The National Institute of Engineering

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi)

Mysuru-570008

ESTD : 1946

DISSERTATION
ON

"HOME SURVEILLANCE SYSTEM"

Submitted in partial fulfillment for the award
of

BACHELOR OF ENGINEERING
IN
"ELECTRONICS AND COMMUNICATION"
For the academic year 2016-2017

Submitted by:

ADARSH PRABHAKAR CHANTAR	**4NI13EC002**
CHARAN G BHAKTHA	**4NI13EC012**
MANJUNATH S G	**4NI13EC026**
MANOJKUMAR H M	**4NI13EC027**

Under the guidance of

Narasimha Kaulgud
Professor, Dept. of ECE

Carve Create Conquer

THE NATIONAL INSTITUTE OF ENGINEERING

Department Of Electronics and Communication Engineering
Mysuru- 570008.

2016-2017

THE NATIONAL INSTITUTE OF ENGINEERING

Department of Electronics and Communication

Mysuru -570008.

CERTIFICATE

This is to certify that the project entitled **"HOME SURVEILANCE SYSTEM"** has been successfully completed by **ADARSH PRABHAKAR CHANTAR (4NI13EC002), CHARAN G BHAKTHA (4NI13EC012), MANJUNATH S G (4NI13EC026), MANOJKUMAR H M (4NI13EC027)** of 8th semester B.E. who carried out the project work under guidance of "**DR.NARASIMHA KAULGUD"** in the Partial fulfillment for the award of degree of Bachelor Engineering in Electronics and Communication Engineering of Visvesvaraya Technological University, Belagavi during the year 2016-17. It is certified that all corrections/Suggestions indicated during internal assessment have been incorporated in the report. The Project has been approved in partial fulfillment for the award of the said degree as per academic regulations of the National Institute of Engineering (An Autonomous Institution under Visvesvaraya Technological University, Belagavi).

Dr.Narasimha Kaulgud, Ph.D.,	**Dr.Rohini Nagapadma,** Ph.D.,	
Project Guide,	Head of the Department,	Principal,
Department of ECE,	Department of ECE.	NIE, Mysuru.

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

We would take this opportunity to express our sincere gratitude and respect to lot of eminent personalities. Without whose constant encouragement and gratitude, the project of ours wouldn't become reality.

We are grateful to **Dr. Rohini Nagapadma**, HOD, Department of Electronics and Communication Engineering, NIE Mysore for her support facilitating the progress of this work.

We sincerely extend our thanks to our guide **Dr.Narasimha Kaulgud**, Professor, Department of Electronics and Communication Engineering, NIE Mysore for their valuable guidance, constant assistance and support for the betterment of this project.

Our hearty thanks to our friends, non-teaching staff and all those other people who have supported us during the project.

ADARSH PRABHAKAR CHANTAR

CHARAN G BHAKTHA

MANJUNATH S G

MANOJKUMAR H M

CONTENTS

ABSTRACT

Home surveillance systems are widely used now a day. An intelligent surveillance system can provide multiple functions to the users. The focus of the home surveillance systems is to monitor the home with the wireless systems along with the help of the camera with the lowest possible cost of investment.

The surveillance system sends notification to the homeowner in case of any movements inside the home along with the frame in which movement is being captured to the home owner.

The surveillance system must also be able to distinguish between the family members and other people near the door. To distinguish between the family members and the intruders, for that it must be able to detect the face recognize it. The detection and the recognition process is done inside the processor and if the face is not recognized then it must send notification to the home owner along with the captured frame of the non-family member.

LIST OF FIGURES

LIST OF ACRONYMS

- ARM – Advanced RISC Machine
- ASCII – American Standard Code for Information Interchange
- CCTV – Closed Circuit Television Cameras
- CRT- Cathode Ray Tube
- CSI – Camera Serial Interface
- DSI – Discrete Stream Interface
- ESMTP –Extended Simple Mail Transfer Protocol
- GCC – GNU Compiler Collection
- Gmail – Google mail
- GPIO Pins – General Purpose Input Output Pins
- GUI – Graphical User Interface
- HDMI – High Definition Multimedia Interface
- IMAP – Internet Message Access Protocol
- LBPH – Local Binary Patterns Histograms
- LCD –Liquid Crystal Display
- MIME – Multi-purpose Internet Mail Extensions
- MTA – Mail Transfer Agent
- NVT ASCII – Network Virtual Terminal ASCII
- Open CV – Open Source Computer Vison
- POP – Post Office Protocol
- RISC – Reduced Instruction Set Computer
- SD – Secure Digital
- SMTP – Simple Mail Transfer Protocol
- TLS – Transport Layer Security
- USB –Universal Serial Bus
- XML – Extensible Markup Language

1. INTRODUCTION

1.1. OVERVIEW

Surveillance is monitoring of activities, or other changing information for the purpose of influencing, managing, directing, or protecting people[1]. This can include observation from a distance by means of electronic equipment (such as closed-circuit television (CCTV) cameras) or interception of electronically transmitted information (such as Internet traffic or phone calls). It can include relatively low-technology methods such as human intelligence agents and postal interception. The word *surveillance* comes from a French phrase for "watching over" (*sur* means "from above" and *veiller* means "to watch")

Today many home owners have a home surveillance system. Traditionally these systems have been built in an *ad hoc* fashion with direct wired connections between the control center and the sensors. This is changing due to the use of local area network technology for the interconnections (be they wired or wireless) and the fact that the controlling system is increasingly connected to the internet.

The connection to the internet enables home owners (and potentially others) to access information collected by the home security and monitoring system from any place.

Some vacation houses are located in rural areas, and they might be used only during vacation time or the house is locked whole day due to their work and used only during night. Home owners cannot easily check on the condition of their home every day. Therefore, surveillance is needed to inform the homeowner if the house has been broken into. The services that can be provided by such a system can also be very convenient for families with children. When the children play in different rooms, using this surveillance system the parent(s) can easily monitor the movement of the children within the house.

The home surveillance system also includes face detection and recognition process, face detection includes detecting the human face and recognition includes matching the detected face with the face of the family members and comes to the conclusion whether they are family members or not.

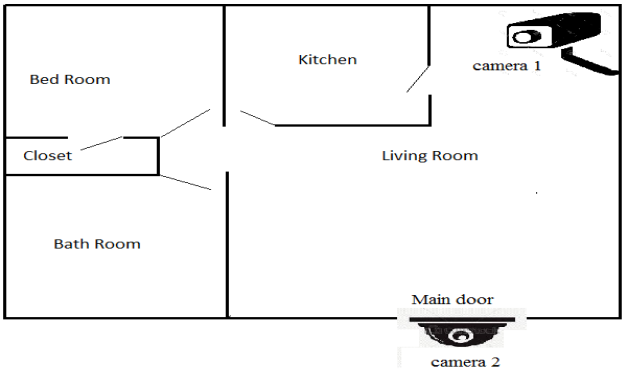


Fig 1.1: Home plan of Surveillance

The process of face detection helps the home owner in the following manner, the owner need not open the door every time to see who are all coming near the door. If the person coming near the door is a non-family member, intruder alert message is sent to the owner through the email along with the captured intruder face.

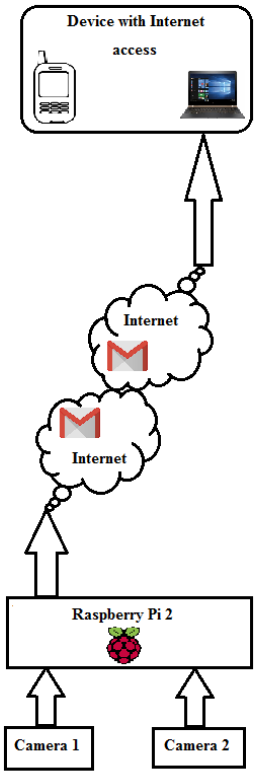


Fig 1.2: Block diagram of Home Surveillance System

1.2. TOOL SET

1.2.1. Camera

A camera is an optical instrument for recording or capturing images, which may be stored locally, transmitted to another location, or both. The images may be individual still photographs or sequences of images constituting videos or movies. The camera is a remote sensing device as it senses subjects without physical contact.

Fig 1.3: Logitech c170 USB Camera

Camera used in our work is **Logitech c170** whose specifications are

- Video capture: Up to 1024 x 768 pixels
- Logitech Fluid Crystal™ Technology*
- Photos: Up to 5 megapixels (software enhanced)
- Built-in mic with noise reduction
- Hi-Speed USB 2.0 certified (recommended)
- Universal clip fits laptops, LCD or CRT monitors

The home surveillance system uses two cameras say camera 1 and camera 2. The camera 1 is placed inside the home for the movement detection and camera 2 is placed near the door for the face detection and recognition process.

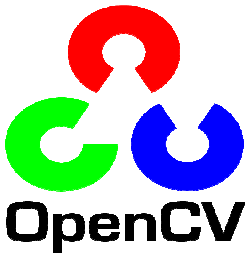
1.2.2. [1]Python

It is one of the widely used high-level programming language, created by Guido van Rossum and first released in 1991.

An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. More details on python are available in[2].

Python is used as our programming language since it is easy to learn and large community is available where we can clear our any doubts regarding the syntax or any other problem.

1.2.3. Open CV



This is an open source computer vision and machine learning software library. OpenCV is free for both academic and commercial use since it is released under a BSD license.

It has C[2], C++, Java and Python interfaces. Windows, Linux, Mac OS, iOS and Android platforms are supported. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. It is written in optimized C/C++. The library can take advantage of multi-core processing. More details on OpenCV are available in[3].

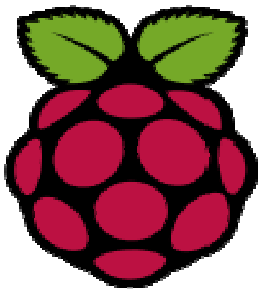
We have used OpenCV since it is open source and consists of vast optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements,

[1] Copyright of all the logos belongs to respective organizations

[2] Recent versions of OpenCV have stopped supporting C

track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, *etc*. These libraries are supported for python programming language.

1.2.4. Raspberry Pi



It is the main part of the home surveillance system. The Raspberry Pi 2 Model B is the second generation Raspberry Pi.

Compared to the Raspberry Pi 1 it has:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM

Like the (Pi 1) Model B+, it also has:

- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- Video Core IV 3D graphics core

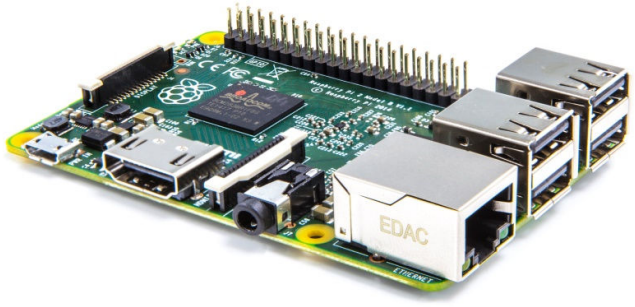


Fig 1.4: Raspberry Pi 2 Model B

Because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, as well as Microsoft Windows 10.The

Raspberry Pi 2 has an identical form factor to the previous (Pi 1) Model B+ and has complete compatibility with Raspberry Pi 1. More features of RaspberryPi are available in[4].

Raspberry is used because it has its own OS Raspbian and it supports python, OpenCV and Logitecth webcams.

1.2.5. Gmail

It is a free, advertising-supported email service developed by Google. Users can access Gmail on the web and through the mobile apps for Android and iOS, as well as through third-party programs that synchronize email content through POP or IMAP protocols.

Gmail at launch had an initial storage capacity offer of 1 gigabyte per user, a significantly higher number than the 2 megabytes of storage its competitors such as Hotmail offered at that time. Today, the service comes with 15 gigabytes of free storage. Users can receive emails up to 50 megabytes in size, including attachments, while they can send emails up to 25 megabytes. In order to send larger files, users can insert files from Google Drive into the message.

1.3. BASICS OF DIGITAL IMAGE

1.3.1. Digital Image

A digital image is a numeric representation of (normally binary) a two-dimensional image. It is a 2D discrete signal or an N x N array of elements. Each element in a array is a number which represents the sampled intensity.

A digital image consists of picture elements called pixels. Pixels are the smallest sample of an image. A pixel represents the brightness at one point.

1.3.2. Types of Images

1.3.2.1. Binary Image

It is also called as Black-and-White image. Binary images take only two values, *i.e.*, either '0' or '1'. The brightness gradient cannot be differentiated in binary image. A gray-scale image can be converted into binary image by thresholding. Geometric properties of an object such as location, centroid or orientation of the object can be easily extracted from binary image.

Fig 1.5: Example for Binary image

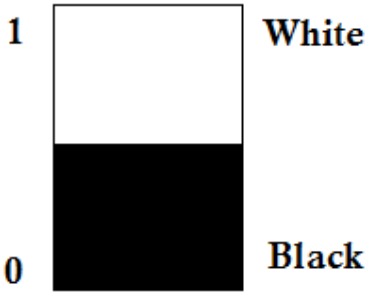


Fig 1.6: Binary image intensity levels

1.3.2.2. Gray-scale Image

Gray-scale image contain only brightness information. Each pixel value in a gray-scale image corresponds to an amount or quality of light. The brightness graduation can be differentiated in gray-scale image. In a gray-scale image, each pixel is represented by a byte or word, the value of which represent the light intensity at that point in the image. An 8-bit image will have a brightness variation from 0 to 255 where '0' represents black and '1' represents white.

Fig 1.7: Example for gray-scale image

Color intensity	DEC	HEX	BIN
	0	0x00	00000000
	16	0x10	00010000
	32	0x20	00100000
	48	0x30	00110000
	64	0x40	01000000
	80	0x50	01010000
	96	0x60	01100000
	112	0x70	01110000
	128	0x80	10000000
	144	0x90	10010000
	160	0xA0	10100000
	176	0xB0	10110000
	192	0xC0	11000000
	208	0xD0	11010000
	224	0xE0	11100000
	255	0xFF	11111111

Fig 1.8: Gray-scale image intensity levels

1.3.2.3. Colour Image

A colour image has three values per pixel and they measure the intensity and chrominance of light. Each pixel is a vector of colour components. Common colour spaces are RGB(Red, Green, Blue), HSV (hue, Saturation, Value), and CMYK (Cyan, Magenta, Yellow, Black). Uncompressed data-rate of a colour image is three times that of gray-scale image.

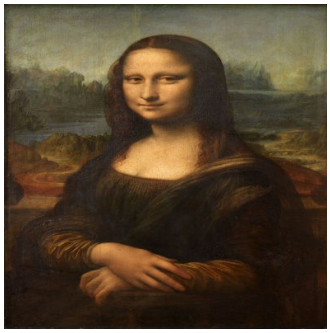


Fig 1.9: Example for colour image

1.3.3. Morphological Operations on Images

Morphological Operations set of operations that process images based on shapes. Morphological operations apply a structuring element to an input image and generate an output image. The most basic morphological operations are two: Erosion and Dilation. They have a wide array of uses, *i.e.,* :

- Removing noise
- Isolation of individual elements and joining disparate elements in an image.
- Finding of intensity bumps or holes in an image

1.3.3.1. Erosion

The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). So what it does? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

All the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises, detach two connected objects *etc*.

1.3.3.2. Dilation

In Dilation, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.

1.3.4. Image Smoothing Techniques

Smoothing, also called blurring, is a simple and frequently used image processing operation. There are many reasons for smoothing. Removal of noise is one of them.

Smoothing operation can be done by applying a *filter* to our image. Most common filters are Normalized Box Filter, Gaussian Filter, Median Filter.

1.3.4.1. Normalized Box Filter

This filter is the simplest of all. Each output pixel is the *mean* of its kernel neighbors (all of them contribute with equal weights)

1.3.4.2. Gaussian Filter

Gaussian filter is probably the most useful filter (although not the fastest). Gaussian filtering is done by convolving each point in the input array with a *Gaussian kernel* and then summing them all to produce the output array.

1.3.4.3. Median Filter

The median filter run through each element of the signal (in this case the image) and replace each pixel with the median of its neighboring pixels (located in a square neighborhood around the evaluated pixel).

2. MOTION DETECTION

2.1. INTRODUCTION

Motion detection is the process of detecting a change in the position of an object relative to its surroundings or a change in the surroundings relative to an object. Motion detection can be achieved by either mechanical or electronic methods. When motion detection is accomplished by natural organisms, it is called motion perception.

Motion can be detected by:

- Infrared (passive and active sensors)
- Optics (video and camera systems)
- Radio Frequency Energy (radar, microwave and tomographic motion detection)
- Sound (microphones and acoustic sensors)
- Vibration (triboelectric, seismic, and inertia-switch sensors)
- Magnetism (magnetic sensors and magnetometers)

The method we have implemented in our work is optics (Video and camera systems)

2.1.1. Terminologies used

Base frame: Frame which is taken initially during the setup is known as Base frame. Base Frame is different for different camera locations. This frame is stored in the database.

Current frame: Frame which is taken at that instant of time is known as Current frame. Current frame changes at each and every instant. Current frame is stored only if the motion inside the home is detected, if the movement is not found it will be simply discarded.

2.2. IMPLEMENTATION

This phase includes detection of movements inside the home. The camera is placed inside the home in all the rooms where the motion has to be detected. The camera continuously captures the frame inside the home.

Camera placed inside the home, camera 1 continuously monitors the room. As soon as the Raspberry pi is turned ON, the base frame or the first frame is initialized to none. Later the frame captured by the camera 1 is stored in the current frame. Current frame is converted to gray-scale for better comparison. If Base frame is None then the gray scale current frame is

stored in the base frame. This is done only during the first iteration. The absolute difference between the current frame and base frame is computed. If there is any difference between the base frame and the current frame then the current frame is temporarily stored so that the image can be used to send notification to the home owner else the current frame is ignored and goes to the next frame. Again Current frame is updated and the previous frame is stored in the base frame. Here the Current frame is compared with previous frame (base frame).

All the computations are done in Raspberry Pi and the codes are written in Python language and making use of some functions of OpenCV. When the movement is detected the notification has to be sent through the email to the owner.

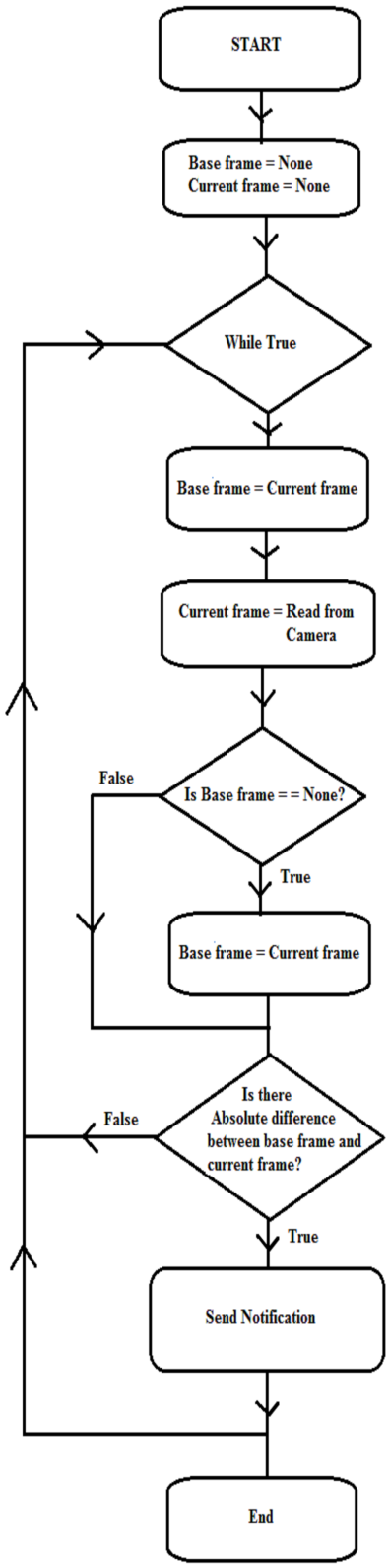


Fig 2.1 : Flowchart for Motion detection

2.3. WORKING

(a) Import required libraries and modules: We are using SMTP protocol for sending mail camera module to capture video and we are using system time and date for timestamp. In order to use the above features we need to import the required libraries.

(b) Defines required parameters such as from address, to address, password, text *etc.,* to send email to the user: In this step we defines the Gmail address of the system and the user along with the password of the system Gmail account from which we are sending the notification. We also define the text which should be in the body of the email.

(c) Start the SMTP Gmail connection:

Using module "`em.startserver(from_address,password)`" we login to the Gmail account. Before logging in, we put the SMTP connection in TLS mode. All the commands that follow will be encrypted.

(d) Create an camera object for `cv2.VideoCapture()` class: OpenCV provides a very simple interface to capture live stream with camera. In order to capture the video, the **Video Capture** object camera is created here. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Only one camera that is Camera1 is used for motion detection. Hence we have passed 0 (or -1). We can select the second camera by passing 1 and so on. After that, the camera starts capturing the images by frame-by-frame. It is also possible to create `cv2.VideoCapture(filename)` object. Filename is the name of the opened video file (Eg. video.avi) or image sequence (Eg., img_%02d.jpg, which will read samples like img_00.jpg, img_01.jpg, img_02.jpg.)

(e) Take a variable '`gray`', set it to None: The first frame has to be initialized with NULL. Because while executing this command the program is running whereas camera is not running or the camera is not yet started here. Hence it is initialized None easily in the first iteration of the while loop.

(f) Set the base frame to `gray`: The gray image is assigned to '`baseFrame`' in every loop. It will be 'None' in the first iteration. In all iterations that follow, gray-scale converted image of the previous frame will be assigned. Every image is converted to

gray scale for easy comparison of the images in the matrix form, rather than to compare two Captured images directly.

(g) Capture the current frame from camera object using read function: `cv2.VideoCapture.read()` usually returns two values. 0^{th} value is the Boolean value that gives true if a frame is grabbed and false if camera has been disconnected, or there are no more frames in video file. 1st frame decodes and returns the grabbed video frame, and this is the one that we need.

(h) Resize the captured frame by maintaining aspect ratio using resize function of `imutils` library: This resize function of `imutils` maintains the aspect ratio and provides the keyword arguments width and height so the image can be resized to the intended width/height while

(1) maintaining aspect ratio and

(2) ensuring the dimensions of the image do not have to be explicitly computed by the developer.

(i) Convert captured frame to the gray scale image and store it in variable 'gray': In OpenCV, for colour conversion, we use the function `cv2.cvtColor(input_image, flag)` where flag determines the type of conversion.

(j) Remove the Guassian noise from the image and store it in variable `gray`**:** In image processing, a Gaussian blur (also known as Gaussian smoothing) is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise and reduce detail.

(k) If base frame is `None` , assign `gray` to base frame. Initialize the base-Frame to the current-Frame in first iteration. In all remaining iterations, this loop is not at all executed.

(l) Calculate the per-element absolute difference between base frame and gray. Assign it to '`frameDelta`': Calculates the per-element absolute difference between two arrays or between an array and a scalar.

(m) Increase the white region in the image or size of foreground object using dilation and assign it to '`thresh`': Here, a pixel element is '1' if at least one pixel under the kernel

is '1'. So it increases the white region in the image or size of foreground object increases. It is also useful in joining broken parts of an object.

(n) Find Contours in thresh: Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. In simple words, a contour refers to the outline or silhouette of an object. The contours are a useful tool for shape analysis and object detection and recognition.

(o) Compute the bounding box for the contour, draw it on the frame, and update the text: `cv2.boundingRect()` is a straight rectangle, it doesn't consider the rotation of the object. So area of the bounding rectangle won't be minimum. `cv2.rectangle` draws a simple, thick, or filled up-right rectangle.

(p) Print the timestamp and Room status on the frame using `putText` function: The function `putText` renders the specified text string in the image. Symbols that cannot be rendered using the specified font are replaced by question marks.

(q) Display security feed: Using `cv2.imshow` function display the security feed.

(r) Send notification to the user if motion is detected: Send the email to the user using the '`sendmail`' module written in `emailattach` program. It takes following arguments:

`fa`: Sender's Mail Address(*i.e.,* of our System)

`ta`: Receivers Mail Address (*i.e.,* of owner)

`pw`: Gmail Password of Sender

`path`: Path of the image stored when motion is detected

`fn`: Name of the image file

`textm`: Message to be printed along with the image notification

(s) Jump to step v if Escape key is pressed.

(t) Jump to step f.

(u) Destroy all created windows and release the camera. Also end the SMTP Gmail connection: `em.exitserver()` Logout from Gmail server using the '`exitserver`' module written in `emailattach` program. `Camera.release()` closes video file or capturing device of the camera object of cv2.VideoCapture

created. `cv2.destroyAllWindows()` function destroys all of the opened High GUI windows.

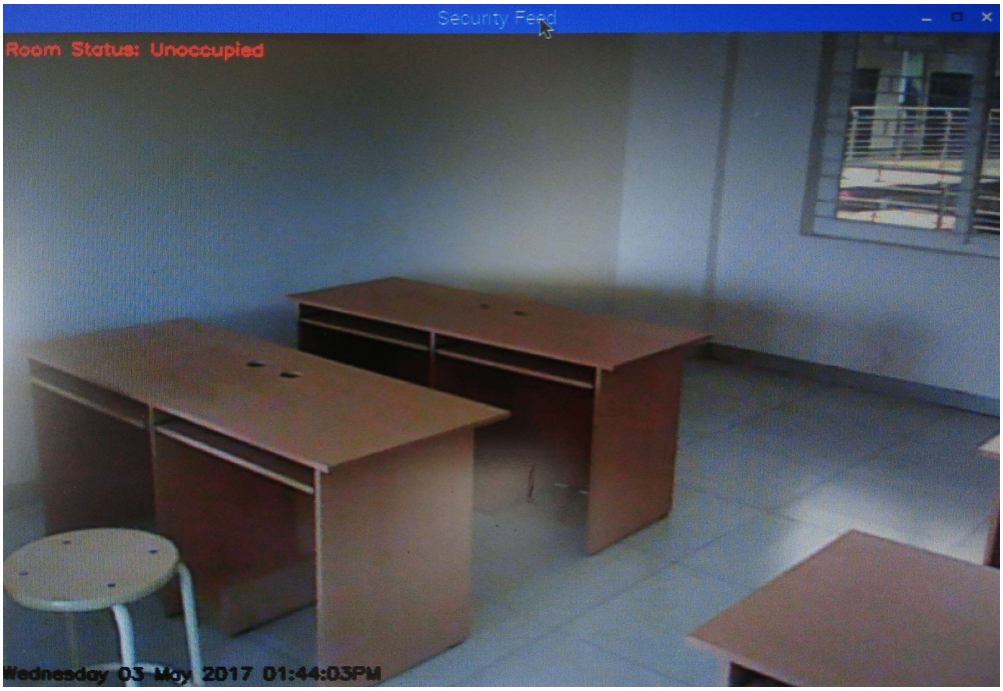


Fig 2.2 : Image of the room when there is no motion

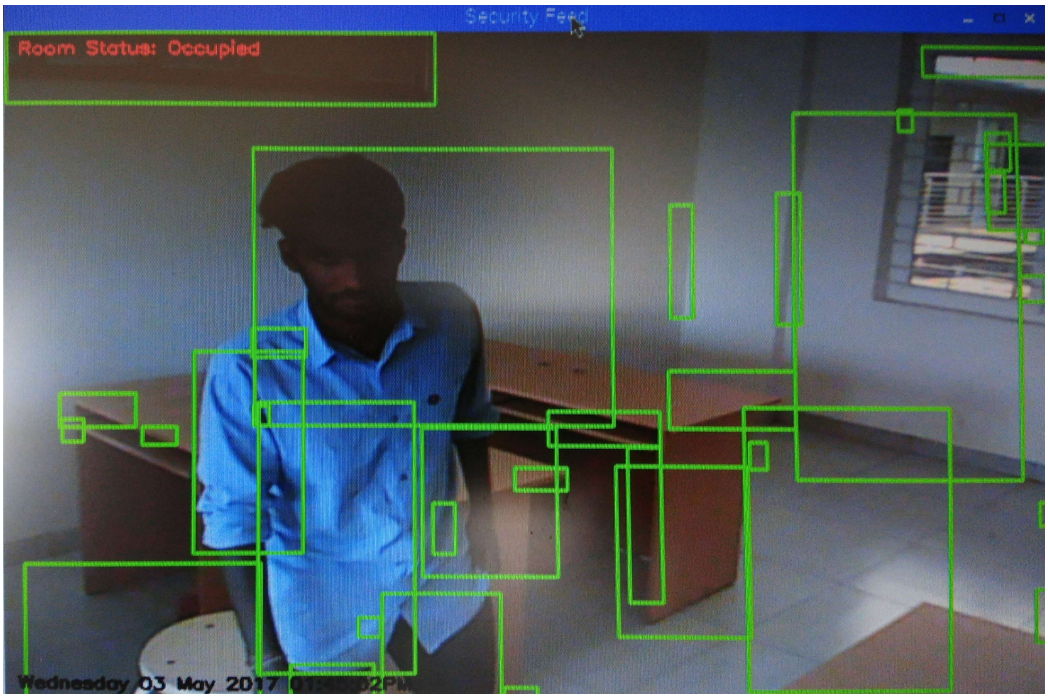


Fig 2.3 : Image of the room when motion is detected

3. FACE DETECTION AND RECOGNITION

3.1. INTRODUCTION

A **face recognition system** is a computer application capable of identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a face database.

It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems. Recently, it has also become popular as a commercial identification and marketing tool.

3.2. IMPLEMENTATION

3.2.1. Overview

The camera 2 placed near the door is used for face detection and recognition process. As shown in the flowchart face detection camera runs continuously in a loop and it comes out of the loop only when the face is not recognized, the non-recognized face is considered as a intruder and the face of the intruder is captured and stored.

The whole face recognition phase can be divided in to 3 steps.

i. Building an image database of the members of the house.

ii. Detecting the faces in the database and use it to train the face recognizer.

iii. Detect faces in the captured frame and compare it with the trained database.

Database of the images should be taken with different lighting and different expressions of the members else even the members of the house may be identified as intruders. All the images are stored in a folder.

OpenCV contains many pre-trained classifiers for object detection which is based on Haar-Cascade based classifier. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper[5]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

OpenCV already contains many pre-trained classifiers for face, eyes, smile *etc.,* and are stored in XML files. By using those XML files, all the faces in image database is

detected, converted to gray-scale and saved. The camera will be looking for a face continuously. If a face is detected, that frame is compared with the trained database and if a matching frame is not found in the database, the current frame is captured and it is sent as a notification through the email to the owner.

3.2.2. Haar Cascade Classification

OpenCv mainly provides two cascade classifiers: Haar-based and LBP based classifier.

The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, *etc*.). Follow the pixels along a circle, *i.e.,* clockwise or counter-clockwise.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the frequency of each "number" occurring (*i.e.*, each combination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
- Optionally normalize the histogram.
- Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

We use Haar based classifier since is accurate than LBP. Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper[5]. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, haar features shown in below image are

used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

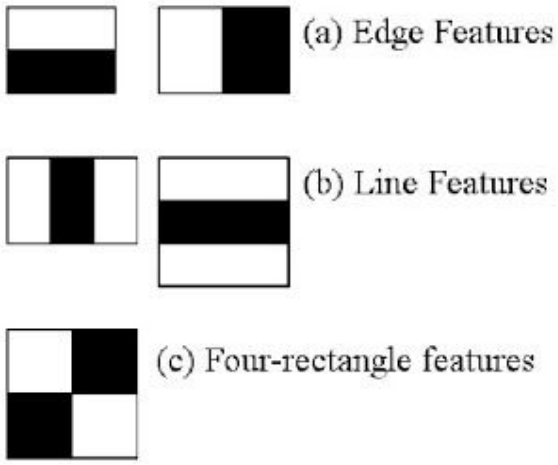


Fig 3.1: Features of Haar cascade

Now all possible sizes and locations of each kernel is used to calculate plenty of features. (Lot of computation needs to be done. Even a 24x24 window results over 160000 features). For each feature calculation, we need to find sum of pixels under white and black rectangles. To solve this, Paul Viola and Michael Jones introduced the integral images. It simplifies calculation of sum of pixels, how large may be the number of pixels, to an operation involving just four pixels. It makes things super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant. We select the best features out of 160000+ features using Adaboost.

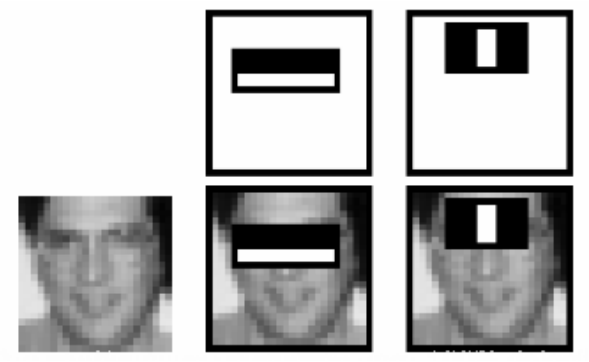


Fig 3.2: Applying line and edge features on a image

For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. But obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that best classifies the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then again same process is done. New error rates are calculated. Also new weights. The process is continued until required accuracy or error rate is achieved or required number of features are found).

Final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier. The paper[5] says even 200 features provide detection with 95% accuracy. Their final setup had around 6000 features. (A reduction from 160000+ features to 6000 features is a big gain).

So now we can take an image. Take each 24x24 window. Apply 6000 features to it. Check if it is face or not. It is little inefficient and time consuming.

In an image, most of the image region is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot. Don't process it again. Instead focus on region where there can be a face. This way, we can find more time to check a possible face region.

For this Paul Viola and Michael Jones introduced the concept of Cascade of Classifiers. Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one. (Normally first few stages will contain very less number of features). If a window fails the first stage, discard it. We don't consider

remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.

Detector proposed in[5] had 6000+ features with 38 stages with 1, 10, 25, 25 and 50 features in first five stages. (Two features in the above image is actually obtained as the best two features from Adaboost). According to authors of[5], on an average, 10 features out of 6000+ are evaluated per sub-window.

OpenCV comes with a trainer as well as detector. If we want to train our own classifier for any object like car, planes *etc*., we can use OpenCV to create one. Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smile *etc*. Those XML files are usually stored in opencv/data/haarcascades/ folder.

3.2.3. LBPH Face Recognition

3.2.3.1. Introduction

OpenCV has a **FaceRecognizer** class for face recognition, so one can start experimenting with face recognition right away. The currently available algorithms are:

- Eigenfaces
- Fisherfaces
- Local Binary Patterns Histograms

The Eigenface approach began with a search for a low-dimensional representation of face images. Sirovich and Kirby (1987) showed that principal component analysis could be used on a collection of face images to form a set of basis features. These basis images, known as Eigenpictures, could be linearly combined to reconstruct images in the original training set. If the training set consists of M images, principal component analysis could form a basis set of N images, where $N < M$.

Differing from the Eigenface concept, the fisherface method tries to maximize the ratio of the between-class scatter versus the within-class scatter. The result of this shapes the projections so that the distances between the classes are at a maximum, while the distances between samples of the same class are at a minimum. A possible disadvantage is if the

between-class scatter is large, then the within-class scatter might also still be of a relatively large value.

Eigenfaces and Fisherfaces find a mathematical description of the most dominant features of the training set as a whole. LBPH analyzes each face in the training set separately and independently. More differences between different face recognition algorithms are available in[6].

LBPH concentrates on extracting local features from images. The idea is to not look at the whole image as a high-dimensional vector, but describe only local features of an object. The features extracted this way will have a low-dimensionality implicitly. But it was observed that the image representation that are given doesn't only suffer from illumination variations. The local description has to be at least a bit robust against things like scale, translation or rotation in images. The Local Binary Patterns methodology has its roots in 2D texture analysis. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighborhood. Take a pixel as center and threshold its neighbors against. If the intensity of the center pixel is greater-equal its neighbor, then denote it with 1 and 0 if not. It will end up with a binary number for each pixel, just like 11001111. So with 8 surrounding pixels it will end up with 2^8 possible combinations, called *Local Binary Patterns* or sometimes referred to as *LBP codes*. The first LBP operator described in literature actually used a fixed 3 x 3 neighborhood just like this:

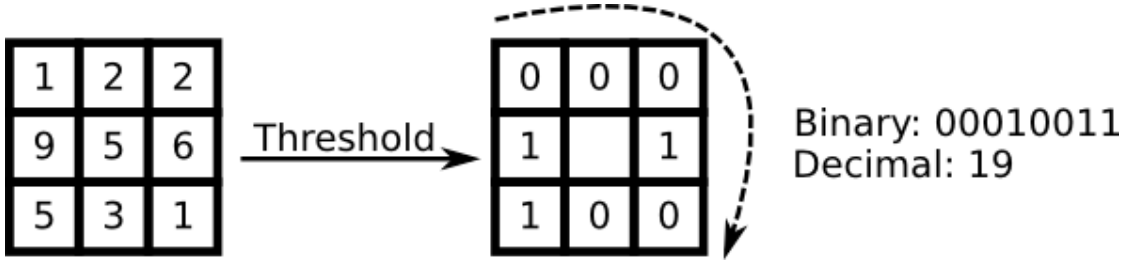


Fig 3.3: LBP operator with 3x3 neighborhood

3.2.3.2. Algorithmic Description

A more formal description of the LBP operator can be given as:

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

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \qquad (1)$$

This description enables us to capture very fine grained details in images. But it was observed that a fixed neighborhood fails to encode details differing in scale. So the operator was extended to use a variable neighborhood in [10]. The idea is to align an arbitrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:

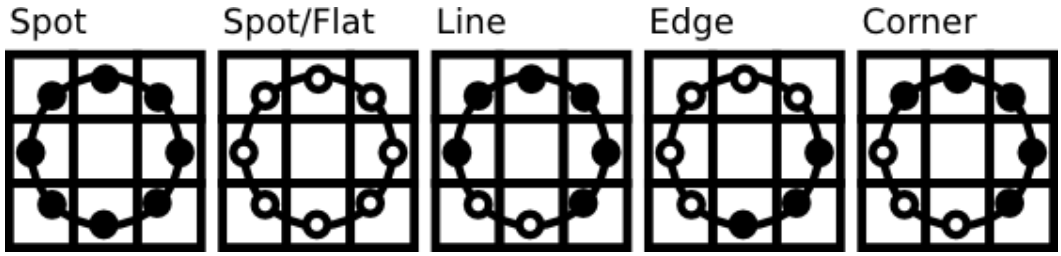


Fig 3.4: Variable neighborhood in LBPH

For a given Point (x_c, y_c) the position of the neighbor $(x_p, y_p), p \in P$ can be calculated by:

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

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

Where R is the radius of the circle and P is the number of sample points.

The operator is an extension to the original LBP codes, so it's sometimes called *Extended LBP* (also referred to as *Circular LBP*) . If a points coordinate on the circle doesn't correspond to image coordinates, the point get's interpolated. Computer science has a bunch of clever interpolation schemes, the OpenCV implementation does a bilinear interpolation:

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

By definition the LBP operator is robust against monotonic gray scale transformations. We can easily verify this by looking at the LBP image of an artificially modified image.

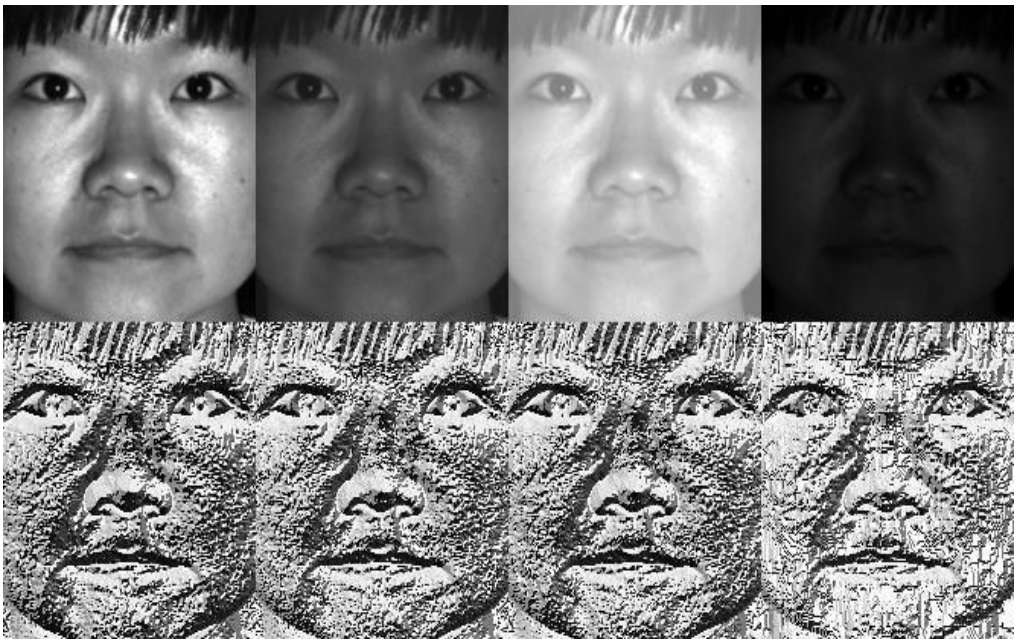


Fig 3.5: LBP image of an artificially modified image

To incorporate the spatial information in the face recognition model, the basic idea that is followed is to divide the LBP image into m local regions and extract a histogram from each. The spatially enhanced feature vector is then obtained by concatenating the local histograms (not merging them). These histograms are called *Local Binary Patterns Histograms*.

3.3. WORKING

3.3.1. Working of BuildData and Training

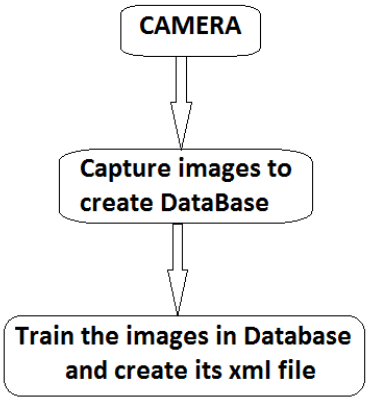


Fig 3.6: Flowchart for building database and training

(a) Import required libraries and modules: We are using camera module to capture video and we are using system time and date for timestamp. In order to use the above features we need to import the required libraries.

(b) Define path for `"which_ever_object_to_detect.xml"` using variable `cascpath`: Set the path to the `haarcascades` depending on what type of objects you want to detect.

(c) Create an object `object_cascade` of the class `cv2.CascadeCassifier(above_path)`: For face like objects we have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits).

(d) Create a recognizer object using `cv2.createLBPHfacerecognizer()`: An object of `cv2.face.createLBPHFaceRecognizer()` class is created. LBPH stands for Local Binary Patterns Histograms which is the algorithm provided by OpenCV for face recognition.

(e) Define a `get_images_and_labels` function in order get the list of Images and their Labels**:** Pass the path of the database as parameter. Using function `os.path.join(path, f) for f in os.listdir(path)` go into the directory, then read the image and convert to gray-scale and also Convert the image format into numpy array. Detect the face in the image. If face is detected, append the face to '`images`' and the label to '`labels`'.

(f) Set path to create Database: Define the path where the database must be created and store it in a variable named '`dirpath`'.

(g) Create directory: Using `os.mkdir()` create the directory to store the images.

(h) Open the camera by creating '`camera`' object of `cv2.VideoCapture()` class: In order to capture the video, the Video Capture object '`cap`' is created here. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera.

(i) Define Images list: Declare a list with variable name `images`.

(j) Read the frame using read function of '`camera`' object and store it in variable frame: `cv2.VideoCapture.read()` usually returns two values. 0^{th} value is the Boolean value that gives true if a frame is grabbed and false if camera has been disconnected, or there are no more frames in video file. 1st frame decodes and returns the grabbed video frame, and this is the one that we need. Hence we specify '1' in square braces and assign it to a variable '`frame`' by writing `frame = camera.read()[1]` where 'camera' is cv2.VideoCapture object.

(k) Convert the captured frame to gray scale and store it in variable '`gray`': Using the function `cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)` convert the image to the gray scale.

(l) Using `faceCascade.detectMultiScale` detected faces are returned as a list of rectangles in list faces: This function detects the actual face – and is the key part of our code, so let's go over the options. The detectMultiScale function is a general function that detects objects. Since we are calling it on the face cascade, that is what

it detects. A simple, green colored rectangle is drawn around the Detected face `cv2.rectangle`.

(m) Show the Image and number of faces detected: Using `cv2.imshow` display the frame and the faces detected. Give the name of the window as parameter to the function `cv2.imshow()`.

(n) If Escape key is pressed jump to step r: The control is jumped out of the while loop.

(o) If 's' button is pressed append it to images list.

(p) Jump to step j.

(q) Release the camera, destroy all windows.

(r) Save Images list. Define path of the images in database.

(s) Call the `get_images_and_labels` function and get the face images and the corresponding labels

(t) Using `recognizer.train()` train the images that are present in database.

(u) Save the trained database using the function `recognizer.save()` and name it as trained.xml.

3.3.2. Working of Face Recognition

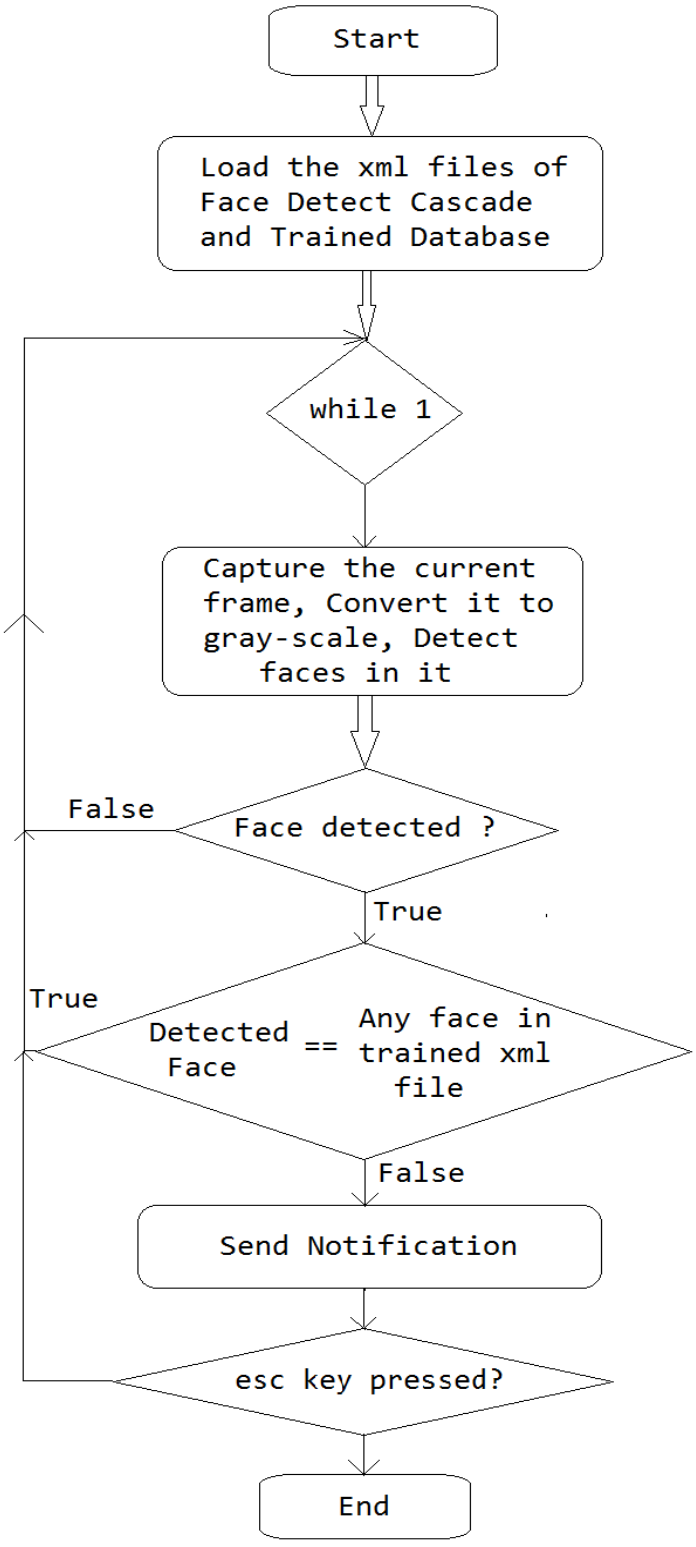


Fig 3.7: Flowchart for face detection and recognition

(a) Import required libraries and modules: We are using camera module to capture video and we are using system time and date for timestamp. In order to use the above features we need to import the required libraries.

(b) Starts a SMTP Gmail connection: It uses module `'startserver()'` from `emailattach` program which takes two arguments, Gmail ID and password of the system.

(c) Define '`cascpath`': The cascade is just an XML file that contains the data to detect faces. Specify the path to `'haarcascade_frontalface_alt.xml'` which is required for face detection.

(d) Creating the cascade and initialize it with face cascade: This loads the face cascade into memory so it's ready for use. The OpenCV cascade breaks the problem of detecting faces into multiple stages. For each block, it does a very rough and quick test. If that passes, it does a slightly more detailed test, and so on. The algorithm may have 30-50 of these stages or cascades, and it will only detect a face if all stages pass. The advantage is that the majority of the pictures will return negative during the first few stages, which means the algorithm won't waste time testing all 6,000 features on it. Instead of taking hours, now face detection can be done in real time.

(e) Create arrays to append images and labels, later for training. Set count to 1 which is used to store the count of intruders detected.

(f) An object of `cv2.face.createLBPHFaceRecognizer()` class is created: An object of `cv2.face.createLBPHFaceRecognizer()` class is created. LBPH stands for Local Binary Patterns Histograms which is the algorithm provided by OpenCV for face recognition.

(g) Load the trained images of the database using `cv2.CascadeClassifier.load(filename)` where `filename` is the name of the file from which the classifier is loaded. The file may contain an old haar classifier trained by the `haartraining` application or a new cascade classifier trained by the `traincascade` application.

(h) Define Images list: Declare a list with variable name `Images`.

(i) Press Escape key to exit from the program.

(j) Take a variable ‘`roi`’ and set it to None. '`roi`' is used to check whether face is detected or not.

(k) Create '`cap`' object of `cv2.VideoCapture()` class: In order to capture the video, the Video Capture object camera is created here. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera.

(l) Capture the current frame from camera object using read function: `cv2.VideoCapture.read()` usually returns two values. 0^{th} value is the Boolean value that gives true if a frame is grabbed and false if camera has been disconnected, or there are no more frames in video file. First frame decodes and returns the grabbed video frame, and this is the one that we need. Hence we specify '1' in square braces and assign it to a variable '`frame`' by writing `frame = cap.read()[1]` where '`cap`' is cv2.VideoCapture object.

(m) `FaceCascade.multiscale()` :This function detects the actual face – and is the key part of our code, so let’s go over the options. The detectmultiscale function is a general function that detects objects. Since we are calling it on the face cascade, that is what it detects.

(n) Show the faces in captured image

(o) If '`roi`' is None, No face is detected in the frame and no need to perform any recognition.

(p) If '`roi`' is not None, assign gray-scale image of current frame to a variable '`testImage`' and display the Security Feed.

(q) Create a Numpy array using the '`testImage`' and of datatype 'uint8'. Uinit8 is unsigned integer (0 to 255)

(r) Take a integer variable '`lcount`' and set it to zero. Define a array '`onlyFaces`'

(s) The method `append()` appends a passed object into the existing list. Gray scale image of the detected face is appended to '`onlyFaces`'.

(t) Create an empty array '`newTemp`'. Convert the Detected face to numpy array, display it and save it.

(u) There are chances of occurance of a run time error 'TypeError' when we use `model.predict(newtemp)`. To avoid it, we use Exception Handling concept. We observed that this error won't occur when faces are matching. Hence we send a notification to the user with captured frame when TypeError occurs.

(v) If confidence is less than a defined value and label is greater than 0 only then the matching of the image with the database must be shown else it must show as intruder is detected and email must be sent to the user saying Intruder is detected.

(w) Destroy all created windows and release the camera. Also end the SMTP Gmail connection: `em.exitserver()`. Logout from Gmail server using the '`exitserver`' module written in `emailattach` program. `cap.release()` closes video file or capturing device of the camera object of `cv2.VideoCapture` created. `cv2.destroyAllWindows()` function destroys all of the opened High GUI windows.

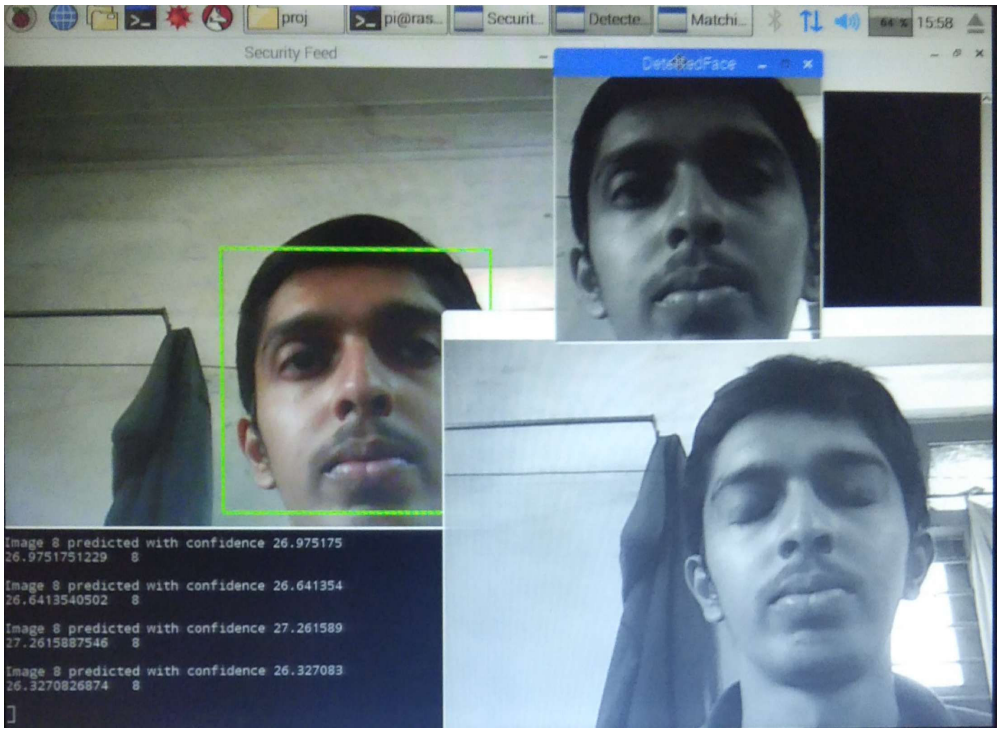


Fig 3.8: Image when face of the owner is recognized

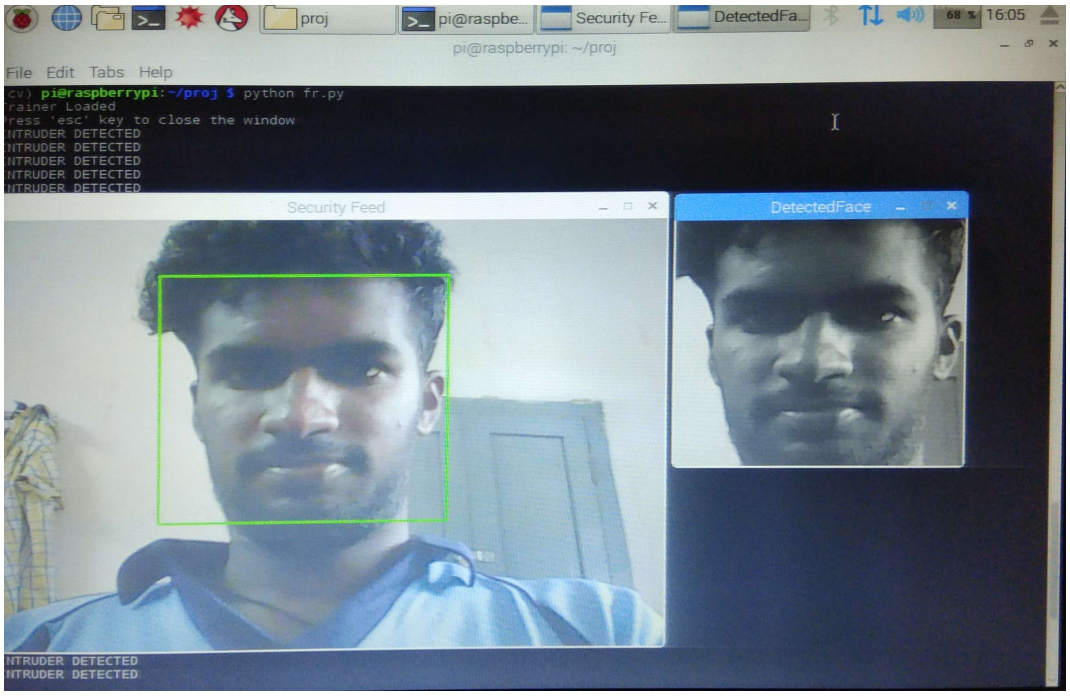


Fig 3.9: Image when intruder is detected

4. SENDING NOTIFICATION

4.1. INTRODUCTION

The notification sending part is the final part. Whenever either the motion is detected or the face is not recognized the notification has to be sent. The notification is sent through email. The email address of the owner is entered in the program. An email with an alert message and the captured image is sent to the owner. The mail can be sent to multiple users if we enter multiple email addresses.

4.2. SMTP and MIME

The actual mail transfer is done through message transfer agents. To send mail, a system must have the client Mail Transfer Agent (MTA), and to receive mail, a system must have a server MTA. The formal protocol that defines the MTA client and server in the Internet is called the Simple Mail Transfer Protocol (SMTP). As we said before, two pairs of MTA client/server programs are used in the most common situation. Figure 4.2 shows the range of the SMTP protocol in this scenario. SMTP is used two times, between the sender and the sender's mail server and between the two mail servers.

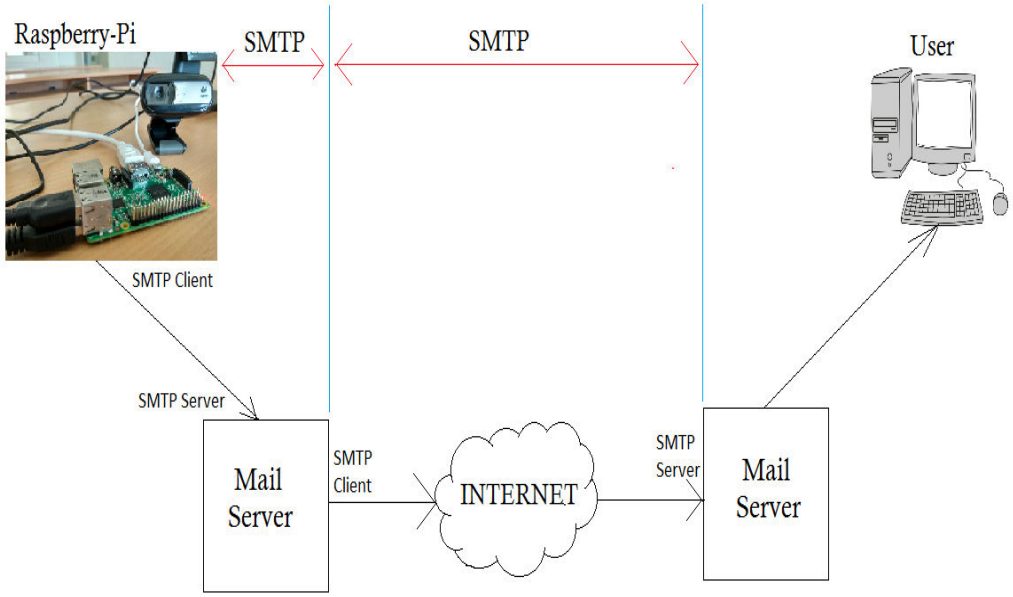


Fig 4.1: Range of SMTP Protocol

We make use of a native library in Python - "`smtplib`" to send emails. The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or Extended SMTP (ESMTP) listener daemon. SMTP connection is put in Transport Layer Security (TLS) mode to ensure security. Port number 587(for smtp.gmail server) is used for sending emails.

Electronic mail has a simple structure. Its simplicity, however, comes at a price. It can send messages only in NVT 7-bit ASCII format. In other words, it has some limitations. For example, it cannot be used for languages that are not supported by 7-bit ASCII characters (such as French, German, Hebrew, Russian, Chinese, and Japanese).

Also, it cannot be used to send binary files or video or audio data. Multipurpose Internet Mail Extensions (MIME) is a supplementary protocol that allows non-ASCII data to be sent through email. MIME transforms non-ASCII data at the sender site to NVT ASCII data and delivers them to the client MTA to be sent through the Internet. The message at the receiving side is transformed back to the original data.

We can think of MIME as a set of software functions that transforms non-ASCII data (stream of bits) to ASCII data and vice versa We need MIME protocols to send attachments.

4.3. WORKING

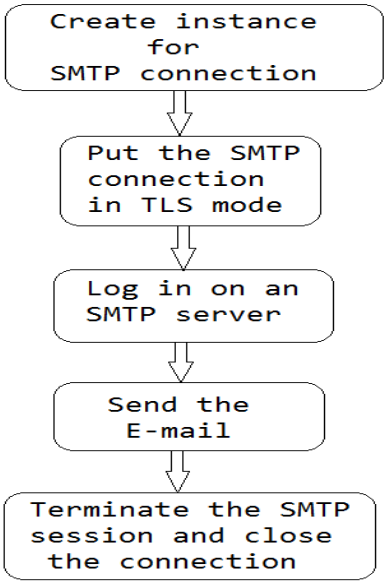


Fig 4.2: Flowchart for sending notification

(a) We have written a separate code '`emailattach.py`' which have modules to setup SMTP connection, send mail and to end the SMTP connection. We import this code in Motion detection and Face Recognition Programs.

(b) Import required libraries and modules: We need '`smtplib`' library for sending mail, '`encoders`' to encode the messages and MIME libraries to send attachments.

(c) Create an SMTP object, each object is used for connection with one server: It takes two arguments. The first argument is the server's hostname, the second is the port. The port used varies depending on the server.

(d) A module to start SMTP connection is created: Put the SMTP connection in TLS mode. All SMTP commands that follow will be encrypted. Log in on an SMTP server that requires authentication. The arguments are the username and the password to authenticate with.

(e) A module to send the email is created: It requires parameters such as from address, to address, password, file path, filename, and text message to send. MIMEMultipart objects are created and attachments are attached to the mail. The messages are encoded and mail is sent using the module `sendmail(from_addr, to_addr, message_with_attachments)` provided by native library '`smtplib`'.

(f) Create a module to end SMTP connection: It ends SMTP connection using the module `quit()` provided by native library '`smtplib`'.

5. RESULTS AND SUMMARY

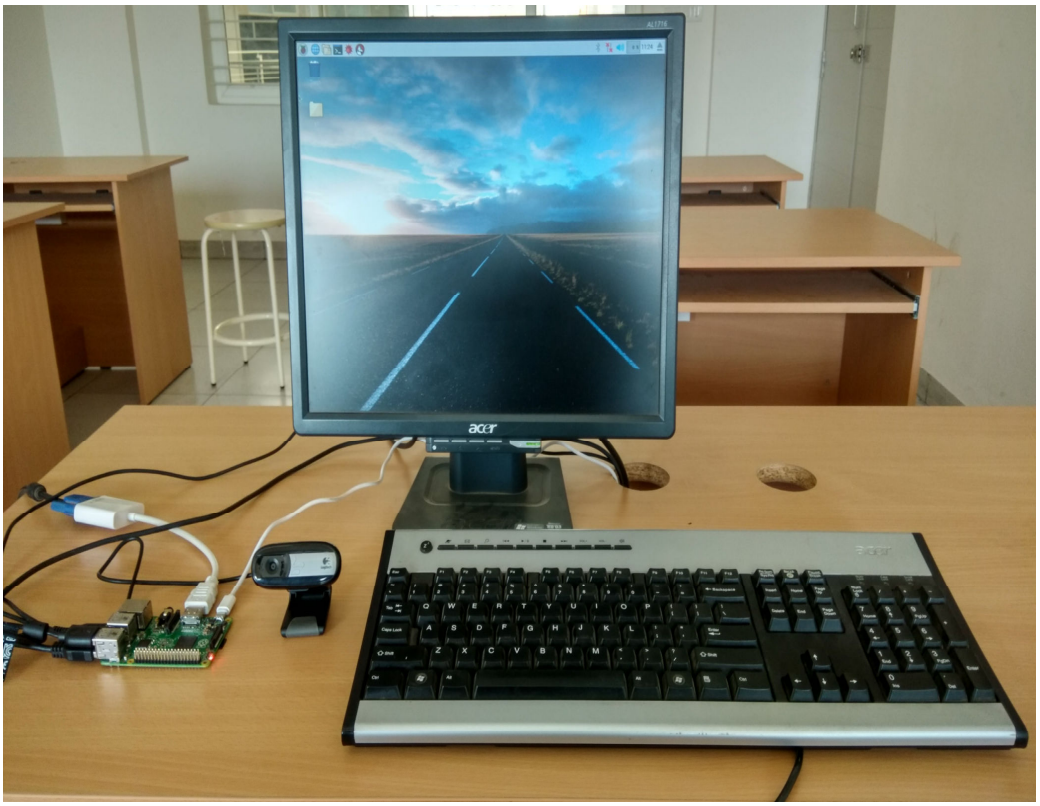


Fig 5.1: Complete project module

The home owner gets the notification with the motion detected frame along with the time stamp on the frame. The face of the person is detected and recognized. The comparison between the captured face and the faces stored in the database is performed. The notification is sent to owner using Gmail for both the motion detection and face recognition to owner separately.

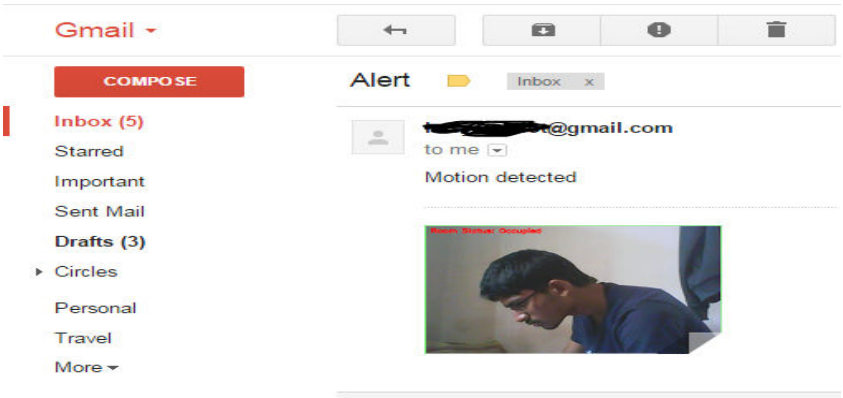


Fig 5.2: Image of email notification upon motion detection

In motion detection if we take first frame as base frame as in[9], suppose during the capture of the first frame if there was a person or an object and not during the second frame or the further frames then it will continuously show that the motion is detected because even though there was no person or that object in the second frame at that instant but the first frame has the person and it is continuously compared with the base frame and it will always show a person or an object difference. Because of this the home owner will continuously get notification from his mail saying motion is detected. Hence capture of base frame should be done carefully.

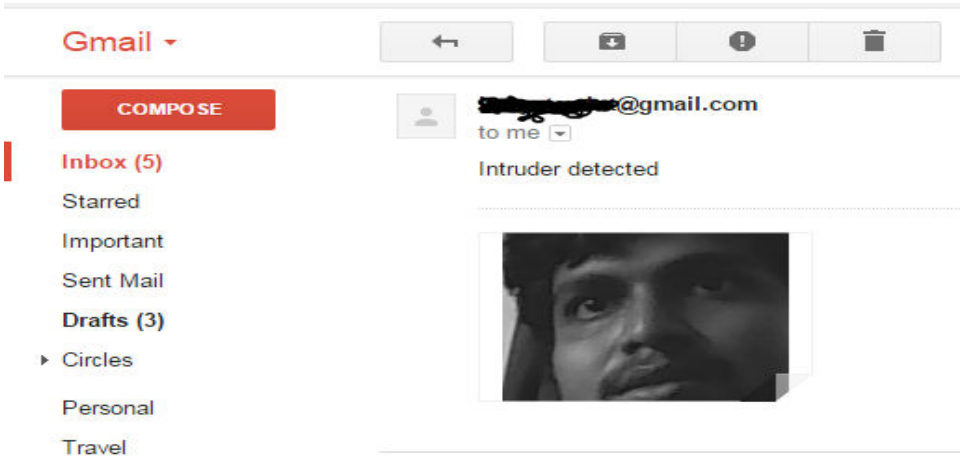


Fig 5.3: Image of email notification upon face recognition

In face detection and recognition, the database should consist of images of faces taken in different positions and in different day lights. Suppose images for the database are taken during bright day then the owner may be recognized as intruder during cloudy day. For error corrections in the code, we have referred[8].

6. CONCLUSIONS AND FUTURE WORK

6.1. CONCLUSION

In current work we have designed the system such that it is able to do both the motion detection and face recognition process. The motion detection helps the home owner keep an eye on his home without any worries. Face detection and recognition part helps the family people to easily enter into their home whereas non-family members are detected as intruders and alert the home owners. The work carried out here has been published in[9].

6.2. FUTURE WORK

The project of home surveillance can be extended to do the live streaming process. The live streaming part helps to access the live video of the activities inside the home through mobile or desktop which has internet connectivity.

This can also be improved by developing the separate android, apple, windows apps which has all the options for motion detection, face detection and recognition and live streaming part. The user can select what he wants to use through that app. We can also send the notification and alert messages through that app only without using Gmail server.

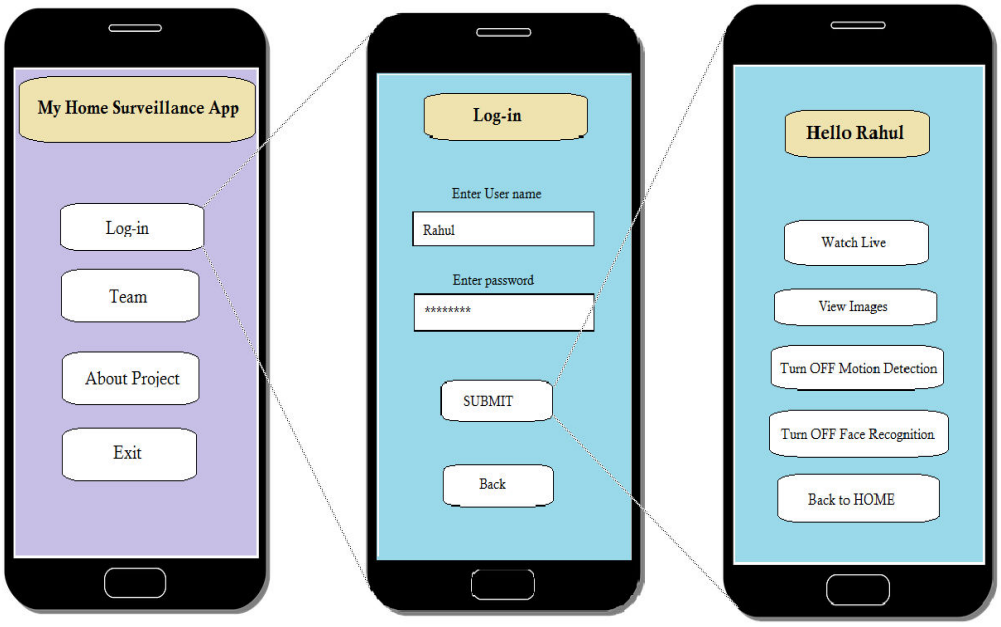


Fig 6.1: Application to control Home Surveillance System that could be developed

REFERENCES

[1] https://en.wikipedia.org/wiki/Surveillance

[2] https://www.python.org

[3] http://docs.opencv.org

[4] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[5] Viola, Paul and Michael J. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001. Volume: 1, pp.511–518.

[6] Belhumeur, P. N., Hespanha, J., and Kriegman, D. *Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection.* IEEE Transactions on Pattern Analysis and Machine Intelligence 19, 7 (1997), 711–720.

[7] Behrouz A. Forouzan, Data Communications & Networking, 4th Edition, McGraw-Hill,2007

[8] https://www.stackoverflow.com

[9] Adarsh Prabhakar Chantar, Charan G Bhakta, Manjunath S G, Manojkumar H M. *Home Surveillance System using Raspberry Pi.* International Research Journal of Engineering and Technology Volume 4, Issue 4 , Apr2017, p.p. 2770-2773

[10] Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen. *Face recognition with local binary patterns*. Computer vision-eccv 2004, p.p. 469–481.