
    cascaded_frontalface = "haarcascades/haarcascade_frontalface_default.xml"
cascaded_facea = bd.CascadeClassifier(cascade_frontalface)
    image_dira = 'persons_data'
    person_name = sys.argv[1]
path = os.path.join(image_dira, person_namea)
if not os.path.isdir(image_dira):
os.mkdir(image_dira)
if not os.path.isdir(patha):
os.mkdir(patha)
n = 0
    t = 0

defcaptured_images(self):
videoing = bd.VideoCapture(0)
while True:
t += 1
reta, area = videoing.read()
      input_image = nap.array(area)
```

```
            output_image = processing_images(input_image)
bd.imshow('Video', output_image)
if bd.waitKey(1) & 0xFF == ord('d'):
videoing.release()
bd.destroyAllWindows()
return

defprocessing_images(self, input_image):
area = bd.flip(input_image, 1)
        resizeda_width, resizeda_height = (112, 92)
ifn < max_images:
            g = bd.cvtColor(area, bd.COLOR_BGR2GRAY)
            g_resizeda_image = bd.resize(g, ((int(round(g.shape[1] / resizing_factor)),
int(round(g.shape[0] / resizing_factora)))))
persons = cascaded_facea.detectMultiScale(
                g_resizeda_image,
scaleFactor=1.1,
minNeighbors=5,
minSize=(30, 30),
flags=bd.CASCADE_SCALE_IMAGE
            )
if len(persons) > 0:
areasa = []
for (xa, ya, wa, ha) in persons:
areasa.append(wa * ha)
                max_area, idxa = max([(vala, idxa) for idxa, vala in enumerate(areasa)])
                face_sel = persons[idxa]

xa = face_sel[0] * resizing_factor
ya = face_sel[1] * resizing_factor
wa = face_sel[2] * resizing_factor
ha = face_sel[3] * resizing_factor

face = g[ya:ya + ha, xa:xa + wa]
                face_resizeda = bd.resize(face, (resized_width, resized_height))
                imga_no = sorted([int(fn[:fn.find('.')]) for fn in os.listdir(path) if fn[0] !=
'.'] + [0])[-1] + 1

ift % frequency == 0:
bd.imwrite('%s/%s.png' % (path, imga_no), face_resizeda)
n += 1
print ("Captured image: ", n)

bd.rectangle(area, (xa, ya), (xa + wa, ya + ha), (0, 255, 0), 3)
bd.puttheText(area, person_names, (x - 10, y - 10), sv.FONT_HERSHEY_PLAIN,
1, (0, 255, 0))
elifn == max_images:
print ("Data captured for training. Press 'q' to exit.")
            self.n+=1
return area


if __name__ == '__main__':
trainer = Add_person()
    trainer.capture_images()
print ("Type in next user to train, or press Recognize")
```

Where,

```
for (x,y,w,h) in faces:
cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
roi_g_image = g_image[y:y+h, x:x+w]
roi_color = img[y:y+h, x:x+w]
```

If face is detected , it returns the position of detected face as a rectangle with the leftup corner (xa,ya) and having “wa” as its Width and “ha” as its Height ==> (xa,ya,wa,ha). Fig 3.5 represents creating dataset

```
for (xa, ya, wa, ha) in persons:
areasa.append(wa * ha)
```

```
 max_area, idxa = max([(vala, idxa) for idxa, vala in enumerate(areasa)])
face_sel = persons[idxa]
```

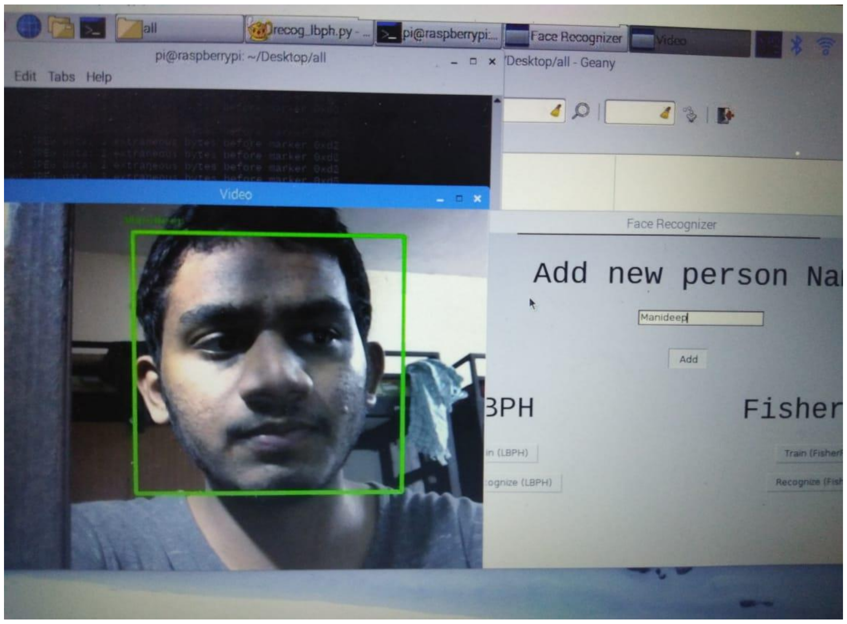


Fig 3.5 creating dataset

3.4 Summary

In this chapter the design of criminal identification using face detection has been described. We have used the Haar Cascade algorithm and OpenCV to identify the face and various facial features. The LBPH algorithm describes how the face is recognized from the database. In the next chapter the cost analysis of various components of the project is given.

CHAPTER 4
COST ANALYSIS

4.1 List of components and their cost

The costs of the various components used in this project are given in Table 4.1.

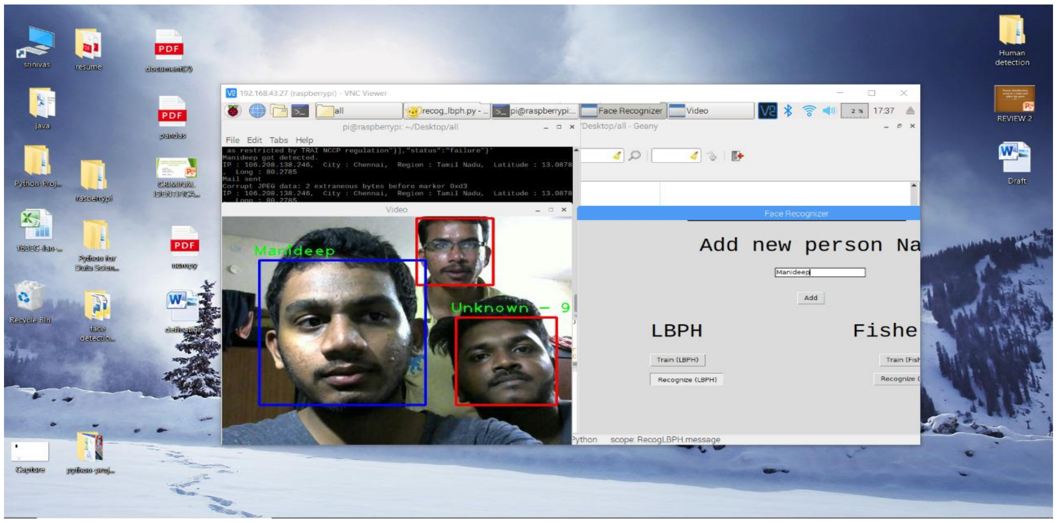
Table 4.1 List of components and their costs

Components	**cost**
Raspberry pi	Rs 3200
Web Camera module	Rs 1700
Bread board, Buzzer and LEDs	Rs 200
Jump wires	Rs 75
Total	Rs 5175

CHAPTER 5

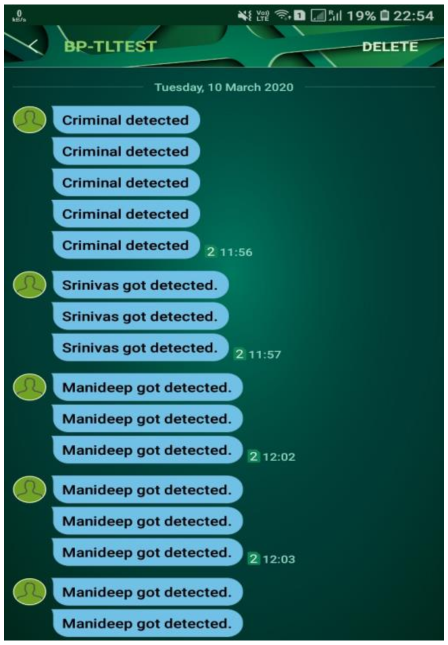
RESULTS AND ANALYSIS

If a person is detected in front of the camera, the system recognizes the face and compare it with images that are present in the database. This system can recognize the multiple faces at an instant and alert the authorities simultaneously. If the face is recognized by the system, and the face matches with that of the images in the criminal database, then the authorities are alerted by the buzzer sound, the SMS, Gmail which contains the image, location, name of the criminal is sent to the authorities. In Fig 5.1 we can see that the known face is recognized and the name is displayed and the unknown faces are not detected. Fig 5.2 and Fig 5.3 shows the message that it sent to the mobile through SMS and Gmail.



```
 as restricted by TRAI NCCP regulation"}],"status":"failure"}'
Manideep got detected.
IP : 106.208.138.246,  City : Chennai,  Region : Tamil Nadu,  Latitude : 13.0878
,  Long : 80.2785
Mail sent
Corrupt JPEG data: 2 extraneous bytes before marker 0xd3
IP : 106.208.138.246,  City : Chennai,  Region : Tamil Nadu,  Latitude : 13.0878
,  Long : 80.2785
```

Fig 5.1 Recognition of known face

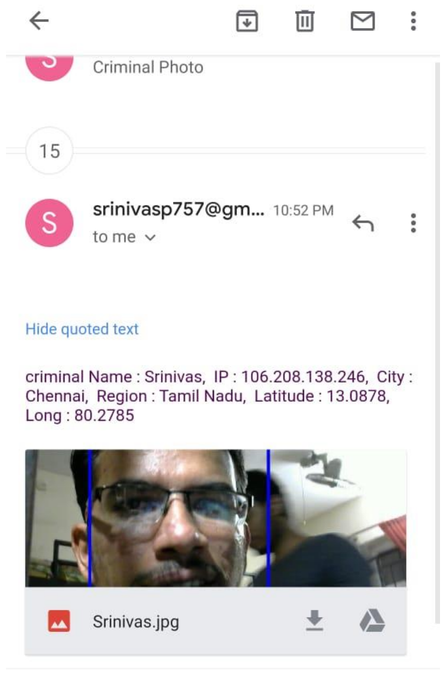


BP-TLTEST

Tuesday, 10 March 2020

Criminal detected
Criminal detected
Criminal detected
Criminal detected
Criminal detected 11:56
Srinivas got detected.
Srinivas got detected.
Srinivas got detected. 11:57
Manideep got detected.
Manideep got detected.
Manideep got detected. 12:02
Manideep got detected.
Manideep got detected.
Manideep got detected. 12:03
Manideep got detected.
Manideep got detected.

Fig 5.2 Text message when criminal is detected



Criminal Photo

srinivasp757@gm... 10:52 PM
to me

Hide quoted text

criminal Name : Srinivas, IP : 106.208.138.246, City : Chennai, Region : Tamil Nadu, Latitude : 13.0878, Long : 80.2785

Srinivas.jpg

Fig 5.3 Gmail alert

5.1 Summary

In this chapter the analysis of criminal identification using face detection is implemented and we have seen the results. In our project, when we take more number of datasets of a person, the accuracy of the face detection and recognition is increased. Also we have seen that when an unknown person comes in front of the prototype, the picture of that unknown person is sent through Gmail. If the known person comes in front of the prototype, the security system will allow the person inside. This mechanism is shown using the servo motor as a door. In the next chapter, the conclusion and the future work of the project is presented

CHAPTER 6

CONCLUSION AND FUTURE WORK

Therefore we conclude the thesis on Criminal Identification using Face detection and suggests additional functionalities for future implementation.

6.1 Conclusion

The face recognition has vast area of applications. We have designed an efficicient method of detecting the faces and we have used this concept of machine learning and face detection to recognize the criminals trespassing and alert the authorities in a very less time by notifying them so that they can take the action as soon as possible. The efficiency and the precision of this system is satisfying and is constructed at a very less cost. .

6.2 Future work

The present project can be improved in various areas and can be implemented in a lot of applications. Installing them in the entries and adding a servo motor to them can be used as a security doors, where only specified people can be identified. By further improving the face recognition by using a more efficient algorithm, we can detect the facial expressions of the criminal and identify if that particular person if he is committing a crime or telling lies etc.

CHAPTER 7

APPENDIX

7.1 Face Dataset

```
import numpy as nap
import cv2 as sv
import sys
import os

frequency = 5
resizing_factor = 4
max_images = 50


class Add_person:
    def __init__(self):
        cascaded_frontalface =
"haarcascades/haarcascade_frontalface_default.xml"
  cascaded_facea = sv.CascadeClassifier(cascade_frontalface)
          image_dira = 'persons_data'
          person_namea = sys.argv[1]
          path = os.path.join(image_dira, person_namea)
            if not os.path.isdir(image_dira):
              os.mkdir(image_dira)
          if not os.path.isdir(patha):
              os.mkdir(patha)
n = 0
        t = 0

      def captured_images(self):
          videoing = sv.VideoCapture(0)
```

```
         while True:
t += 1
           reta, area = videoing.read()
           input_image = nap.array(area)
           output_image = processing_images(input_image)
           sv.imshow('Video', output_image)
             if sv.waitKey(1) & 0xFF == ord('d'):
               videoing.release()
               sv.destroyAllWindows()
               return

   def processing_images(self, input_image):
       area = sv.flip(input_image, 1)
       resizeda_width, resizeda_height = (112, 92)
       if n < max_images:
           g = sv.cvtColor(area, sv.COLOR_BGR2GRAY)
           g_resizeda_image = sv.resize(g, ((int(round(g.shape[1] /
resizing_factor)), int(round(g.shape[0] / resizing_factora)))))
           persons = cascaded_facea.detectMultiScale(
               g_resizeda_image,
               scaleFactor=1.1,
               minNeighbors=5,
               minSize=(30, 30),
               flags=sv.CASCADE_SCALE_IMAGE
           )
           if len(persons) > 0:
               areasa = []
               for (xa, ya, wa, ha) in persons:
                   areasa.append(wa * ha)
               max_area, idxa = max([(vala, idxa) for idxa, vala in
enumerate(areasa)])
               face_sel = persons[idxa]

               xa = face_sel[0] * resizing_factor
```

```
                ya = face_sel[1] * resizing_factor
                wa = face_sel[2] * resizing_factor
                ha = face_sel[3] * resizing_factor

                face = g[ya:ya + ha, xa:xa + wa]
                face_resizeda = sv.resize(face, (resized_width,
resized_height))
                imga_no = sorted([int(fn[:fn.find('.')]) for fn in os.listdir(path)
if fn[0] != '.'] + [0])[-1] + 1

                if t % frequency == 0:
 sv.imwrite('%s/%s.png' % (path, imga_no), face_resizeda)
n += 1
print ("Captured image: ", n)

                sv.rectangle(area, (xa, ya), (xa + wa, ya + ha), (0, 255, 0), 3)
          sv.putText(area, person_names, (x - 10, y - 10),
sv.FONT_HERSHEY_PLAIN, 1, (0, 255, 0))
        elif n == max_images:
print ("Data captured for training. Press 'q' to exit.")
            self.n+=1
        return area


if __name__ == '__main__':
    trainer = Add_person()
    trainer.capture_images()
    print ("Type in next user to train, or press Recognize")
```

7.2 Face Training

```
importnumpyasnap
importcv2ashv
importsys
```

```
importos

class TrainLBPHFace:
    def __init__(self):
        cascaded_frontalface =
"haarcascades/haarcascade_frontalface_default.xml"
face_cascade = hv.CascadeClassifier(cascade_frontalface)
        image_dira = 'persons_data'
         os.path.isdir('trained_data'):
            os.mkdir('trained_data')
model = hv.face.LBPHFaceRecognizer_create()

    def lbph_data(self):
        images = []
        targets = []
        indexs = 0

        for (sub_force, force, file) in os.walk(image_dira):
            for subdira in force:
 image_paths = os.path.join(image_dira, subdira)
                for fn in os.listdir(image_path):
                    paths = image_path + '/' + fn
                    tags = indexs
                    images.append(hv.imread(paths, 0))
                    targets.append(int(tags))
                indexs += 1
        (images, targets) = [nap.array(iteme) for iteme in [images, targets]]

        self.model.train(images, targets)
        self.model.write('trained_data/lbph_trained_data.xml')
        print ("Training completed successfully")
        return
```

```
if __name__ == '__main__':
    trainings = TrainLBPHFace()
    trainings.lbph_data()
    print ("Type in next user to train, or press Recognize")
```

7.3 Face Recognition

```
importnumpyasnap
importcv2asbd
importsys
importos

import RPi.GPIO as hk
from time import sleep

resizing_factors = 4

class RecogLBPH:
    def __init__(self):
        cascade_frontalface =
"haarcascades/haarcascade_frontalface_default.xml"
cascaded_face = bd.CascadeClassifier(cascaded_frontalface)
image_dir = 'persons_data'
        model = bd.face.LBPHFaceRecognizer_create()
        person_names = []

    def load_trained_data(self):
        names = {}
        key = 0
        for (sub_d, d, filet) in os.walk(image_dir):
            for subdira in d:
                names[key] = subdira
                key += 1
        self.names = names
        self.model.read('trained_data/lbph_trained_data.xml')

    def show_video(self):
        videoing = bdVideoCapture(0)
        while True:
            reta, framt = videoing.read()
            input_image = nap.array(framt)
            output_image, self.person_names =
self.processing_image(input_image)
            bd.imshow('Video', output_image)

            if bd.waitKe &0xFF == ord('d'):
                videoing.release()
                bd.destroyAllWindows()
                return

    def processing_imaga(self, input_image):
        framt = bd.flip(input_image,1)
        resised_width, resised_height = (112, 92)
        g_image = bd.cvtColor(frame, bd.COLOR_BGR2GRAY)
        g_resized_image = bd.resize(g_image,
((int(round(g_image.shape[1] / resizing_factor)),
int(round(g_image.shape[0] / resizing_factor)))))
        image_specs = self.cascaded_face.detectMultiScale(
                g_resised_image,
                scale_Factors=1.3,
                minNeighbours=4,
                MinSise=(30, 30),
                flags=bd.CASCADESCALEIMAGE
                )
        persons = []
        for i in range(len(image_specs)):
            face_i = image_specs[i]
            xa = face_i[0] * resizing_factors
            ya= face_i[1] * resizing_factors
            wa = face_i[2] * resizing_factors
            ha = face_i[3] * resizing_factors
            face = g_image[y:y+h, x:x+w]
            faceresized = bd.resize(face, (resizedwidth, resizedheight))
            confidence = self.model.predict(faceresized)
            if confidence[1]<80:
                person = self.names[confidence[0]]
              bd.rect(framt, (xa,ya), (xa+wa, ya+ha), (255, 0, 0), 3)
                bd.putText(framt, '%s ' % (person), (xa-10, ya-10),
bd.FONT_HERSHYPLAIN,2,(0, 255, 0), 2)
                bd.imwrite('%s.jpg'%(person),framt)
                text = person+" got detected."
                print(text)
            else:
                person = 'Unknown'
              bd.rect(framt, (xa,ya), (xa+wa, ya+ha), (0, 0, 255), 3)
                bd.put_Text(framt, '%s' % (person), (xa-10, ya-10),
bd.FONT_HERSHYPLAIN,2,(0, 255, 0), 2)
            persons.append(person)
        return (framt, persons)


if __name__ == '__main__':
    recognising = RecogLBPH()
    recognising.load_trained_data()
    print ("Press 'q' to quit video")
    recognizer.show_video()
```

7.4 Text message code

```
def message(self,text):
        import urllib.request
        import urllib.parse

        Location = self.location()
        p = 'Criminal Name : '+text
        Text = p+',  '+''.join(Location)
        out = Text.split(',')

defsendSMS(akey, numas, sendst, mes):
            datat = urllib.parse.urlencode({'Apikey': akey, 'Numbers': numas,
                              'Msg': mes, 'Senderr':sendst})
            datat = data.encode('utf-8')
requezt = urllib.request.Request("https://api.textlocal.in/send/?")
            fp = urllib.requestt.urllopen(requezt, data)
frat = fp.read()
            return frat
akey= 'MTNASm0yU60-wDkipeHSe1unQUu0nOfK19rTsdO1aR'
respa = sendSMS(akey, '919866817037', 'TXTLCL', out)
```

7.5 Gmail code

```
import smtplib
        import os
        import imghdr
        from email.message import EmailMessage

        Location = self.location()
email_address = 'srinivasp757@gmail.com'
```

```
        password = 'SRin,**01'

msg = EmailMessage()
msg['Subject'] = 'Criminal Detected'
msg['From'] = email_address
msg['To'] = 'srinivas.pasupula01@gmail.com'

        p = 'criminal Name : '+person
        Text = p+',  '+''.join(Location)

msg.set_content(Text)

        with open(person+'.jpg','rb') as f:
file_data = f.read()
file_type = imghdr.what(f.name)
file_name = f.name

msg.add_attachment(file_data, maintype='image', subtype=file_type,
filename=file_name)

with sp.SMTPSSL('stmp.gmail.com',465) as stmp:
smtp.login(email_addresst,passwordd)
smtp.send_message(mga)
            print('Mail sent')
```

7.6 Location code

```
import requests

        res = requests.get('https://ipinfo.io/')
        data = res.json()

        city = data['city']
```

```
ip = data['ip']
        region = data['region']
loc = data['loc']
lat,log = loc.split(',')
        output = 'IP : '+ip+',  '+'City : '+city+',  '+'Region : '+ region+',
'+'Latitude : '+lat+ ',  '+'Long : '+log
        print(output)
        return output
```

7.7 Alerting System code

```
importRPi.GPIO as hk
fromtime import sleep

if confidence[1]<80:
hk.setwarnings(False)
hk.setmode(hk.BOARD)
hk.setup(12,hk.OUT,initial=hk.LOW)
hk.setup(7,hk.OUT)
                person = self.names[confidence[0]]
             cv2.rectangle(framt, (xa,ya), (xa+wa, ya+ha), (255, 0, 0), 3)
               cv2.putText(framt, '%s ' % (person), (x-10, y-10),
cv2.FONT_HERSHEY_PLAIN,2,(0, 255, 0), 2)
                cv2.imwrite('%s.jpg'%(person),framt)
                text = person+" got detected."
                messages = self.message(text)
                print(text)
mail_message = self.mail_message(person)
hk.output(12, hk.HIGH)
                sleep(0.5)
hk.output(12, hk.LOW)
```

```
                sleep(0.5)
hk.output(7, hk.HIGH)
                sleep(0.5)
hk.output(7, hk.LOW)
                sleep(0.5)
hk.output(12, hk.HIGH)
                sleep(0.5)
hk.output(12, hk.LOW)
                sleep(0.5)
hk.cleanup()

            else:
hk.setwarnings(False)
hk.setmode(hk.BOARD)
hk.setup(10,hk.OUT,initial=hk.LOW)
                person = 'Unknown'
             cv2.rectangle(framt, (xa,ya), (xa+wa, ya+ha), (0, 0, 255), 3)
                cv2.putText(frame, '%s' % (person), (x-10, y-10),
cv2.FONT_HERSHEY_PLAIN,2,(0, 255, 0), 2)
hk.output(10, hk.HIGH)
                sleep(1)
hk.output(10, hk.LOW)
                sleep(1)
hk.cleanup()
persons.append(person)
```

7.8 code with recognizer,Text message and Gmail

```
importnumpyas nap
importcv2 as bd
importsys
```

```
importos

importRPi.GPIO as hk
fromtime import sleep

resizing_factors = 4

class RecogLBPH:
   def __init__(self):
      cascade_frontalface =
"haarcascades/haarcascade_frontalface_default.xml"
cascaded_face = bd.CascadeClassifier(cascaded_frontalface)
image_dir = 'persons_data'
      model = bd.face.LBPHFaceRecognizer_create()
      person_names = []

   def load_trained_data(self):
      names = {}
      key = 0
      for (sub_d, d, filet) in os.walk(image_dir):
         for subdira in d:
            names[key] = subdira
            key += 1
      self.names = names
      self.model.read('trained_data/lbph_trained_data.xml')

   def show_video(self):
      videoing = bd.VideoCapture(0)
      while True:
         reta, framt = videoing.read()
         input_image = nap.array(framt)
         output_image, self.person_names =
self.processing_image(input_image)
         bd.imshow('Video', output_image)
```

```
          if bd.waitKey(1) & 0xFF == ord('d'):
             videoing.release()
             bd.destroyAllWindows()
             return

   def processing_imaga(self, input_image):
       framt = bd.flip(input_image,1)
       resised_width, resised_height = (112, 92)
       g_image = bd.cvtColor(frame, bd.COLOR_BGR2GRAY)
       g_resized_image = bd.resize(g_image, ((int(round(g_image.shape[1] /
resizing_factor)), int(round(g_image.shape[0] / resizing_factor)))))
       image_specs = self.cascaded_face.detectMultiScale(
             g_resised_image,
             scaleFactors=1.3,
             minNeighbors=4,
             MinSize=(30, 30),
             flags=bd.CASCADE_SCALE_IMAGE
             )
       persons = []
       for i in range(len(image_specs)):
          face_i = image_specs[i]
          xa = face_i[0] * resizing_factors
          ya= face_i[1] * resizing_factors
          wa = face_i[2] * resizing_factors
          ha = face_i[3] * resizing_factors
          face = g_image[y:y+h, x:x+w]
          face_resized = bd.resize(face, (resized_width, resized_height))
          confidence = self.model.predict(face_resized)
          if confidence[1]<80:
hk.setwarnings(False)
hk.setmode(hk.BOARD)
hk.setup(12, hk.OUT,initial= hk.LOW)
hk.setup(7, hk.OUT)
             person = self.names[confidence[0]]
           cv2.rectangle(framt, (xa,ya), (xa+wa, ya+ha), (255, 0, 0), 3)
```

```
                cv2.putText(framt, '%s ' % (person), (x-10, y-10),
cv2.FONT_HERSHEY_PLAIN,2,(0, 255, 0), 2)
                cv2.imwrite('%s.jpg'%(person),framt)
                text = person+" got detected."
                messages = self.message(text)
                print(text)
mail_message = self.mail_message(person)
hk.output(12, hk.HIGH)
                sleep(0.5)
hk.output(12, hk.LOW)
                sleep(0.5)
hk.output(7, hk.HIGH)
                sleep(0.5)
hk.output(7, hk.LOW)
                sleep(0.5)
hk.output(12, hk.HIGH)
                sleep(0.5)
hk.output(12, hk.LOW)
                sleep(0.5)
hk.cleanup()
            else:
hk.setwarnings(False)
hk.setmode(hk.BOARD)
hk.setup(10, hk.OUT,initial= hk.LOW)
                person = 'Unknown'
              cv2.rectangle(framt, (xa,ya), (xa+wa, ya+ha), (0, 0, 255), 3)
                cv2.putText(frame, '%s' % (person), (x-10, y-10),
cv2.FONT_HERSHEY_PLAIN,2,(0, 255, 0), 2)
hk.output(10, hk.HIGH)
                sleep(1)
hk.output(10, hk.LOW)
                sleep(1)
hk.cleanup()
persons.append(person)
return (frame, persons)
```

```
    def message(self,text):
        import urllib.request
        import urllib.parse

        Location = self.location()
        p = 'Criminal Name : '+text
        Text = p+',  '+''.join(Location)
        out = Text.split(',')

        def sendSMS(apikey, numbers, sender, message):
            data = urllib.parse.urlencode({'apikey': apikey, 'numbers': numbers,
                                 'message': message, 'sender':sender})
            data = data.encode('utf-8')
request = urllib.request.Request("https://api.textlocal.in/send/?")
            f = urllib.request.urlopen(request, data)
            fr = f.read()
            return fr

        apikey = 'MTNASm0yU60-wDkipeHSe1unQUu0nOfK19rTsdO1aR'
        resp = sendSMS(apikey, '919866817037', 'TXTLCL', out)
        #print(resp)

    def mail_message(self,person):
import smtplib as sp
        import os
        import imghdr
        from email.message import EmailMessage as em

        Locations = self.location()
email_addresst = 'srinivasp757@gmail.com'
        passwordd = 'SRin,**01'

msga = EmailMessage()
msga['Subject'] = 'Criminal Detected'
```

```
msga['From'] = email_addresst
msga['To'] = 'srinivas.pasupula01@gmail.com'

        p = 'criminal Name : '+person
        text_to_be_sent = p+',  '+''.join(Locations)

msga.set_content(text_to_be_sent)

        with open(person+'.jpg','rb') as f:
file__data = f.read()
file__type = imghdr.what(f.name)
file__name = f.name

msga.add_attachment(file__data, maintype='image', subtype=file__type,
filename=file__name)

        with sp.SMTP_SSL('smtp.gmail.com',465) as smtp:
smtp.login(email_addresst,passwordd)
smtp.send_message(msga)
            print('Mail sent')

    def location(self):
        import requests

        res = requests.get('https://ipinfo.io/')
        data = res.json()

        city = data['city']
        ip = data['ip']
        region = data['region']
        loc = data['loc']
        lat,log = loc.split(',')
        output = 'IP : '+ip+',  '+'City : '+city+',  '+'Region : '+ region+',
'+'Latitude : '+lat+ ',  '+'Longitude : '+log
        print(output)
        return output



if __name__ == '__main__':
    recognizer = RecogLBPH()
    recognizer.load_trained_data()
    print ("Press 'q' to quit video")
    recognizer.show_video()
```

CHAPTER 8

BIBLIOGRAPHY

1. Swathi. V, Steven Fernandes," Raspberry Pi Based Human Face Detection", published in International of advanced research in computer and communication engineering Vol. 4, Issue 9, September 2015
2. Víctor Bautista Saiz; FernanGallego, GPU: Application for CCTV systems, Published in: Security Technology (ICCST), 2014
3. Ping Hsin Lee, VivekSrinivasan, and ArvindSundararajan. Face Detection, Final Year Project, Stanford University, 2014
4. Rakish, V. S., P. R. Suresh, and Studfish N. George, "An improved real-time surveillance system for home security system using Beagle Board SBC, Sigsbee, and FTP web server," IEEE Int. Con, 2012, pp. 1240-1244.
5. Ansari, AamirNizam, Mohamed Sedky, Neelam Sharma, and AnuragTyagi, "An Internet of things approach for motion detection using Raspberry Pi," IEEE Int. Con. Intelligent Computing and the Internet of Things, 2014, pp. 131-134.
6. K. Gopalakrishnan, V. Sathish Kumar "embedded image capturing system using the raspberry pi system" International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) Volume 3, Issue2, March– April 2014.
7. Anoop Mishra "Embedded Image Capturing & Digital Converting Process using Raspberry Pi System interfacing and Comparison of Generation 2 versus Generation 1 models in Raspberry Pi" et al,/(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 6 (2), 2015, 1798-1801.
8. Shamanth G S Ashwin Kashyap ''Face detection using Raspberry pi and python''.NCPD 2016 July 2016.
9. S. C. Gaddam, N. V. K. Ramesh and Hema Dhanekula, ''Face Recognition Based Attendance Management System with Raspberry PI 2 using Eigen Faces Algorithm, ''ARPN journal vol -11, NO.13, july 2016.

10. Monekalla s praveena, Tallapelli Suresh “Face Recognition System using Raspberry PI and Principle Component Analysis” ISSN Neel RamakantBorkar,Sonia Kuwelkar “Real-Time Implementation Of Face Recognition System”Proceedings of the IEEE 2017 International Conference on Computing Methodologies and Communication
11. Sourav Roy, Md Nasir Uddin, Md Zahirul Haque, Md Jahidul Kabir “Design and Implementation of the Smart DoorLock System with Face Recognition Method using the Linux Platform Raspberry Pi.
12. Ishita Gupta, Varsha Patil, Chaitali Kadam, Shreya Dumbre “Face Detection and Recognition using Raspberry Pi” 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)19-21 December 2016, AISSMS, Pune, India
13. Zhang Jian, Song Wan-juan .Face Detection for Security Surveillance System”. The 5th International Conference on Computer Science & Education Hefei, China. August 24–27, 2010
14. Dr. Priya Guptaa, Nidhi Saxenaa, Meetika Sharmaa, Jagriti Tripathia. “Deep Neural Network for Human Face Recognition” Published Online January 2018 in MECS AISSMS, Pune, India.
15. Thulluri Krishna Vamsi, Kanchana Charan Sai, VijayalakshmiM ”Face Recognition based door unlocking system using Raspberry Pi, Volume 5, Issue 2, September, 2015.

BIODATA

Name : P.Srinivas
Mobile Number : 8792092030
E-mail : srinivas.pasupula01@gmail.com
Permanent Address : Hospet, Karnataka

Name : M.Manideep
Mobile Number : 8792092030
E-mail : manideepm1998@gmail.com
Permanent Address : Peddapalli, Telangana

Name : K.Hemanth Kumar
Mobile Number : 7397327918
E-mail : hemanth02.kadiyala@gmail.com
Permanent Address : Vijayawada,
Andhra Pradesh