Intruder Alert System

By:

Aditya Abasaheb Patil (11197003)

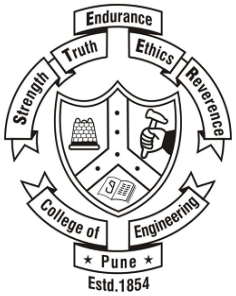
Aishwarya Pandurang Kadam (111907005)

Mohammad Aman (111907008)

Under the guidance of :

Dr. S.P. Mahajan

Dr. Mrs. P.P. Shingare



Department of Electronics and Telecommunications Engineering

College of Engineering, Pune

2022

Abstract

This report entails the details on how an Android Mobile Camera is interfaced with Raspberry Pi Microcontroller with the aid of IP Webcam Application for Face Recognition. When a person is found in front of the camera, and is not the user then the face is declared as unrecognised.

On detecting the unrecognised person, the system takes a snapshot and is transmitted to the owner as an image notification via Telegram. Envisaging the difficult time of the common masses facing burglary and robbery; safeguarding our valuable possession is the need of the hour. Also, the existing technologies are expensive and unaffordable to the lower sections of the epechlon. Hence, this is the main motive behind the project. The authors of this report have explored the interfacing of Android Mobile Camera on Raspberry Pi, Image Processing using Python and OpenCV.

This report will allow you to traverse through the technology behind how Interfacing of cameras with Raspberry Pi works and how to set the Raspberry Pi. We have included the images of various components, their brief information enough for even a novice to grasp the concept, and simulation photos that demonstrate the result. There is a flowchart in the report that guides you through the entire procedure in a gist.

Contents

1. INTRODUCTION

1.1 Brief History

A facial recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image. For facial recognition we've taken the aid of tools like Python Programming library, Image Processing and OpenCV.

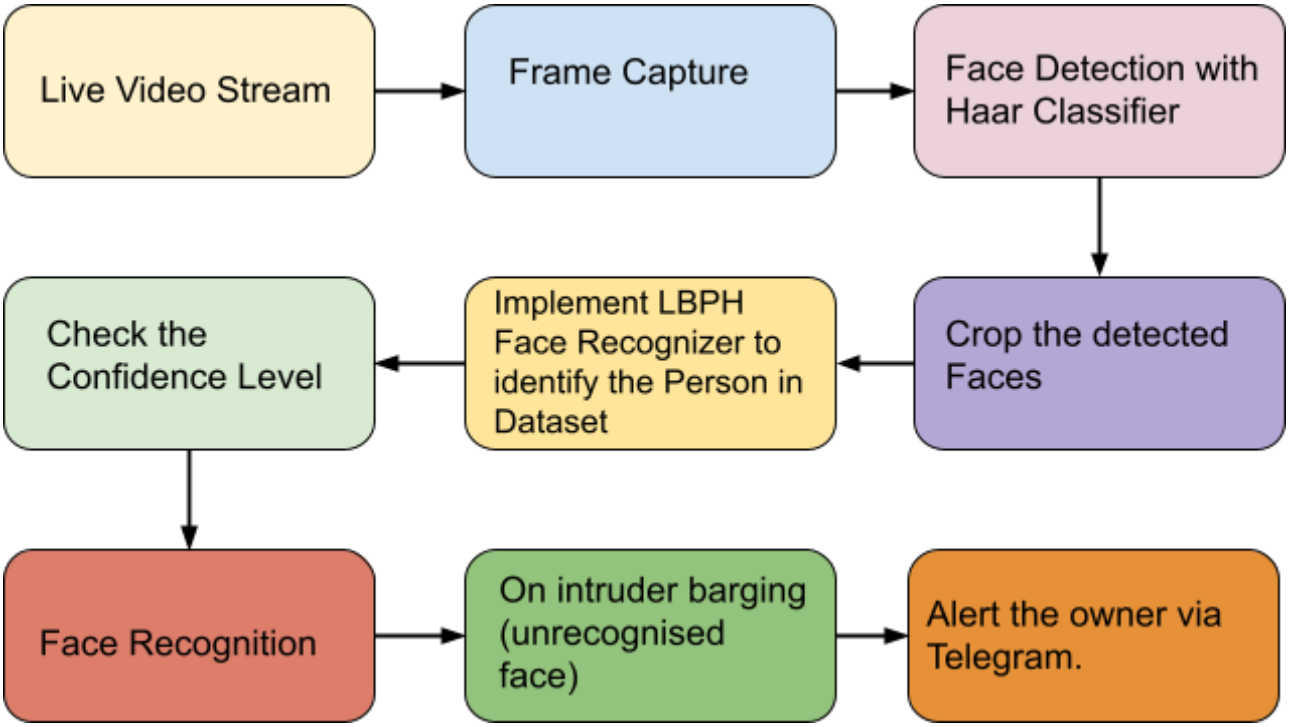
The earliest pioneers of facial recognition were Woody Bledsoe, Helen Chan Wolf and Charles Bisson. In 1964 and 1965, Bledsoe, along with Wolf and Bisson began work using computers to recognise the human face. Their initial work involved the manual marking of various "landmarks" on the face such as eye centres, mouth etc. These were then mathematically rotated by a computer to compensate for pose variation. The distances between landmarks were also automatically computed and compared between images to determine identity.

In the 1970s Goldstein, Harmon and Lesk extended the work to include 21 specific subjective markers including hair colour and lip thickness in order to automate the recognition. This led to the advancements in Facial Recognition technology.

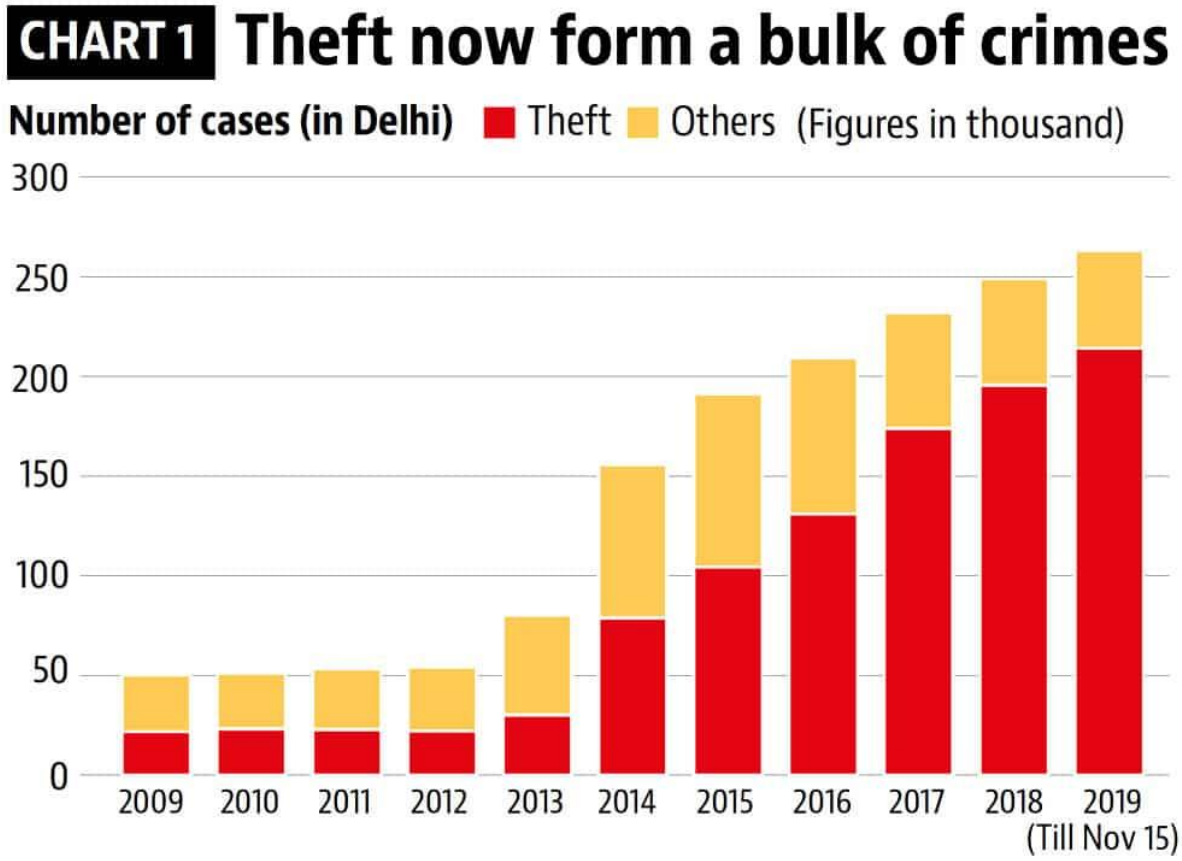
It was in the 1980s and 1990s that advanced Mathematical techniques like Linear Algebra were implemented to acquire fine precision in the results of Face Recognition. This led to development of Biometric softwares. Further, there have been a lot of advancements in this sector.

Today, Face Recognition plays a vital role in maintaining the security of nations, helps in biometric systems and is available to the common masses via their cell phones.

1.2 Overall Block Diagram of the Project



1.3 Problem Definition:



As we can see thefts are increasing year after year there seems an absence of a perfect system to avoid the robbery.This is the case of just one state in India.The graph of India would be more terrible than this graph. Due to technological advancements burglars are becoming more technology oriented which helps them in their work.

So we have to make a system that detects any intruder trying to open the locker and take away the contents in the locker. The camera which is fitted on the locker captures every person in its surroundings and alerts the user of an unknown person opening the locker.

1.4 Objectives

To interface Mobile Camera with Raspberry Pi using the IP Webcam, an Android Application. Recognise the face with the aid of Python, Image ProcessingOpenCV and Haar Cascade Frontal Face Features. On detecting the faces, check if the face is known or else alert the owner using Telegram Bot.

15. Technical Specifications/ Features:

Raspberry Pi 3B+ Specifications:

Specifications

Processor:	Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4 GHz
Memory:	1GB LPDDR2 SDRAM
Connectivity:	■ 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ■ Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps) ■ 4 × USB 2.0 ports
Access:	Extended 40-pin GPIO header
Video & sound:	■ 1 × full size HDMI ■ MIPI DSI display port ■ MIPI CSI camera port ■ 4 pole stereo output and composite video port
Multimedia:	H.264, MPEG-4 decode (1080p30); H.264 encode (1080p30); OpenGL ES 1.1, 2.0 graphics
SD card support:	Micro SD format for loading operating system and data storage
Input power:	■ 5 V/2.5 A DC via micro USB connector ■ 5 V DC via GPIO header ■ Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Environment:	Operating temperature, 0–50 °C

How to setup a Raspberry Pi:

1 Downloading Raspbian(OS) and Raspbian Image writer for flashing OS

2 Writing the image

3 Configuring Wifi and Local timezone

4 Updating the firmware

5) Enabling VNC server on Raspberry Pi

6) Installing VNC application on Laptop

7) Setting VNC resolution and connecting raspberry pi through ip address

Python's Image Processing Libraries

There are several Python libraries related to Image processing and Computer vision. The ones that we will use in this report are:

- PIL/Pillow:

This library is mainly appropriate to manipulate simple operations (rotation, resizing, etc.) and very basic image analysis(for example; histogram).

- OpenCV:

OpenCV is an open source computer vision library and it is written in C and C++ which runs under Windows, Linux, Mac OS X, iOS and Android. It is a user-friendly, simple tool to interact with images for classification, segmentation and analysis. Interfaces are available for Python, Java, Ruby, Matlab, and other languages. Although it is written in C/C++, Python bindings are added during the installation. A very simple program used just to show an image can be written as follows:

```
import numpy as np
import cv2
img = cv2.imread('flower.jpg')
cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

As OpenCV is nowadays the most suitable library for computer vision we will use it in the remaining part of the report.

- os:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.

- numpy:

NumPy is a Python library used for working with arrays.It also has functions for working in the domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

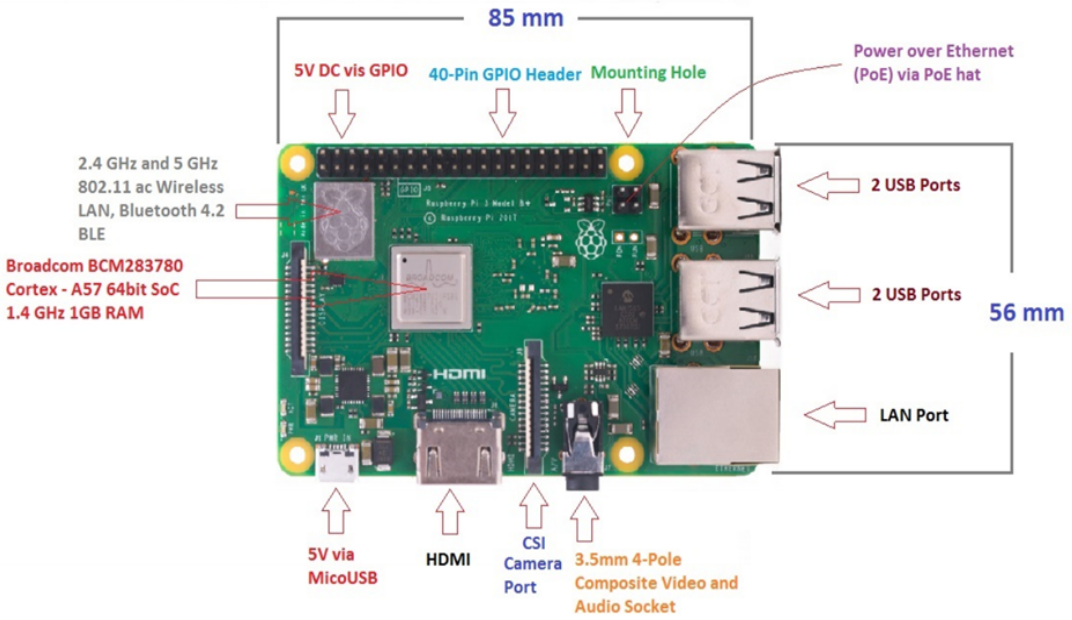
- Telepot

Telepot helps you build applications for Telegram Bot API. It works on Python 2.7 and Python 3. For Python 3.5+, it also has an async version based on asyncio.

2. Hardware/Software Specifications

2.1 Hardware Architecture:

1. Raspberry Pi



The authors of the project used Raspberry Pi 3 Model B+.
It has 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHz and 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. It maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B.

2. IP Webcam

The authors of this report used the IP Webcam Application. The android IP webcam integration connects with Android IP Webcam to turn any Android phone or tablet into a network camera with multiple viewing options. Due to resource constraints, the authors preferred to use it. The readers of the report can use any other viable option to integrate the camera like the Raspberry Pi Camera that can be mounted on the microcontroller.

3. **Power Charger**

As per the datasheet of raspberry pi3 B+ any charger which provides a constant supply of 5V/2.5A can be used to power raspberry pi.
We can use a USB connector instead but it gives a warning of low voltage continuously on the screen.
So it is preferred to use a power charger to turn on raspberry pi.

4. **Micro SD Card:**

The microSD removable miniaturised Secure Digital flash memory cards were originally named T-Flash or TF, abbreviations of TransFlash. TransFlash and microSD cards are functionally identical allowing either to operate in devices made for the other. MicroSD (and TransFlash) cards are electrically compatible with larger SD cards and can be used in devices that accept SD cards with the help of a passive adapter, which contains no electronic components, only metal traces connecting the two sets of contacts. Unlike the larger SD cards, microSD does not offer a mechanical write protect switch, thus an operating-system-independent way of write protecting them does not exist in the general case. SanDisk conceived of microSD when its Chief Technology Officer(CTO) and the CTO of Motorola concluded that current memory cards were too large for mobiles Phones.

2.2 Software Architecture:

Libraries to download in Python

1. Pillow (PIL)

   Python pillow library is used to image class within it to show the image. The image modules that belong to the pillow package have a few inbuilt functions such as load images or create new images, etc.

2. numpy

   Live video is captured of the user and is trained by the model and further is used for the recognition. In all of these processes, image processing plays a vital role. An image is nothing but an array of integers and to tweak those arrays, we have used the numerical python (numpy) library.

3. cv2

   We've used OpenCV for capturing the live video of the intruder, converting the RGB image into grayscale. We found the use of cv2 with its built in Cascade Classifier that integrates with Haar Cascade Frontal Face Features to recognize the human faces.

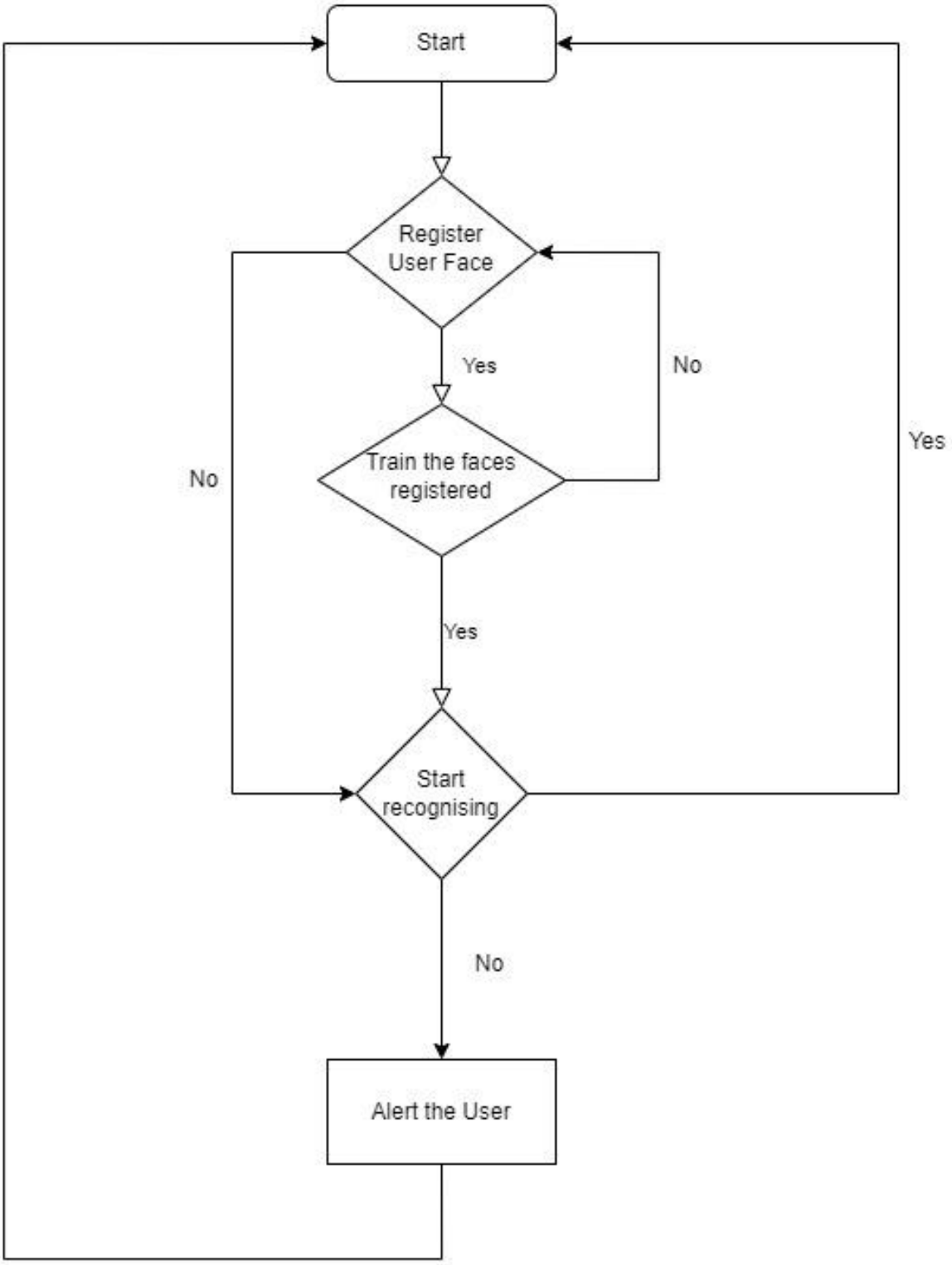
4. os

   The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. You first need to import the os module to interact with the underlying operating system.

5. telepot

   We've used the telepot library in order to build applications for the Telegram BOT API. Since Telegram is a handy application for messaging, it is convenient for the user to check the received messages on the Telegram app. So, we imported the telepot library.

2.3 Flowchart



2.4 Code

2.4.1 Code for face Recognition

The code consists of three different as follows:

1) 01_face_dataset.py
2) 02_face_training.py
3) 03_face_recognition.py

Part I

```
import cv2
import os
cam = cv2.VideoCapture("http:192.162.21.234:8080/video") # IP address of IP webcam application
cam.set(3, 640) # set video width
cam.set(4, 480) # set video height
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# For each person, enter one numeric face id
face_id = input('\n Enter user ID end press <Enter> ==>  ')
print("\n Initializing face capture. Look at the camera and wait ...")
# Initialize individual sampling face count
count = 0
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1
        # Save the captured image into the datasets folder
                cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
        cv2.imshow('image', img)
    k = cv2.waitKey(100) & 0xff # Press 'ESC' for exiting video
    if k == 27:
        break
```

```
    elif count >= 30: # Take 30 face sample and stop video
        break
# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

Part II

```
import cv2
import numpy as np
from PIL import Image
import os
# Path for face image database
path = '/home/pi/project/facial_recognition/dataset'

os.chdir("/home/pi/project/facial_recognition/trainer")
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector =
cv2.CascadeClassifier("/home/pi/project/facial_recognition/haarcascade_frontalface_
default.xml");
# function to get the images and label data
def getImagesAndLabels(path):
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
        for (x,y,w,h) in faces:
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return faceSamples,ids
```

```
print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
# Save the model into trainer/trainer.yml
recognizer.write('/home/pi/project/facial_recognition/trainer/trainer.yml')
# Print the numer of faces trained and end program
print("\n [INFO] {0} faces trained. Exiting Program".format(len(np.unique(ids))))
```

Part III

```
import cv2
import numpy as np
import os

os.chdir("/home/pi/project/facial_recognition/trainer")
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('/home/pi/project/facial_recognition/trainer/trainer.yml')
cascadePath="/home/pi/project/facial_recognition/haarcascade_frontalface_default.x
ml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_SIMPLEX

#indicate id counter
id = 0

# names related to ids: example ==> KUNAL: id=1,  etc
names = ['None', 'Adi', 'Sahil']

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height
```

```
# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

while True:
    ret, img =cam.read()
    #img = cv2.flip(img, -1) # Flip vertically
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor = 1.2,
        minNeighbors = 5,
        minSize = (int(minW), int(minH)),
       )

    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

        # Check if confidence is less them 100 ==> "0" is perfect match
        if (confidence < 100):
            id = names[id]
            confidence = "  {0}%".format(round(100 - confidence))
        else:
            id = "unknown"
            confidence = "  {0}%".format(round(100 - confidence))

        cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
        cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)

    cv2.imshow('camera',img)

    k = cv2.waitKey(10) & 0xff # Press 'ESC' for exiting video
```

```
    if k == 27:
        break

# Do a bit of cleanup
print("\n [INFO] Exiting Program and cleanup stuff")
cam.release()
cv2.destroyAllWindows()
```

2.4.2 Code for messaging via Telegram

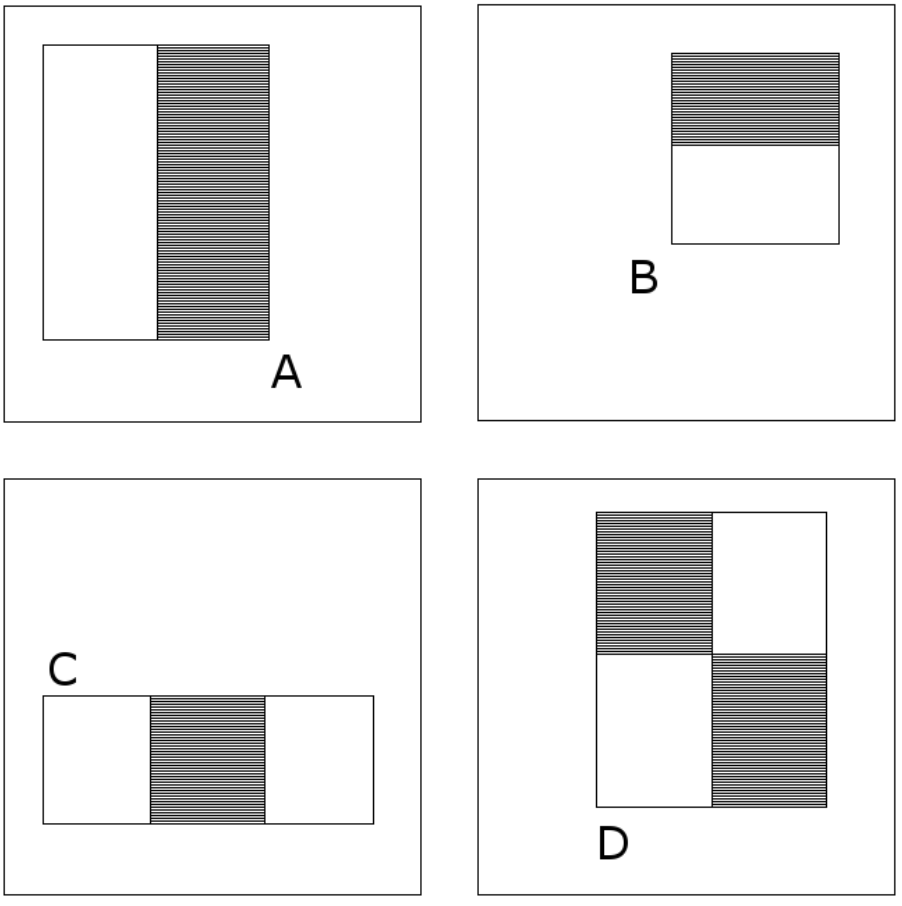
```
import telepot
def handle(msg):
  global telegramText
  global chat_id
  global receiveTelegramMessage

  chat_id = msg['chat']['id']
  telegramText = msg['text']

  print("Message received from " + str(chat_id))

  if telegramText == "/start":
    bot.sendMessage(chat_id, "Welcome to Idris Bot")

  else:
    receiveTelegramMessage = True

def capture():
    print("Capturing photo…")
    camera = cv2.VideoCapture()
    print("Sending photo to " + str(chat_id))
    bot.sendPhoto(chat_id, photo = open('./photo.jpg', 'rb'))

bot = telepot.Bot('5200163941:AAGWKlzIlBWZIPbuOPPtalJI5NxAICOq00Q')
bot.message_loop(handle)
statusText = ""

print("Telegram bot is ready")
```

2.4.3 Algorithm working

Face Detection:

The first task that we perform is detecting faces in the image(photograph) or video stream. When a face is detected, the exact coordinates/location of the face, we extract the face for further processing. The detection of a face is backed by the Haar Cascade Frontal Face Features. Since Haar Cascade Frontal Face Features are used for face detection, the image is converted into a black-and-white picture as the pixels are then arranged from the range of black and white. All human features exhibit similar patterns and these similar patterns are classified as Haar Features. These patterns are observed in a grayscale image, expressing patterns in black and white.

Haar-Cascade Features



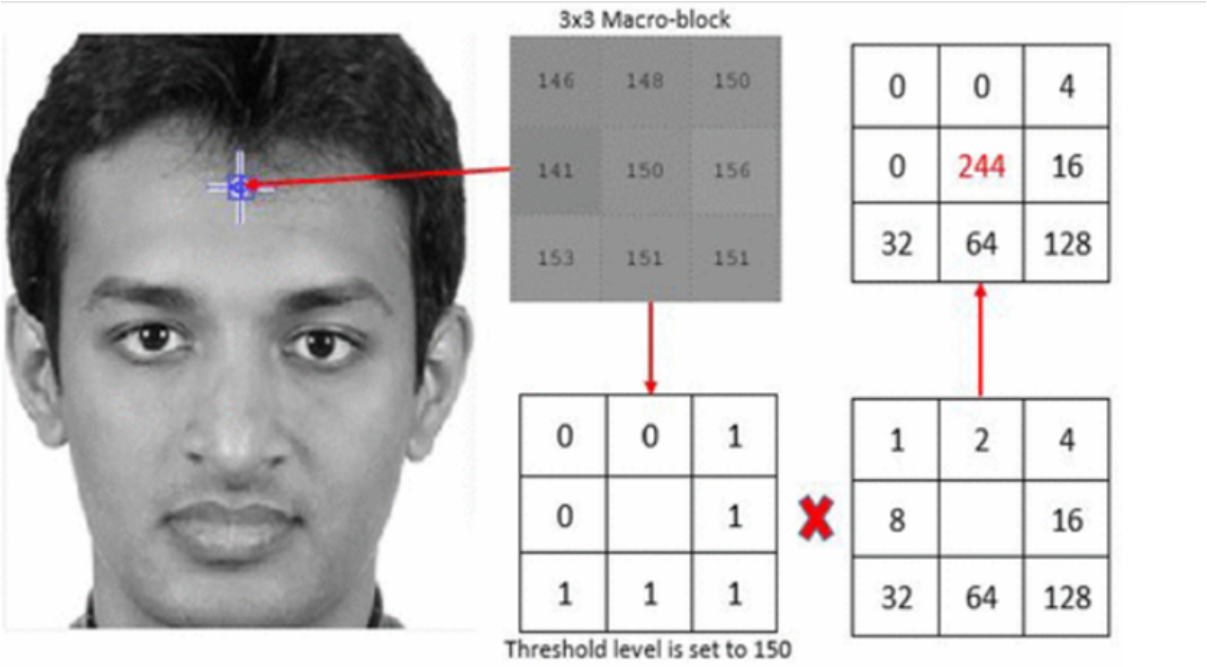
Feature extraction:

Once a face is detected and cropped out according to the coordinates of the rectangle, face embeddings are used to extract the features out of the face. A person's face is taken as an input by a neural network and output representing the most important features of face are expressed as an output.

Once the face embeddings are collected, the neural network learns to output similar vectors that look alike or similar.

LBPH Face Recognition:

Since its discovery, LBPH has been an excellent feature for the classification of certain textures like faces. It requires four distinct parameters to process an image, they are radius (*r*), neighbours (*n*), X-axis and Y-axis.Here X and Y-axis represents the dimensionality of the features grid in vertical and horizontal manner. The first step is to train the algorithm, and to do so, it is necessary to use the correct dataset with facial images of the people that we need to identify. For the computational step, it is imperative to transform an image of a person into a set of 3×3 macro-block for better representation.By doing this, it is possible to pinpoint each and every feature that exists on a person's face. Each macro-block has 9 pixels and they have the range of 0 to 255 as they are of grayscale format.



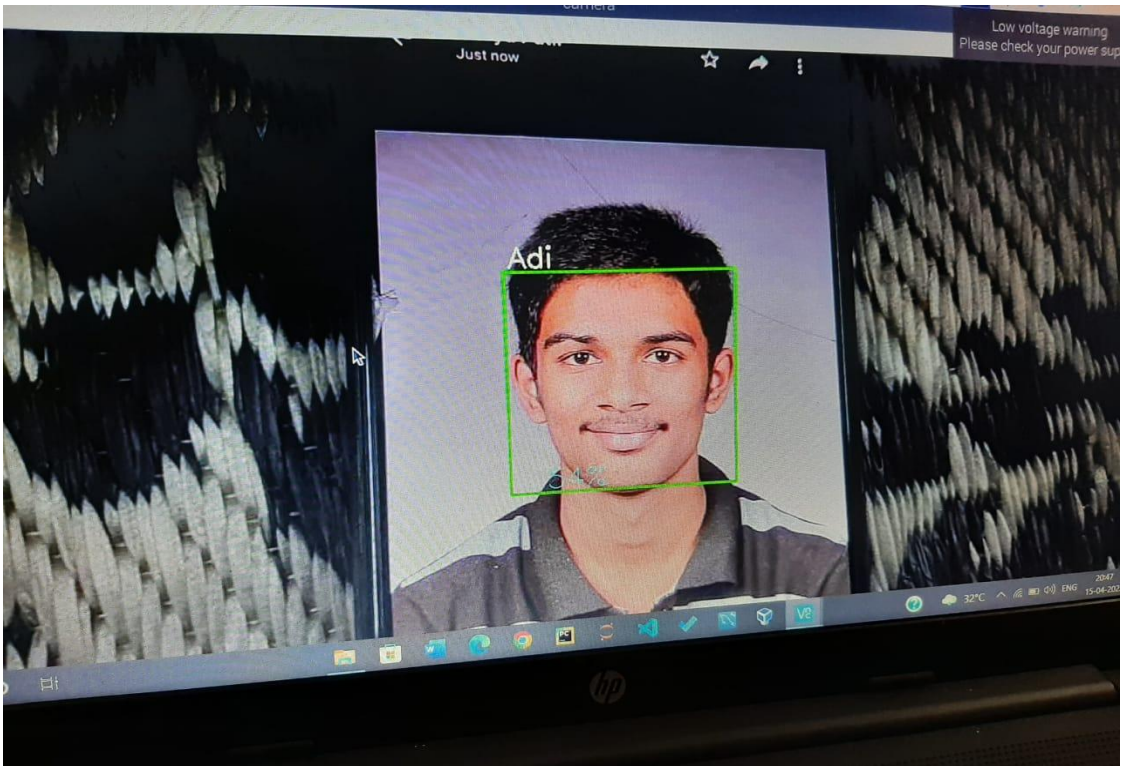
The next step is to convert the macro-block into a binary pattern. The value of the centre pixel is a threshold for the rest of its neighbours. Any value that is equal to or greater than

150 is set to 1 and if they are less, then they are set to 0. The algorithm concatenates this binary information and places it in a clockwise pattern. This process is actually known as Linear Binary Pattern (LPB).

3. Performance Evaluation and Result Analysis

Initial Setup:

Output:



Result Analysis:

As we can see the confidence level of the detection is present below in the image. The region with the higher confidence score should be more likely to contain a face. If a face detection system does not properly detect a face, or provides a low confidence prediction of an actual face, this is known as a missed detection or false negative. If a facial detection system incorrectly predicts the presence of a face at a high confidence level, this is a false alarm or false positive. Similarly, a facial comparison system may not match two faces belonging to the same person (missed detection/false negative), or may incorrectly predict that two faces from different people are the same person (false alarm/false positive).

For use cases where the risk of missed detection or false alarm is higher, the system should use a higher confidence level. You should use a 99% confidence/similarity threshold in scenarios where highly accurate facial matches are important.

Here the threshold is set to 50% for testing purposes.

4. Bill of Materials:

Components	Price
1) Raspberry Pi 3B+	Rs 3500
2) Mobile charger/Power cable	Rs 100
3) SD card(8 GB)	Rs 375
4) Heat sink	Rs 100
Total Cost	Rs 4075

5. Conclusion and future Scope

Face detection and face recognition are two areas of intensive research, because they enable better interaction between computer systems or robots on one side and humans on the other.

Python's face recognition library turns out tobe fast and very reliable tool for face detec-tion and face recognition. As Python is a high level programming language the library is well suited to be used just as a face detection(recognition) procedure in a wider project without the need for a detailed knowledge of the theoretical background of the employed algorithms. So, in our opinion it has a bright future.

In the future it would be very interesting to investigate the possibilities of Python and its libraries for emotion detection. This field is a very hot topic in human machine interface research. Using the results of this research it is possible to vastly improve the social aspects of robots or software packages that can adapt to the user.It namely gives a very reliable feedback in human machine interaction.

6. References

- Automated Facial Recognition System - Wikipedia
- Google Images
- YouTube
- Opensource website
- Setting Up a VNC Server on Your Raspberry Pi : 4 Steps - Instructables
- (PDF) Mask Detection and Social Distance Identification Using Internet of Things and Faster R-CNN Algorithm (researchgate.net)
- Overview of face detection and face comparison - Amazon Rekognition
- Sage Journals
- Raspberry-Pi-Model-Bplus-Product-Brief.pdf (raspberrypi.org)