CHAPTER 1

Introduction

1.1 Problem Statement:

The World Health Organization (WHO) reported that there are 285 million visually-impaired people worldwide. Among these individuals, there are 39 million who are totally blind. There have been several systems designed to support visually-impaired people and to improve the quality of their lives. Unfortunately, most of these systems are limited in their capabilities. In this report, we present a portable assistive device Elektra for visually-impaired people. Thus, the contribution of this report is to discuss in detail the device that is presented in the literature to assist this population and highlight the improvements, advantages, disadvantages, and accuracy. Our aim is to address and present most of the issues of these systems to pave the way for other researchers to design devices that ensure safety and independent mobility to visually-impaired people.

1.2 Project Overview:

The rapid intensive use of technology nowadays has led into a dramatic increase in the demand of its usage in our daily life and makes it more comfortable. There are large numbers of visually-impaired people, which led us to develop such system in order to help them to avoid obstructions. Smart technology has helped blind people in many different life aspects, such as ascending stairs, reading e-mails and using computers and mobile phones. According to World Health Organization (WHO) official statistics in 2009, there are 314 million visually impaired persons in the world in which 45 million of them are blind and 87% of them are from developing countries. Aging is the main factor for blind people and 19% of the world's population, which are above 50 years old are more exposed to lose their vision. Therefore, several methods and devices have been developed and employed to serve blind people as guidance or in any other life aspects.

The ability and capability of vision to human being is an important factor of our life. In this report, we introduce one smart system which is nothing but smart devise which will help visually impaired people. It helps the person to detect obstacle and help them to assist in their daily life. The smart device works on voice based commands.

Elektra is an intelligent guide for the blind person is successfully designed, implemented and tested. The prototype device is facilitating the movement of blind person by warning him/her about any nearby obstacles in order to help him/her during daily activities. The guidance will be provided in the form of audio instructions through the headset and based on real-time situation.

1.3 Hardware Specifications

The hardware used in designing the Elektra is as follows:

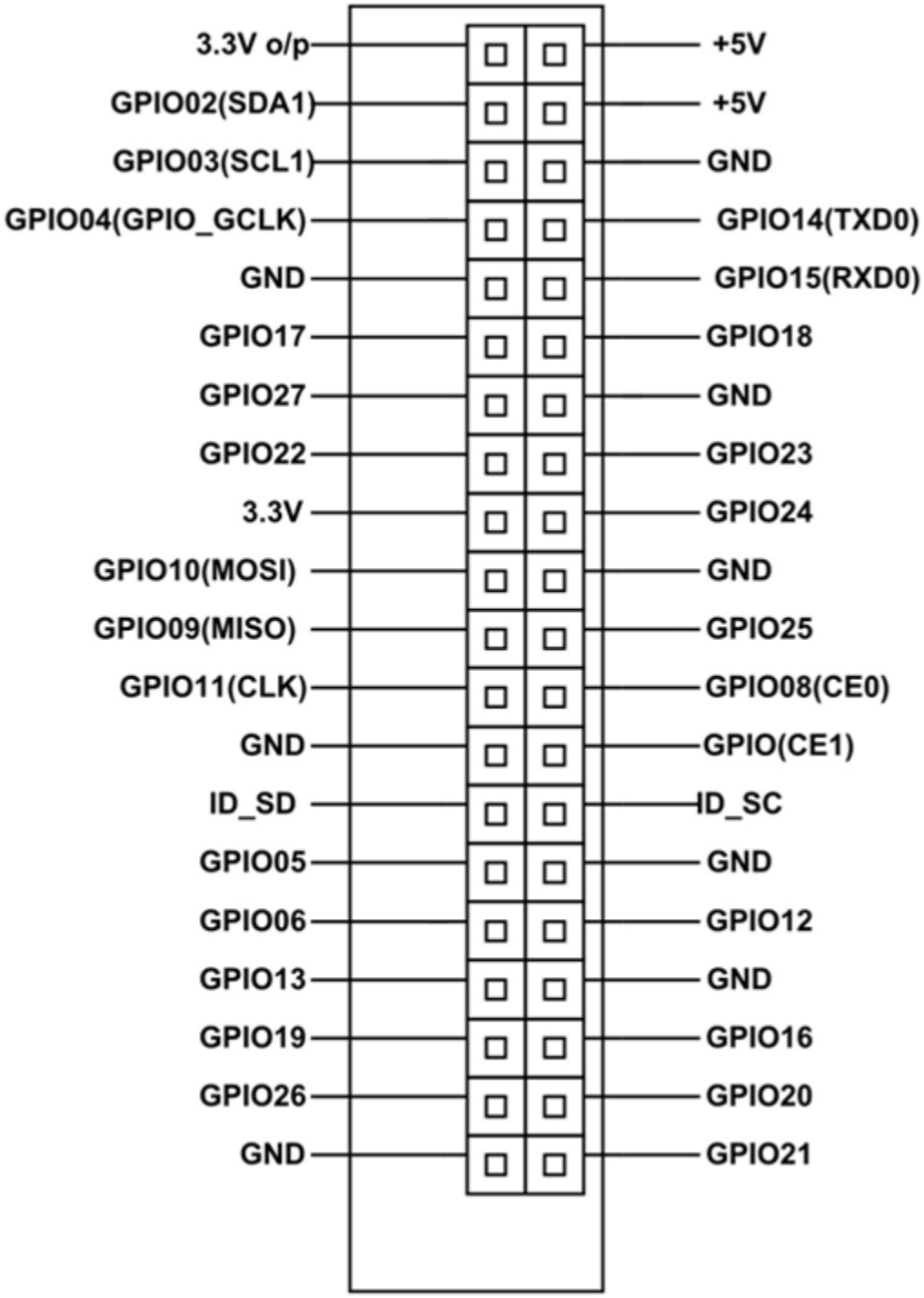
1. Raspberry Pi 3B+ board
2. Display screen
3. Card Reader
4. HDMI cable
5. Camera
6. Ultrasonic sensor
7. Voltage divider
8. Resistors
9. Headphones
10. Micro SD Card
11. Relay
12. Bluetooth

1.3.1 Raspberry Pi 3B+ board:

The Raspberry Pi 3B+ is the fastest Raspberry Pi available, both in terms of raw performance and Wi-Fi speeds.

Fig. Raspberry Pi 3b board

Pin Diagram:


3.3V o/p
GPIO02(SDA1)
GPIO03(SCL1)
GPIO04(GPIO_GCLK)
GND
GPIO17
GPIO27
GPIO22
3.3V
GPIO10(MOSI)
GPIO09(MISO)
GPIO11(CLK)
GND
ID_SD
GPIO05
GPIO06
GPIO13
GPIO19
GPIO26
GND
+5V
+5V
GND
GPIO14(TXD0)
GPIO15(RXD0)
GPIO18
GND
GPIO23
GPIO24
GND
GPIO25
GPIO08(CE0)
GPIO(CE1)
ID_SC
GND
GPIO12
GND
GPIO16
GPIO20
GPIO21

Raspberry Pi-3 Pin Configuration

PIN GROUP	PIN NAME	DESCRIPTION
POWER SOURCE	+5V, +3.3V, GND and Vin	+5V -power output +3.3V -power output GND – GROUND pin
COMMUNICATION INTERFACE	UART Interface(RXD, TXD) [(GPIO15,GPIO14)]	UART (Universal Asynchronous Receiver Transmitter) used for interfacing sensors and other devices.
	SPI Interface(MOSI, MISO, CLK,CE) x 2 [SPI0-(GPIO10 ,GPIO9, GPIO11 ,GPIO8)] [SPI1--(GPIO20 ,GPIO19, GPIO21 ,GPIO7)]	SPI (Serial Peripheral Interface) used for communicating with other boards or peripherals.
	TWI Interface(SDA, SCL) x 2 [(GPIO2, GPIO3)] [(ID_SD,ID_SC)]	TWI (Two Wire Interface) Interface can be used to connect peripherals.
INPUT OUTPUT PINS	26 I/O	Although these some pins have multiple functions they can be considered as I/O pins.
PWM	Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19	These 4 channels can provide PWM (Pulse Width Modulation) outputs. *Software PWM available on all pins
EXTERNAL INTERRUPTS	All I/O	In the board all I/O pins can be used as Interrupts.

Raspberry Pi 3 Technical Specifications

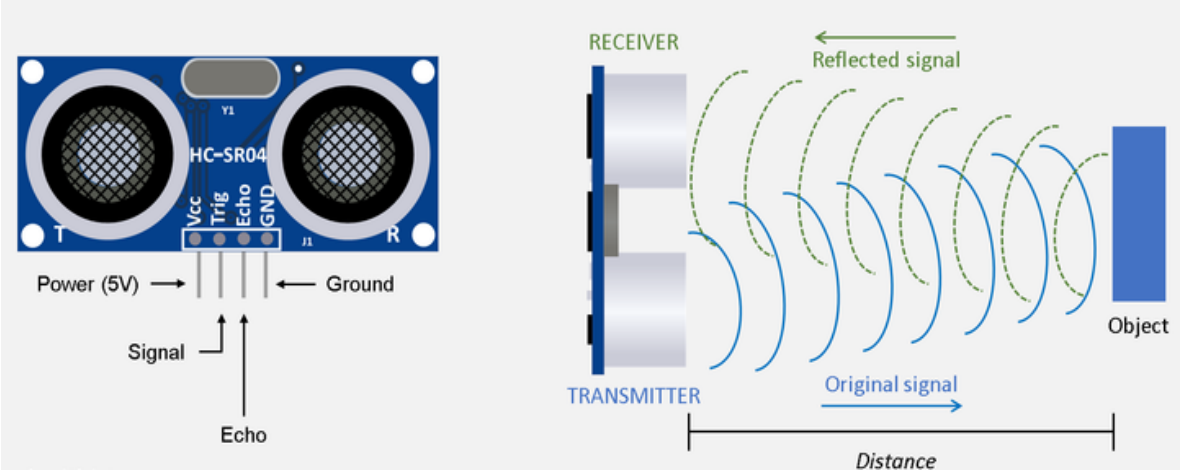
Microprocessor	Broadcom BCM2837 64bit Quad Core Processor
Processor Operating Voltage	3.3V
Raw Voltage input	5V, 2A power source
Maximum current through each I/O pin	16mA
Maximum total current drawn from all I/O pins	54mA
Flash Memory (Operating System)	16Gbytes SSD memory card
Internal RAM	1Gbytes DDR2
Clock Frequency	1.2GHz
GPU	Dual Core Video Core IV® Multimedia Co-Processor. Provides Open GLES 2.0, hardware-accelerated Open VG, and 1080p30 H.264 high- profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure.
Ethernet	10/100 Ethernet
Wireless Connectivity	BCM43143 (802.11 b/g/n Wireless LAN and Bluetooth 4.1)
Operating Temperature	-40ºC to +85ºC

Board Connectors

NAME	DESCRIPTION
Ethernet	Base T Ethernet Socket
USB	2.0 (Four sockets)
Audio Output	3.5mm Jack and HDMI
Video output	HDMI
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane.
Memory Card Slot	Push/Pull Micro SDIO

1.3.2 Ultrasonic sensor (HC-SR04)

- **HC-SR04** is an ultrasonic sensor mainly used to determine the distance of the target object.
- It measures accurate distance using a non-contact technology – A technology that involves no physical contact between sensor and object.
- Transmitter and receiver are two main parts of the sensor where former converts an electrical signal to ultrasonic waves while later converts those ultrasonic signals back to electrical signals.
- The sensor works with the simple formula - **Distance = Speed × Time**



HC-SR04 Sensor Features

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz

Ultrasonic Sensor Pin Configuration

Pin Number	Pin Name	Description
1	Vcc	The Vcc pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

1.3.3 Voltage Divider

The ECHO output is of 5V. The input pin of Raspberry Pi GPIO is rated at 3.3V. So 5V cannot be directly given to the unprotected 3.3V input pin. Therefore we use a voltage divider circuit using appropriate resistors to bring down the voltage to 3.3V. The following equation can be used for calculating resistor values,

Vout = Vin x R2/ (R1+R2)

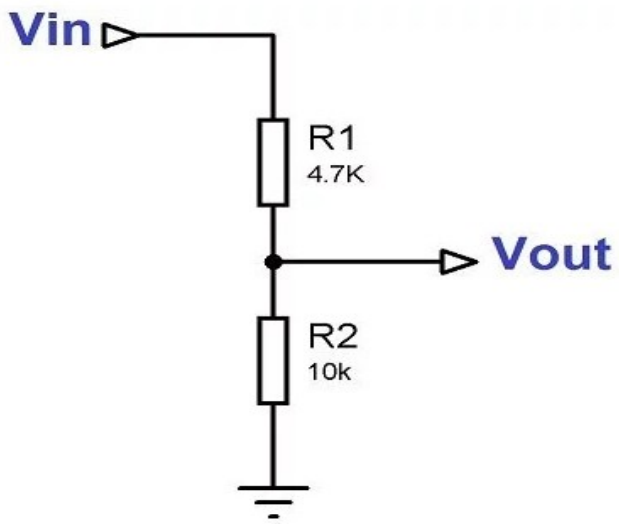


Fig. Circuit diagram of Voltage Divider

1.3.4 LCD Display Screen

It is a 16x2 display screen.

It displays the voice commands given by user and the output generated. It displays number of faces detected and distance calculated.

The screen brightness is regulated by potentiometer.

Fig. Display screen

1.3.5 Camera

The **Pi camera module** is a portable light weight camera that supports Raspberry Pi. It communicates with Pi using the MIPI camera serial interface protocol. It is normally used in image processing, machine learning or in surveillance projects. It is commonly used in surveillance drones since the payload of camera is very less. Apart from these modules Pi can also use normal USB webcams that are used along with computer.

Fig. Camera

Pin Description

Pin Number	Pin Name	Description
1	Ground	System Ground
2,3	CAM1_DN0, CAM1_DP0	MIPI Data Positive and MIPI Data Negative for data lane 0
4	Ground	System Ground
5,6	CAM1_DN1, CAM1_DP1	MIPI Data Positive and MIPI Data Negative for data lane 1
7	Ground	System Ground
8,9	CAM1_CN, CAM1_CP	These pins provide the clock pulses for MIPI data lanes
10	Ground	System Ground
11	CAM_GPIO	GPIO pin used optionally
12	CAM_CLK	Optional clock pin
13,14	SCL0, SDA0	Used for I2C communication
15	+3.3V	Power pin

1.4 Software Specification

The project is divided into three modules:

- Face Detection Module
- Proximity Calculation Module
- Smart Appliances Module

1.4.1 Face Detection Module

Face detection is a type of application classified under “computer vision” technology. It is the process in which algorithms are developed and trained to properly locate faces or objects in images. These can be in real time from a video camera or from photographs.

1.4.1.1 Understanding Haar Cascade

A **facial identification system** is a technology capable of identifying a face of a person from a digital image or a video frame from a video source.

Haar Cascade classifier is based on the Haar Wavelet technique to analyses pixels in the image into squares by function. This uses "integral image" concepts to compute the "features" detected. Haar Cascades use the **AdaBoost** learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers then use cascading techniques to detect face in a image.

Haar cascade classifier is based on Viola Jones detection algorithm which is trained in given some input faces and non-faces and training a classifier which identifies a face.

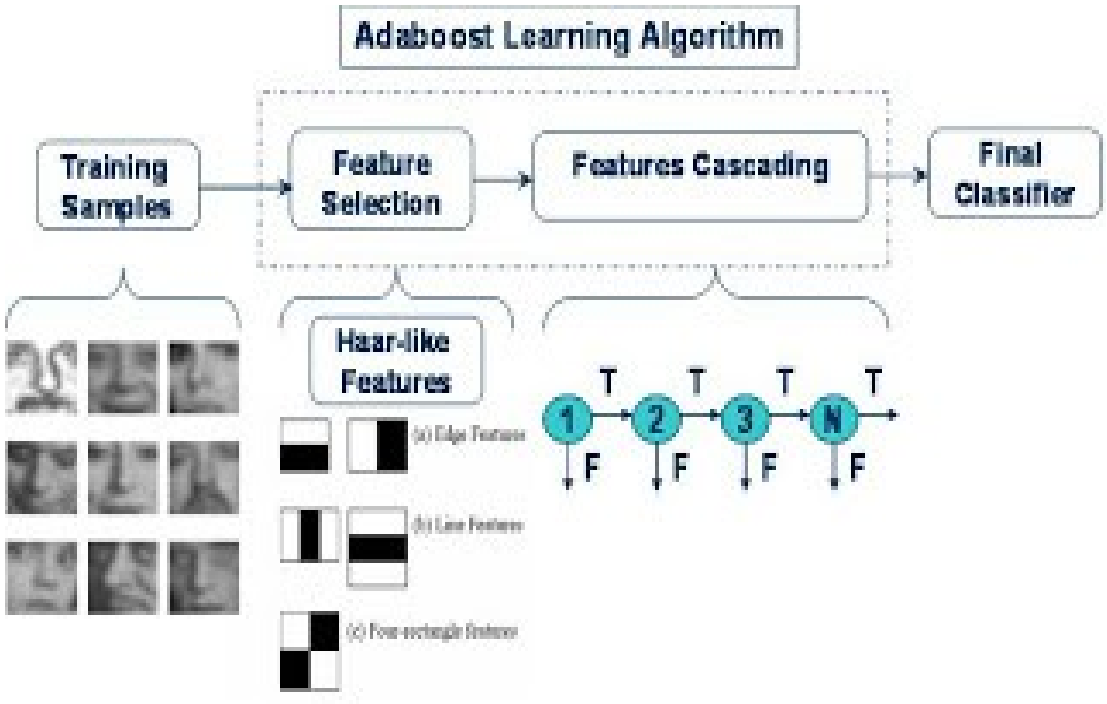


Fig. **Architecture for the Face Detection**

.

1.4.1.2 Feature Extraction

Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images. This process in the algorithm is feature extraction. In feature extraction, the algorithm uses training data to best identify features that it can consider a face.

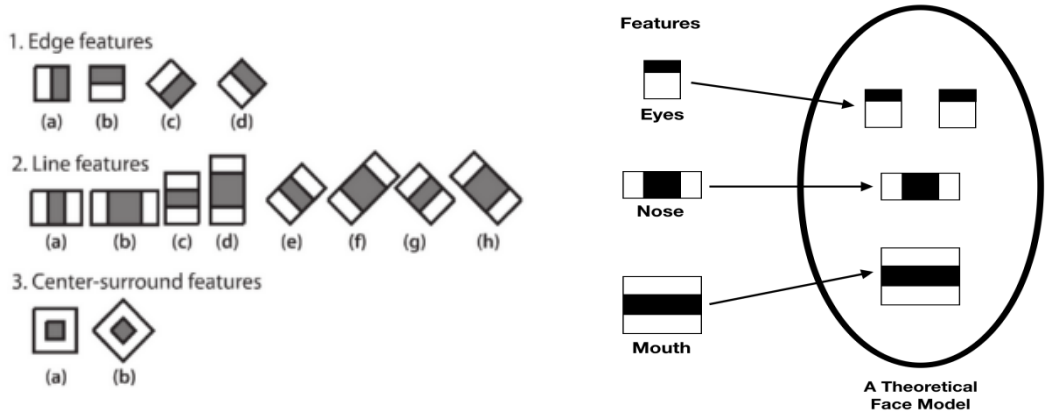


Fig. Haar Feature Extraction

Here are some Haar-Features. The first two are **edge features**, used to detect **edges**. The third is a **line feature**, while the fourth is a **four rectangle feature**, most likely used to detect a **slanted line**. Haar features are similar to these **convolution kernels** which are used to detect the presence of that feature in the given image.

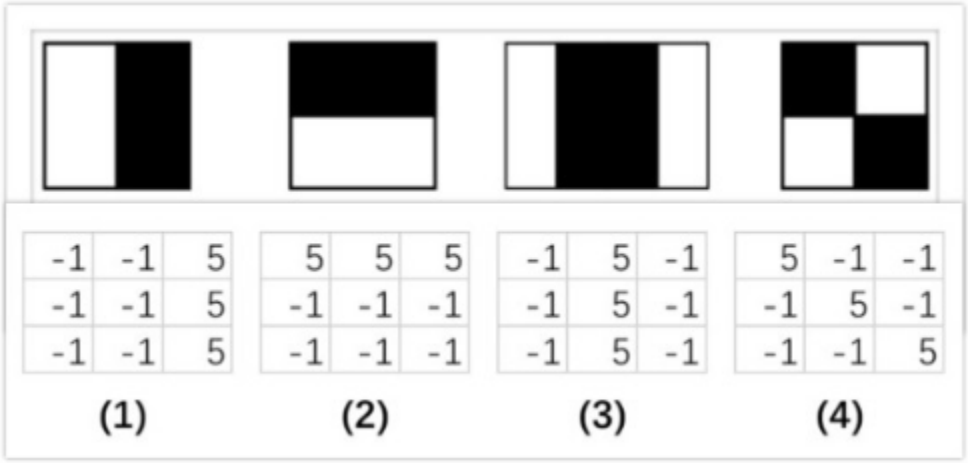


Fig. Haar Feature Representation using convolution kernels

Each feature results in a single value which is calculated by subtracting the sum of pixels under white rectangle from the sum of pixels under black rectangle.

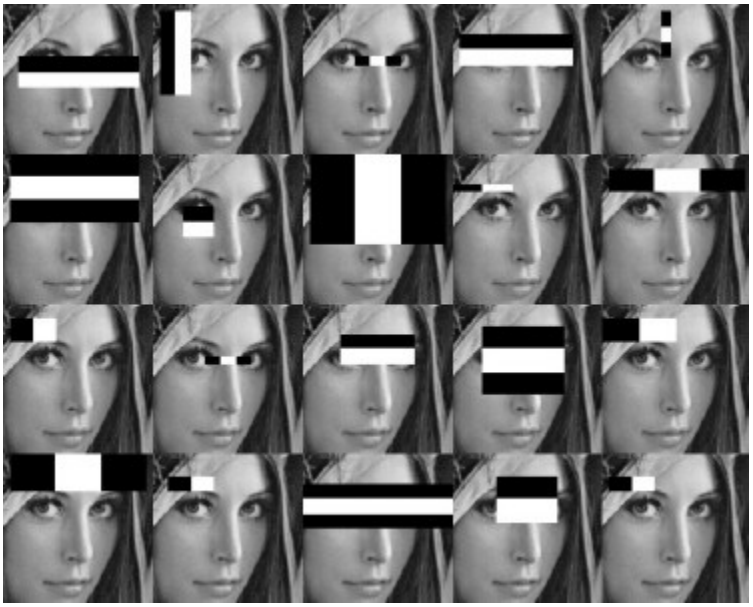


Fig. Haar Features

Every haar feature has some sort of resemblance to identify a part of face. Viola Jones uses 24*24 as base window size and calculates the above features all over the image shifting by 1 PX.

If we consider all possible parameters of the haar features like position, scale and type we end up calculating about 160,000+ features. So we need to evaluate huge set of features for every 24*24 PX .So to avoid this we have a idea to avoid redundant features and pick only those features which are very useful for us. This can be done using AdaBoost.

1.4.1.4 AdaBoost

AdaBoost is used to remove redundant features and choose only relevant features. AdaBoost is used to get the best features among all these 160,000+ features. These features are also called as weak classifiers. After these features are found a weighted combination of all these features in used in evaluating and deciding any given window has a face or not.

Each of the selected features (weak classifiers) are considered okay to be included if they can at least perform better than random guessing. Each of the weak classifier is relevant detecting a part of face. Output of weak classifier is binary if it has identified a part of a face or not.

AdaBoost constructs a strong classifier as a linear combination of these weak classifiers.

Strong classifier = linear sum of weak classifiers

$$F(x) = \sum (\alpha_i * f_i(x))$$

here α_i are corresponding weights to each weak classifier $f_i(x)$.

1.4.1.5 Cascading

Given a input image we need to move our 24*24 window all over the image and compute 2,500 features for every window and take a linear combination of all outputs and see if it exceeds a certain threshold or not.

Instead of calculating 2,500 features for every window we use idea of cascades. We do a sampling of 2,500 features into x different cascades. Now we can detect if there is face or not in different cascades linearly. If $cascade_i$ finds a face in a image then image is passed to next cascade. If no face is found is a cascade we can move to next window. This reduces the time complexity.

1.4.2 Proximity Calculation Module

The Proximity Calculation module calculates the distance between the blind person and the hindrance coming in his path while walking.

This module is implemented through ultrasonic sensor. The sensor has two main components:

- Echo
- Receiver

There are four pins in this sensor:

- Vcc
- GND
- TRIG
- ECHO

The distance is calculated by, **Speed = Distance/Time.**

1.4.2.1 Sensing with Python

First, import the Python GPIO library, import our time library (so we make our Pi wait between steps) and set our GPIO pin numbering.

```
import RPi.GPIO as GPIO

import time

GPIO.setmode (GPIO.BCM)
```

Next, we need to name our input and output pins, so that we can refer to it later in our Python code. We'll name our output pin (which triggers the sensor) GPIO 23 [Pin 16] as TRIG, and our input pin (which reads the return signal from the sensor) GPIO 24 [Pin 18] as ECHO.

```
TRIG = 23

ECHO = 24
```

We'll then print a message to let the user know that distance measurement is in progress. . . .

```
print "Distance Measurement In Progress"
```

Next, set your two GPIO ports as either inputs or outputs as defined previously.

```
GPIO.setup (TRIG,GPIO.OUT)
```

GPIO.setup(ECHO,GPIO.IN)

Then, ensure that the Trigger pin is set low, and give the sensor a second to settle.

GPIO.output(TRIG, False)

print "Waiting For Sensor To Settle"

time.sleep(2)

The HC-SR04 sensor requires a short 10uS pulse to trigger the module, which will cause the sensor to start the ranging program (8 ultrasound bursts at 40 kHz) in order to obtain an echo response. So, to create our trigger pulse, we set out trigger pin high for 10uS then set it low again.

GPIO.output(TRIG, True)

time.sleep(0.00001)

GPIO.output(TRIG, False)

Now that we've sent our pulse signal we need to listen to our input pin, which is connected to ECHO. The sensor sets ECHO to high for the amount of time it takes for the pulse to go and come back, so our code therefore needs to measure the amount of time that the ECHO pin stays high. We use the "while" string to ensure that each signal timestamp is recorded in the correct order.

The time.time() function will record the latest timestamp for a given condition. For example, if a pin goes from low to high, and we're recording the low condition using the time.time() function, the recorded timestamp will be the latest time at which that pin was low.

Our first step must therefore be to record the last low timestamp for ECHO (pulse_start) e.g. just before the return signal is received and the pin goes high.

while GPIO.input(ECHO)==0:

pulse_start = time.time()

Once a signal is received, the value changes from low (0) to high (1), and the signal will remain high for the duration of the echo pulse. We therefore also need the last high timestamp for ECHO (pulse_end).

while GPIO.input(ECHO)==1:

pulse_end = time.time()

We can now calculate the difference between the two recorded timestamps, and hence the duration of pulse (pulse_duration).

pulse_duration = pulse_end - pulse_start

We can plug this calculation into our Python script:

distance = pulse_duration x 17150

Now we need to round our distance to 2 decimal places (for neatness!)

distance = round(distance, 2)

Then, we print the distance. The below command will print the word “Distance:” followed by the distance variable, followed by the unit “cm”

print "Distance:", distance,"cm"

Finally, we clean our GPIO pins to ensure that all inputs/outputs are reset

GPIO.cleanup()

1.4.3 Smart Appliances Module

1.4.3.1 Module Agenda

To provide the user capabilities to control the smart embedded electronic devices in his/her home or workplace.

1.4.3.2 Technologies Used

- Relay
- Bluetooth
- Wi-Fi
- Domain Server
- Raspberry Pi

1.4.3.3 Implemented Module

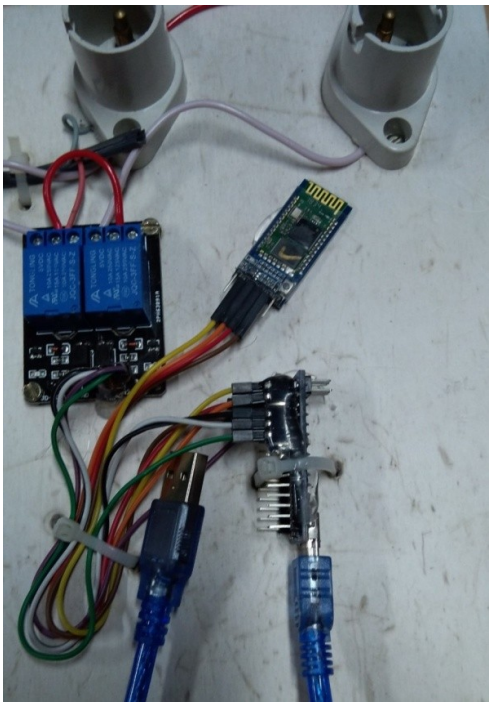


Fig. IOT Module

CHAPTER 2

Literature Survey

2.1 Existing System

The system is Arduino based. The device is embedded with sensors which vibrate when an obstacle is encountered.

2.1.1Hardware components

- Arduino x1
- Ultrasonic sensor x3
- Mobile phone vibrator x2
- Small Buzzer x1
- Power source for the Arduino

2.1.2 Schematics

The ultrasonic sensors have 4 pins each:

- Vcc
- GND
- Trig
- Echo

Connect all the Vcc and GND pins of all the sensors to the Arduino Vcc and GND pins.

Sensor 1

- Trig Pin 1 ---- Arduino A0
- Echo Pin 1 ---- Arduino A1

Sensor 2

- Trig Pin 2 ---- A2
- Echo Pin 2 ---- A3

Sensor 3

- Trig Pin 3 ---- A4
- Echo Pin 3 ---- A5

Left vibrator

- Vcc ---- Arduino Pin 5 (left sensor is 2)
- GND ---- GND

Right vibrator

- Vcc ---- Arduino Pin 6 (right sensor is 3)
- GND ---- GND

Buzzer

- Vcc ---- Arduino Pin 3
- GND ---- GND

2.1.3 Arrangement of sensors

- Place sensor 1 at the front part of the cap.
- Sensor 2 at the left side of the cap.
- Sensor 3 at the right side of the cap.
- Place the vibrators in left and right sides of the cap.

2.2 Proposed System

Elektra is a smart embedded assistive technology build for aiding the visually impaired and to help them in their day to day task. Elektra is a mini hand-held device for blind which may be used with walking cane or can be mounted inside a hat as a smart companion.

There are three modules:

- Face Detection Module
- Proximity Calculation Module
- Smart Appliances Module

Both display and voice feedbacks are provided for each mode of action or event that is occurring. We have used Raspberry Pi3B+ Model for the processing purpose. Ultrasonic sensors are used for the purpose of proximity calculation. Haar Cascade feature based classification algorithm is used for the purpose of the face detection.

The device uses less number of sensors and extracts its smartness by incorporating **Machine Learning** and **Internet of Things**

.

2.2.1 Glimpse of Proposed System

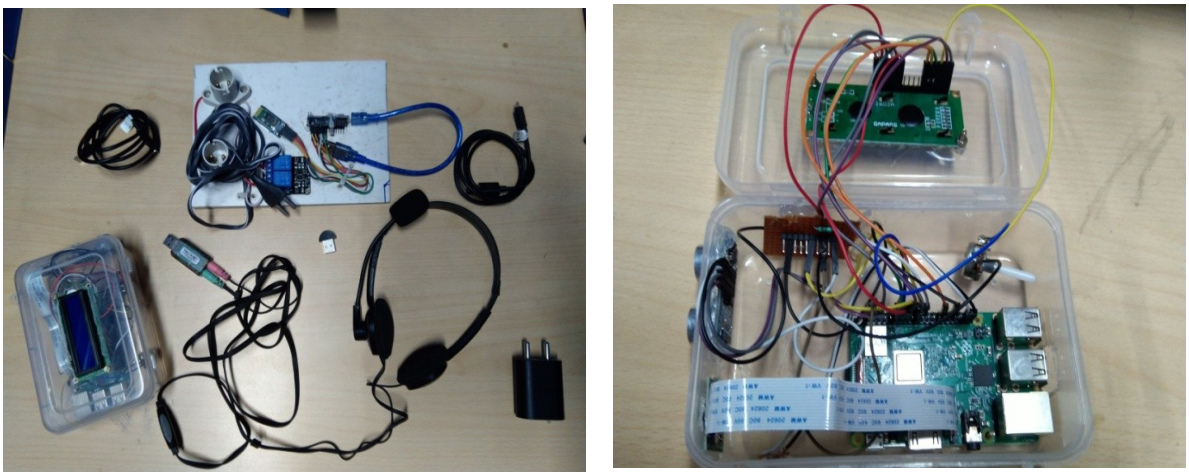


Fig. Elektra

CHAPTER 3

System Analysis and Design

3.1 Haar Feature Based Cascade Algorithm

A Haar Cascade is based on Haar Wavelets which is a sequence of rescaled square-shaped functions which together form a wavelet family or basis.

It is based on the Haar Wavelet technique to analyze pixels in the image into squares by function. This uses machine learning techniques to get a high degree of accuracy from what is called *training data*. This uses *integral image* concepts to compute the *features* detected. Haar Cascades use the AdaBoost learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers.

There are three main constituents of our face detection framework. The first constituent is an image representation called an *integral image* that allows for very fast feature evaluation. In order to compute these features very rapidly at many scales we used the integral image representation for images. The integral image can be computed from an image using a few operations per pixel. Once computed, any one of these Haar-like features can be computed at any scale or location in constant time.

The second constituent is a method for constructing a classifier by selecting a small number of important features using AdaBoost. Within any image sub window the total number of Haar-like features is very large, far larger than the number of pixels. In order to ensure fast classification, the learning process must exclude a large majority of the available features, and focus on a small set of critical features. Feature selection is achieved through a simple modification of the AdaBoost procedure: the weak learner is constrained so that each weak classifier returned can depend on only a single feature. As a result each stage of the boosting process, which selects a new weak classifier, can be viewed as a feature selection process. AdaBoost provides an effective learning algorithm and strong bounds on generalization performance.

The third major constituent is a method for combining successively more complex classifiers in a cascade structure which dramatically increases the speed of the detector by focusing attention on promising regions of the image. The notion behind focus of attention approaches is that it is often possible to rapidly determine where in an image an object might occur. More complex processing is reserved only for these promising regions. The key measure of such an approach is the *false negative* rate of the attentional process. It must be the case that all, or almost all, object instances are selected by the attentional filter. We will describe a process for training an extremely simple and efficient classifier which can be used as a *supervised* focus of attention operator. The term supervised refers to the fact that the attentional operator is trained to detect examples of a particular class. In the domain of face detection it is possible to achieve fewer than 1% false negatives and 40% false positives using a classifier constructed from two Haar-like features. The effect of this filter is to reduce by over one half the numbers of locations where the final detector must be evaluated. Those sub-windows which are not rejected by the initial classifier are processed by a sequence of classifiers, each slightly more complex than the last. If any classifier rejects the sub-window, no further processing is performed. The structure of the cascaded detection process is essentially that of a degenerate decision tree. An extremely fast face detector will have broad practical applications. These include user interfaces, image databases, and teleconferencing.

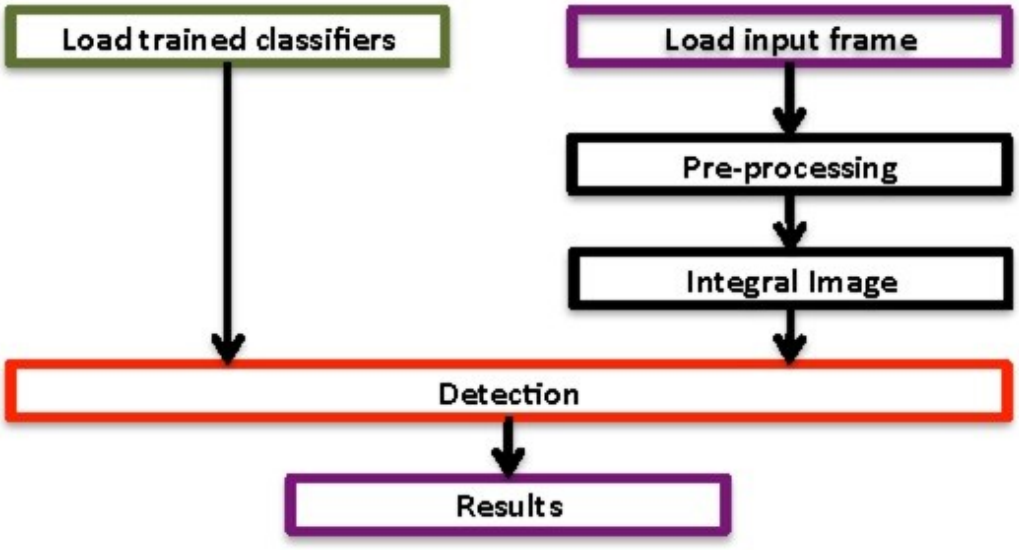


Fig. Flow Chart of Face Detection

3.1.1 Features

Our face detection procedure classifies images based on the value of simple features. There are many motivations for using features rather than the pixels directly. The most common reason is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. For this system there is also a second critical motivation for features: the feature based system operates much faster than a pixel-based system. The simple features used are reminiscent of Haar basis functions. More specifically, we use three kinds of features. The value of a two-rectangle feature is the difference between the sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent (see Figure). A three-rectangle feature computes the sum within two outside rectangles subtracted from the sum in a center rectangle. Finally a four-rectangle feature computes the difference between diagonal pairs of rectangles. Given that the base resolution of the detector is 10x10, the exhaustive set of rectangle features is quite large, over 10,000. Note that unlike the Haar basis, the set of rectangle features is over complete.

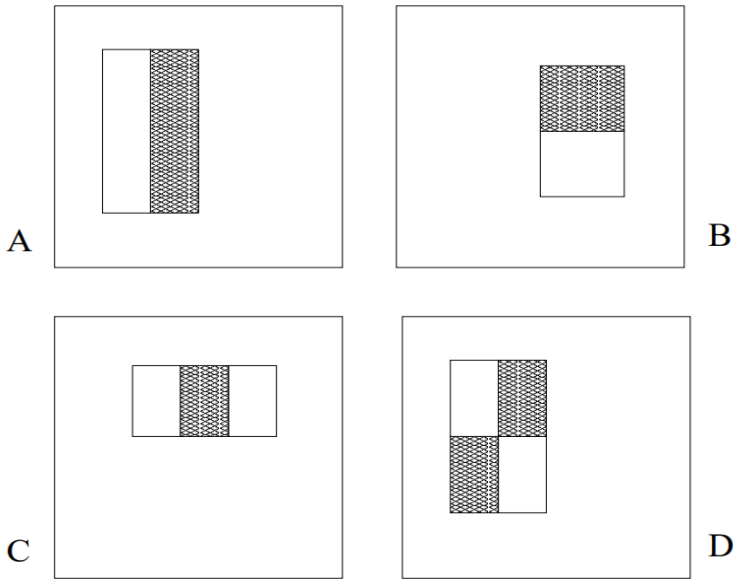


Figure: Example rectangle features shown relative to the enclosing detection window. The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

3.1.2 Integral Image

Rectangle features can be computed very rapidly using an intermediate representation for the image which we call the integral image. The integral image at location *x, y* contains the sum of the pixels above and to the left of *x, y*, inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

Where *ii(x, y)* is the integral image and *i(x, y)* is the original image. Using the following pair of recurrences:

$$s(x, y) = s(x, y-1) + i(x, y) \qquad (1)$$

$$ii(x, y) = ii(x-1, y) + s(x, y) \qquad (2)$$

where (*s(x, y)*is the cumulative row sum, *s(x, -1) = 0* , and *ii(-1, y) = 0*) the integral image can be computed in one pass over the original image. Using the integral image any rectangular sum can be computed in four array references (see Figure). Clearly the difference between two rectangular sums can be computed in eight references. Since the two-rectangle features defined above involve adjacent rectangular sums they can be computed in six array references, eight in the case of the three-rectangle features,
and nine for four-rectangle features.

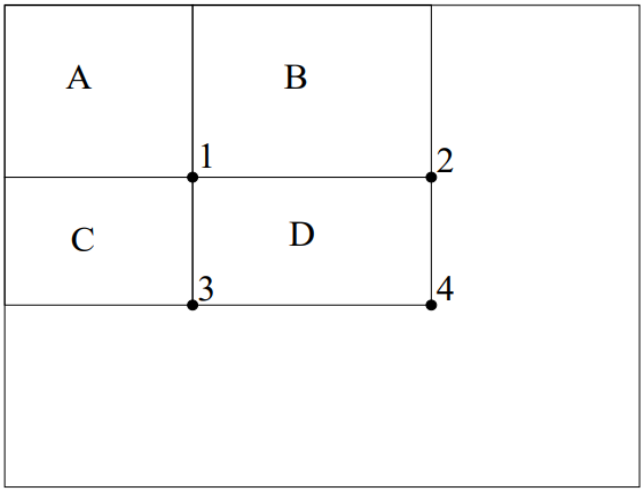


Figure: The sum of the pixels within rectangle *D* can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle *A*. The value at location 2 is *A + B*, at location 3 is *A + C*, and at location 4 is *A + B + C + D*. The sum within can be computed as*4 + 1 - (2 + 3)*.

3.1.3 AdaBoost Classifier

Ada-boost or Adaptive Boosting is one of ensemble boosting classifier proposed by Yoav Freund and Robert Schapire in 1996. It combines multiple classifiers to increase the accuracy of classifiers. AdaBoost is an iterative ensemble method. AdaBoost classifier builds a strong classifier by combining multiple poorly performing classifiers so that you will get high accuracy strong classifier. The basic concept behind AdaBoost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations. Any machine learning algorithm can be used as base classifier if it accepts weights on the training set. AdaBoost should meet two conditions:

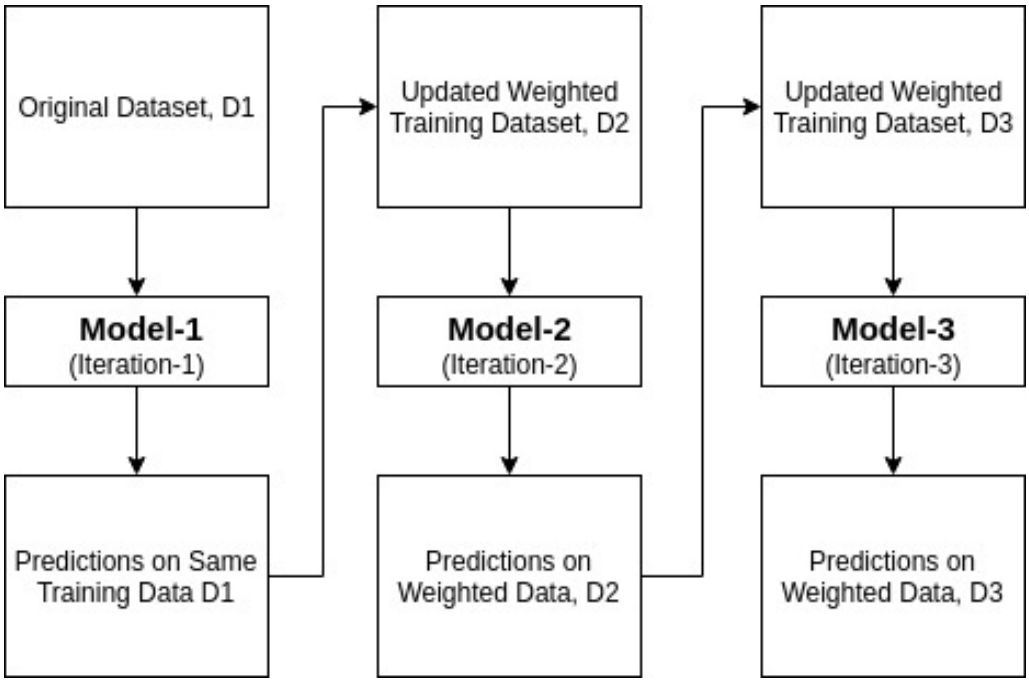
1. The classifier should be trained interactively on various weighed training examples.
2. In each iteration, it tries to provide an excellent fit for these examples by minimizing training error.

How does the AdaBoost algorithm work?

It works in the following steps:

1. Initially, AdaBoost selects a training subset randomly.
2. It iteratively trains the AdaBoost machine learning model by selecting the training set based on the accurate prediction of the last training.
3. It assigns the higher weight to wrong classified observations so that in the next iteration these observations will get the high probability for classification.
4. Also, it assigns the weight to the trained classifier in each iteration according to the accuracy of the classifier. The more accurate classifier will get high weight.
5. This process iterate until the complete training data fits without any error or until reached to the specified maximum number of estimators.

6. To classify, perform a "vote" across all of the learning algorithms you built.



Let's look at the mathematical formula and parameters.

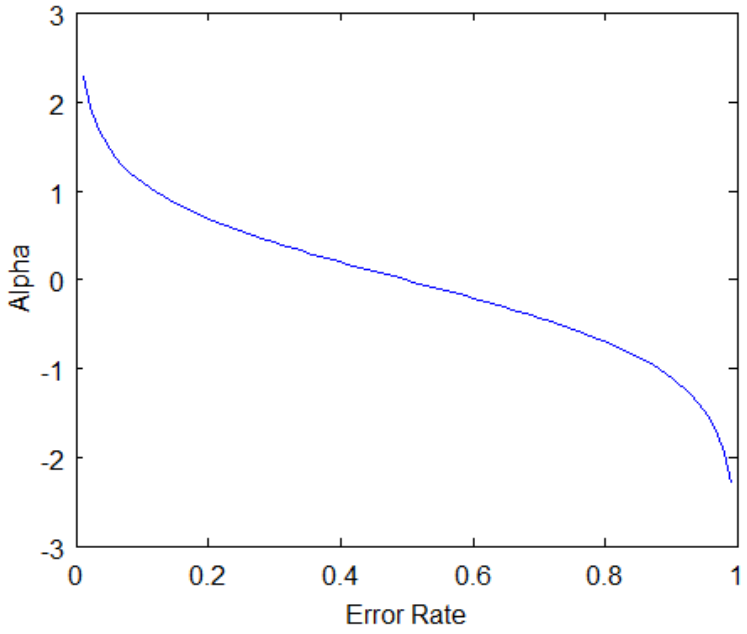
$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$h_t(x)$ is the output of weak classifier t for input x and **$alpha_t$** is weight assigned to classifier.

$alpha_t$ is calculated as follows:

$alpha_t$ = 0.5 * ln((1 — E)/E: weight of classifier is straight forward, it is based on the error rate E. Initially, all the input training example has equal weightage.

A plot of alpha$_t$ v/s error rate



Updating weight of training examples after weak classifier is trained, we update the weight of each training example with following formula:

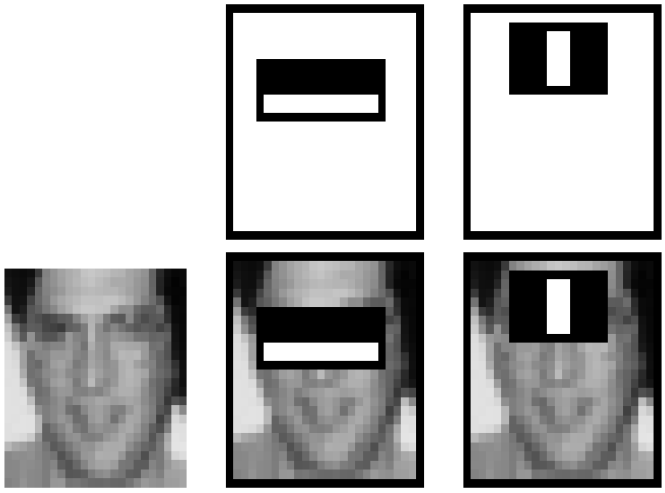
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

D_t is weight at previous level.

We normalize the weights by dividing each of them by the sum of all the weights, Z_t. For example, if all of the calculated weights added up to 15.7, then we would divide each of the weights by 15.7 so that they sum up to 1.0 instead. y_i is y par of training example (x_i, y_i) y coordinate for simplicity.

For the task of face detection, the initial rectangle features selected by AdaBoost are meaningful and easily interpreted. 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 (see Figure). This feature is relatively large in comparison with the detection sub-window, and should be somewhat insensitive to size and location of the face. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

Figure: The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.



3.1.4 Cascading

Given a input image we need to move our 24*24 window all over the image and compute 2,500 features for every window and take a linear combination of all outputs and see if it exceeds a certain threshold or not.

Instead of calculating 2,500 features for every window we use idea of cascades. We do a sampling of 2,500 features into x different cascades. Now we can detect if there is face or not in different cascades linearly. If $cascade_i$ finds a face in a image then image is passed to next cascade. If no face is found is a cascade we can move to next window. This reduces the time complexity.

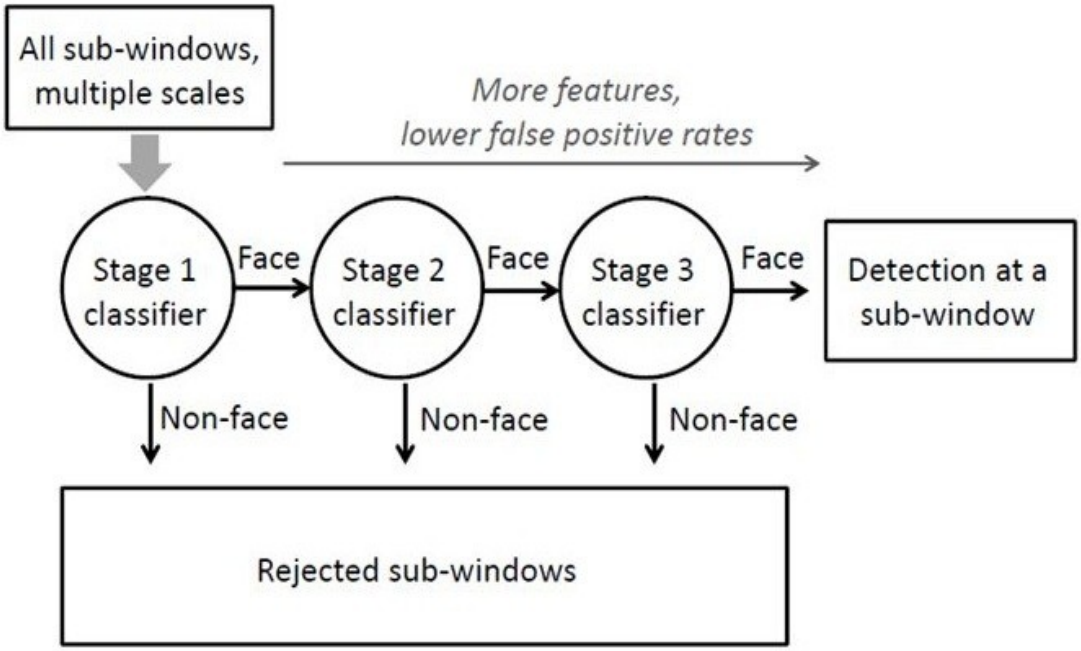


Fig. Cascading Classifiers for Detection

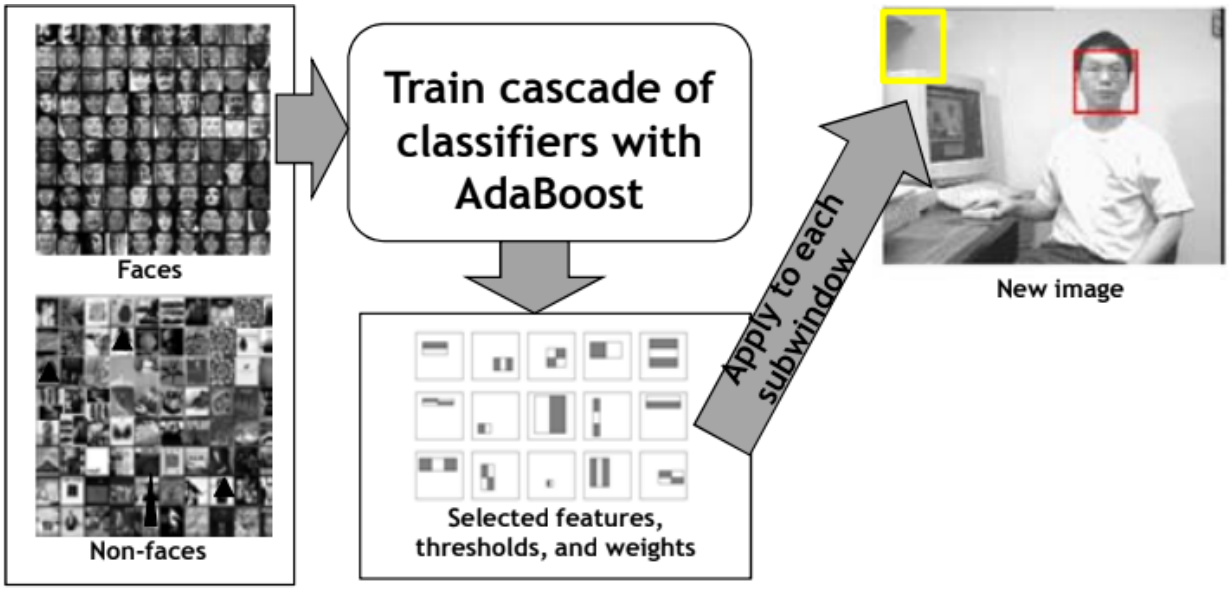
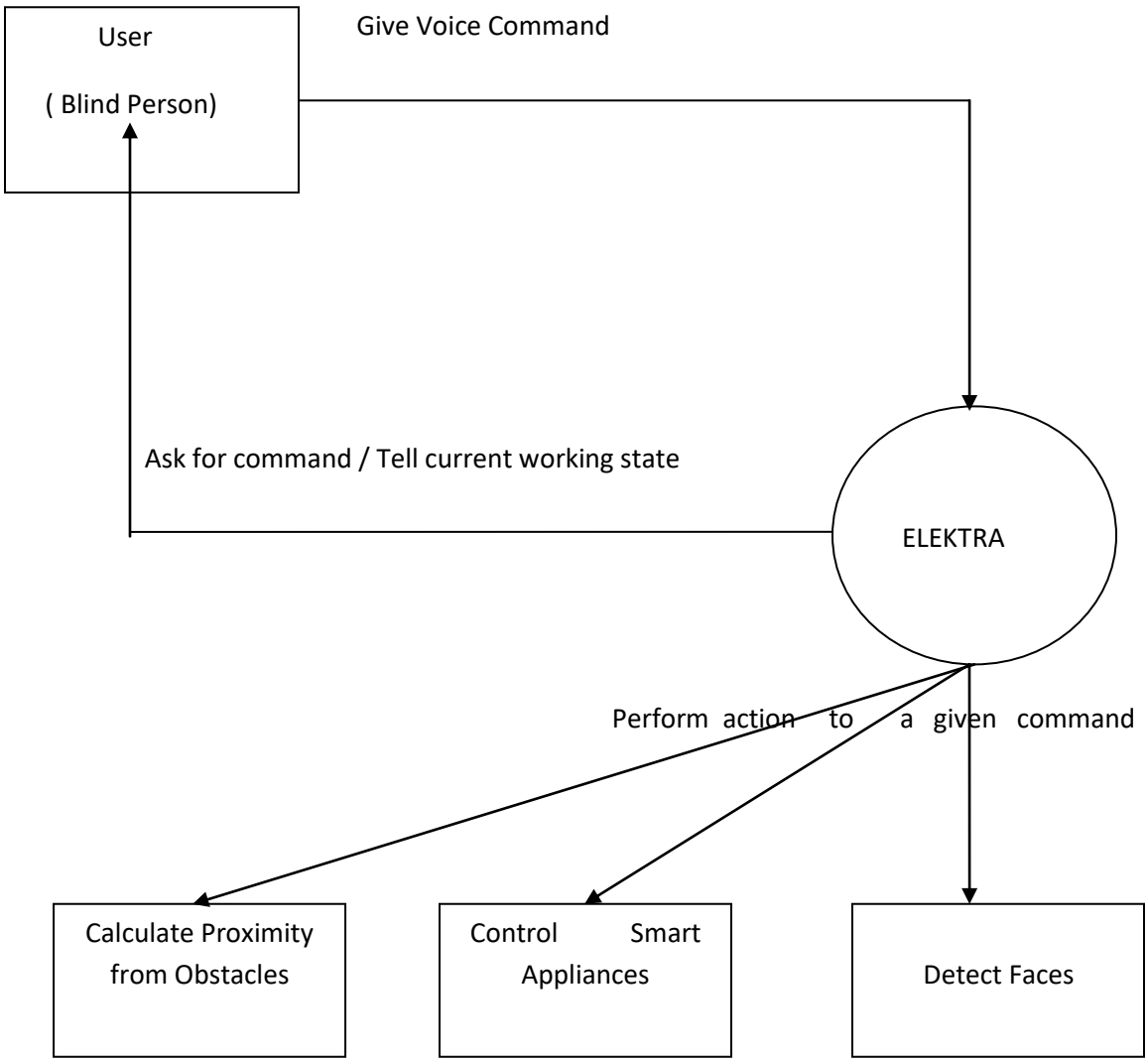


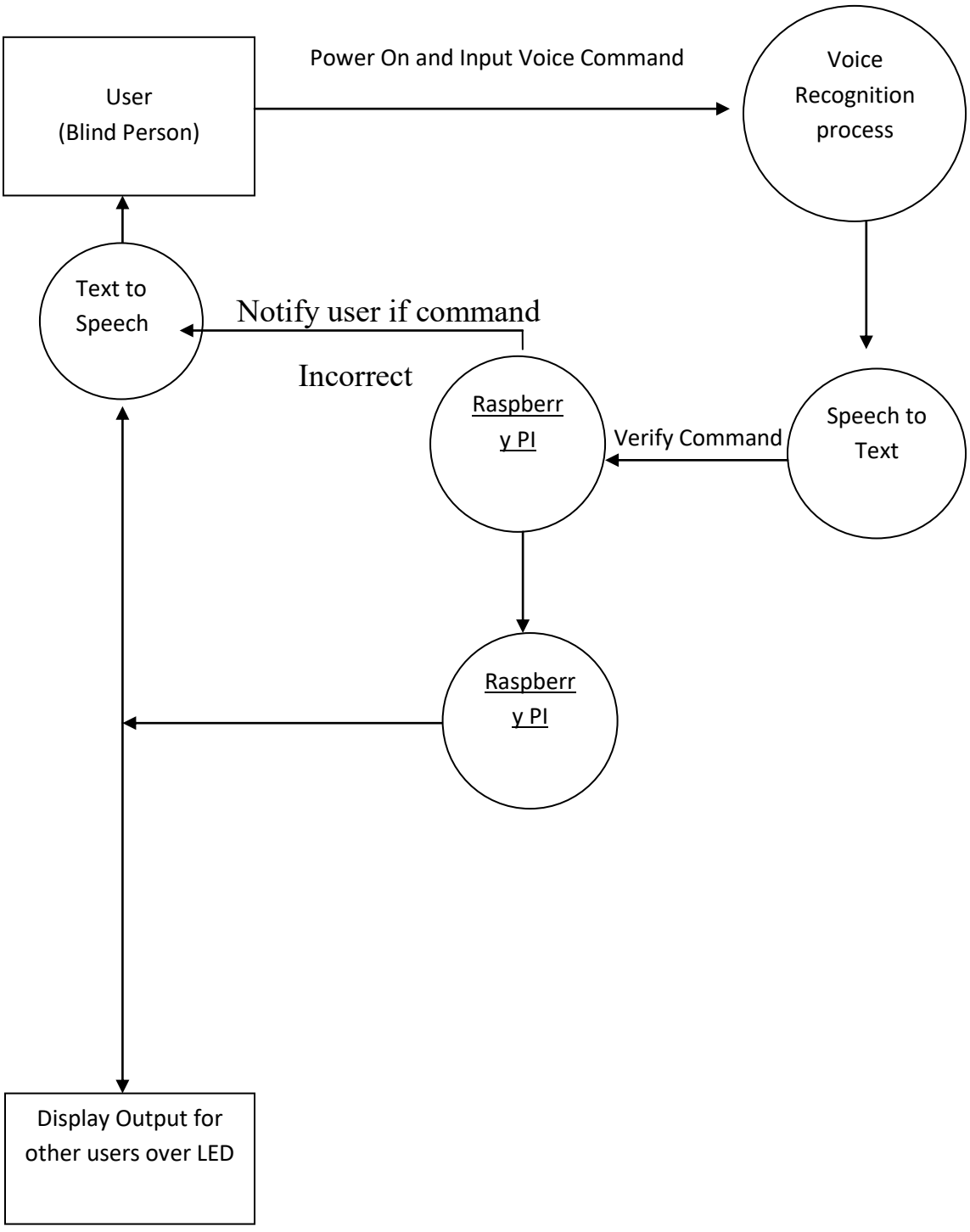
Fig. Viola Jones Summary

3.2 Data Flow Diagram

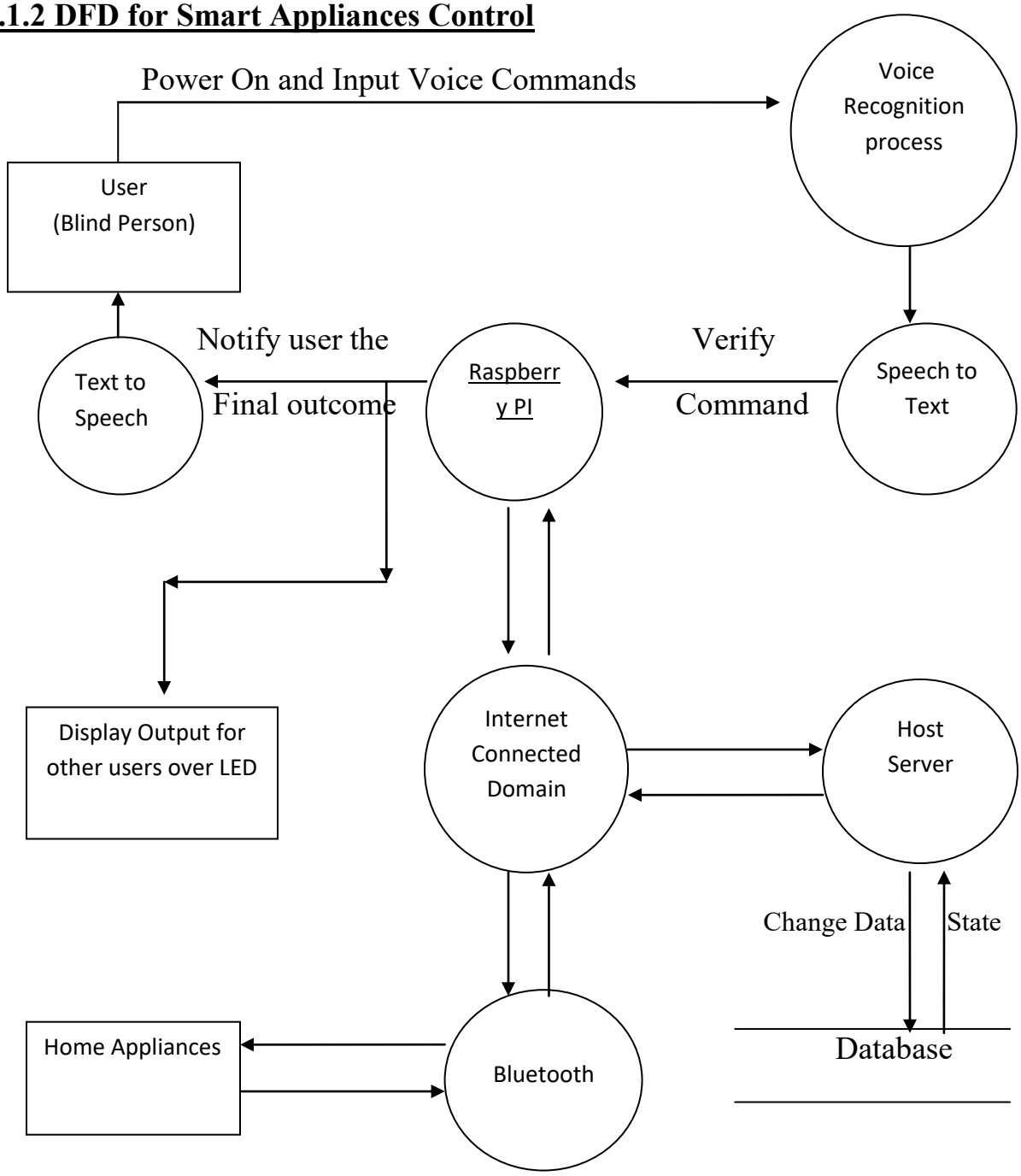
3.2.1 Zero Level DFD



3.2.1.1 DFD for Proximity Calculation :-



3.2.1.2 DFD for Smart Appliances Control



CHAPTER 4

Results and Output

As described above in this report, the reasons why we started developing this project. And the above three chapters enlightens us with a brief detail of the project motto, design, the whole development process and the technology used to develop it.

The project that our team developed that is ELEKTRA basically answers and provide solutions to the following three queries:~

1. Recognition of object without touching.
2. Controlling of embedded smart electronic devices over voice commands.
3. Makes navigation a little bit easier by telling the distance from obstacles.

Fig. Showing module – 1(object) in running state

Fig. Showing module – 2 (prximity) in running state

Fig. Showing module – 3 (IOT) in running state

CHAPTER 5

Future Enhancement

1. We are trying to incorporate the whole system and its working into a mobile app in place of Raspberry Pi by using java, android studio and some machine learning algorithms, which will :-
   1.1 Drastically increases the processing power of the device because of the larger RAM size that we get in a modern smart phone.
   1.2 The device will be able to detect variety of objects.
   1.3 The device will become more users friendly and handy to use.

Fig. ELEKTRA controlling home appliances

over voice commands

2. With the help of smart phone's Global Positioning System the user can be told his exact location and navigation to far places become very easy.

3. One of the most important enhancement on which we will be working upon is to make the device capable of scanning the texts written in English, understanding it and then converting it into the audio for the user.

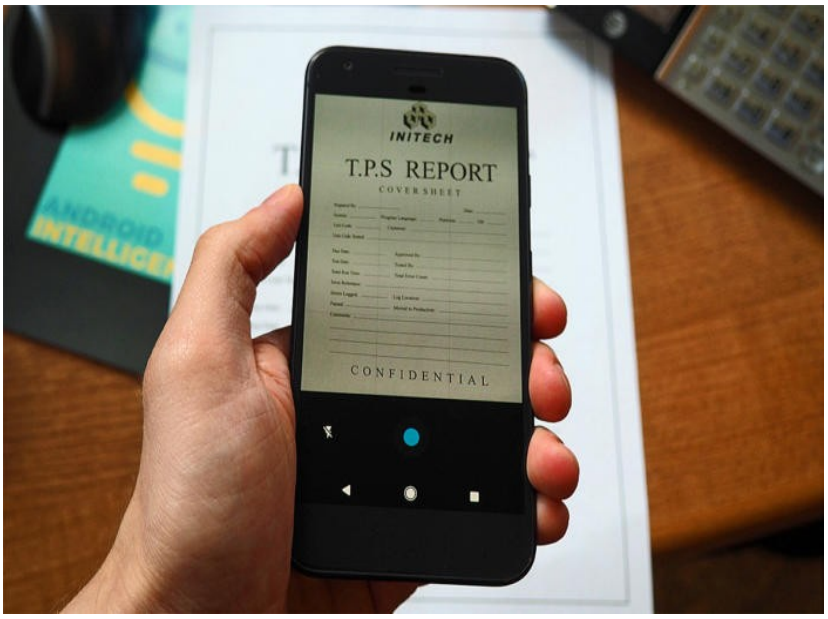


Fig. Scanning of text document

4. We will be doing face recognition along with face detection with the help of convolutional neural network.

Fig. Face Recognition

CHAPTER 6

References

- **https://thepihut.com/blogs/raspberry-pi-tutorials/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi**
- **https://medium.com/@krsatyam1996/haar-cascade-face-identification-aa4b8bc79478**
- Paul Viola, Michael Jones (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. In Conference on Computer Vision and Pattern Recognition 2001.
- www.hackster.io
- azure.microsoft.com
- 3. blogs.window.com