A

DISSERTATION REPORT

ON

HCI USING EYE-MOTION-TRACKING

IS SUBMITTED AS A PARTIAL FULFILLMENT OF THE

MASTER OF TECHNOLOGY

IN

VLSI DESIGN

BY

AKHIL T V

(2015PEV5193)

UNDER THE GUIDANCE OF

Mr. SANJEEV AGRAWAL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

JUNE 2017

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

JAIPUR RAJASTHAN 302017

CERTIFICATE

This is to certify that the M. Tech. thesis report entitled "**HCI Using Eye-Motion-Tracking"** has been successfully completed and presented by **Akhil T V (2015PEV5193)** in partial fulfillment of the degree of **Master of Technology** in **VLSI Design** in the department of Electronics and Communication Engineering during the academic year **2015-2017**. To the best of my knowledge and belief that this work has not been submitted elsewhere for the award of any other degree.

The work has been found satisfactorily carried out by him under my guidance and supervision in the department and is approved for submission.

Date :

Place :

Mr. Sanjeev Agrawal
Associate Professor
Dept. Of ECE.
MNIT Jaipur

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

JAIPUR RAJASTHAN 302017

DECLARATION

I **Akhil T V** (**2015PEV5193**) hereby declare that the dissertation entitled "**HCI Using Eye-Motion-Tracking"** being submitted by me in partial fulfillment of the degree of **Master of Technology** in **VLSI Design** in the department of Electronics and Communication Engineering during the academic year **2015-2017** in a research work carried out by me under the supervision of Mr. Sanjeev Agrawal, and the contents of this dissertation work in full or in parts have not been submitted to any other Institute or University for the award of any degree or diploma. I also certify that no part of this dissertation work has been copied or borrowed from anywhere else. In case any type of plagiarism is found out, I will be solely and completely responsible for it.

Date :

Place :

Akhil T V
2015PEV5193
VLSI Design
Dept. Of ECE.
MNIT Jaipur

ACKNOWLEDGEMENT

It has been a great experience working on the dissertation entitled "**HCI Using Eye-Motion-Tracking"** towards the partial fulfillment for the award of the degree **Master of Technology.**

I take this opportunity to express my deep sense of gratitude and respect towards my Supervisor **Mr. Sanjeev Agrawal**. I am very much indebted to him for the generosity, expertise and guidance I have received from him while working on this project. I express my sincere gratitude to my respected Head of Department, **Prof. K. K. Sharma,** all my faculty members, who helped me during these two years of Master of Technology.

At last but not least, I'm also and always be grateful to my parents and friends and expressing my deep respect towards them, who always support me and encourage me at each and every step of my career and life besides this project.

Date :

Place :

Akhil T V

2015PEV5193

ABSTRACT

Technology is dominating the world and it has become quite difficult in this world to survive for the one having no knowledge in using a computer or a smartphone. But, for the guys who are limb disabled or having any other kind of limitations in using their arms or fingers, face major difficulties in using the computer with ease. So the ways of interaction between the computer and the human have to be changed from the conventional keyboard and mouse to the ones suiting different users. The HCI field has developed huge with time, but for the limb disabled, an impressive economical way is still not in limelight. Fathoming the future possibilities in this area, an alternative way is been proposed and implemented, for the disabled, to interact with the computer. And this proposed method is using the users' face and eye motion tracking for the interaction. This system helps the user to type in a notepad without the help of the hands, without even moving the body. Different algorithms have been used in this project to work this out. The key task to be done here in this project is the user's face and the eye detection in real time, which is accomplished using the Viola-Jones algorithm. The position of the pupil is detected by detecting the circular portion in the detected eye. The gleam inside the pupil is spotted by finding the connected components in the binary image of the pupil. From these pieces of information detected, the user's direction of peeking is assessed.

This HCI has a virtual keyboard having 6 keys. Using the above methods, the key to being selected by the user is determined. The user has to stare at each key for the selected keys got pressed, or, the selected letters got typed in the textbox given in the interface window. A 720p USB webcam is used to capture the real-time images of the user. The whole system is implemented in MATLAB.

CONTENTS

LIST OF FIGURES

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 1

INTRODUCTION & REVIEW

1.1 INTRODUCTION

Our world is moving forward so fast. Everyone has to update themselves with the world to catch up its speed. The world is in a run for new knowledge, so as every individual. Only the knowledge doesn't help, but we have to apply our knowledge to different applications in our daily life. Now the computer is controlling our world. It does everything, from reading a book, to buy things to even control satellites. We can't even list the uses of the computer now in a 200-page notebook. So now knowing the computer means knowing the world itself. Having Computer knowledge has already become a necessity for survival. Realising its importance, our Indian Government has launched a programme '*Digital India*' in the year 2015. *Digital India* is a programme initiated by the Indian Government to make all the useful amenities accessible to the people electronically by a well-developed online cyber setup and by improving the Internet connectivity or by making us technically and digitally galvanised. That much will be the invasion of the computers going to structure our world, making the life almost impossible without it in the near future.

The computer doesn't do everything automatically. We have to give it instructions and pieces of information regularly, which means, we do have to interact with the computer all the time. We use keyboards, mouse etc., for that usually. Human-Computer Interaction (HCI) is the field of study dealing with it. It's not as simple as it seems. Researches have been done in HCI regularly to analyse the existing/current interfaces connecting users with the machine, diagnose the flaws if any, and to make the interfaces more affable and easy to use, which suits different users. The behaviour of the user is the most prominent factor in

designing a particular interface. For example, the functioning of the mouse should be slightly different for a left handed user from a right handed. So, the demand picks the mode.

HCI is a vast advanced area of research. Even then, for the users who can't move their hands freely experience difficulties in operating the computer. A perfect interface for the limb-disabled, with impressive functionalities, is still in its early, not-so-developed stages. Various researches have been done, various techniques have been proposed in this particular area. And most of them are utilising the eye tracking technique to record/detect user's response. In this technique, it mainly detects the direction to which the user is looking, and uses these different detected directions as the user's choice of selection in different environments.

The idea of eye tracking has been started from the late 1800s. In the 1870s, scientists started tracking the sweeping of eyes over different words while reading. Later, scientists starting making devices to track the eye's motion. At the starting they were using contact lens with a reference point, then used different light sources and the light beams reflected from the eyes to track. The use of the camera in this field boosted the eye tracking researches. The idea of application of eye tracking in the HCI was started in the 1980s, mainly to help the disabled.

Now there are different types of eye tracking methods, and these are mainly classified into 3 principle groups [1]:

1. **Eye-Affixed Tracking**: All the types of trackers gauge the angle of rotation of the eyeball. Only the way of measurement varies. In this method, a gadget is affixed to the eye (usually a contact lens with a mirror attached to it) and its movement and variations are recorded according to the eye movements.
2. **Optical-sensor Tracking**: This method makes use of optical sensors like the camera to record the beam of light reflected from the eyes, as its angle of reflection and intensity may vary with the movement of the eyeball.

3. **Electrooculography tracking**: An electrostatic field is present there around the eyes, and it varies as the eyes move or close. And to detect this field, no light source is needed. These small fluctuations in the field, as the user change the gaze, is recorded with the help of the electrodes that are positioned in the user's face, near to the eyes.

The application of eye tracking is not only limited in designing different HCIs. This area studies on how different users react on seeing different pictures, notes etc. to study their behavior or mental conditions; where their primary focus will be while reading a paper, seeing a website, watching games etc. so as to design websites, newspapers, to place advertisements in a web page/newspaper; in automobiles to find the driver's gaze, or in designing anti-sleeping alarms etc.. And in the near future, it will widen its reach, to where ever the user has to interact directly with the machine. The system proposed here is an HCI which utilizes the eye-motion tracking. This system mainly focuses for the limb disabled. It enables them to interact with the machine by looking at different positions. This system can be further modified for everyone to ease their way of interaction and for quicker communication with various systems.

1.2 THESIS OUTLINE

The work done in this thesis is structured in 8 chapters.

Chapter 1 INTRODUCTION & REVIEW

Chapter 2 LITERATURE REVIEW

This chapter describes about the various works that have done in the field of Eye-Gaze –Tracking and its application in different HCIs.

Chapter 3 SYSTEM OUTLINE

The details about how the proposed system works and a general structure of this system is given in this chapter.

Chapter 4 THEORETICAL MODELLING

Various algorithms and the software tools which the proposed made use of are described in this chapter in detail. It gives the ideas about the face & eye detection algorithms, object detection algorithms, various edge detection operators etc.

Chapter 5 SYSTEM ARCHITECTURE

Different stages of implementation of the system, the block diagram and functioning stages, details about the inputs and outputs in each stages are defined here I this chapter.

Chapter 6 EXPERIMENT RESULTS

The final results of the system, the work environment in which the results are taken, the suitable working conditions for the system to work perfectly are described here.

Chapter 7 CONCLUSION & FUTURE WORKS

In this chapter, the conclusion about the whole thesis work, the portions where the work could have been done better, the limitations and depending factors of the system's results are described. The modifications which can be done to this system, and the future scopes of the system are also explained here.

Chapter 8 REFERENCES

Chapter 2
LITERATURE REVIEW

HCI for the limb disabled is not a fresh area and I'm not the first one to work in this field. Numerous works have been done and various techniques have been proposed/introduced in the area. Most of them made use of tracking the face or eye movement as it is the next best way to interact, after the fingers. Detecting the direction of one's stare who can't move his body, is one of the better ways to communicate with others. Not only for the limb disabled, but the concept of eye motion tracking in HCI makes it easy and swift for all.

Even if various eye gaze tracking systems exist or proposed since a long time, the tracking and measure of eye behaviour and gaze direction were until recently a very complex and expensive task mainly reserved for research or military labs. The key advancements in this field been materialised by the emergence of the head-mounted eye tracker. But the uncomfortability in using this limited its application in daily life appliances.

Various other systems have proposed in the same field. C H Morimoto et al. proposed an eye-gaze tracking system [2.1] at 1999. It utilizes an economic live pupil detection technique which could be applicable to various human-machine interaction systems. The proposed system uses two Infra-Red sources, one kept on-axis and the other off-axis to the iris center. A camera captures the bright and dark pupil images formed by these sources and by thresholding the difference between these 2 frames, the pupil is localized. Then the glint is detected for determining the user's gaze. Even though it was said to be a low cost system, the total cost was more than 5000$.

Darius Miniotas proposed an HCI in which eye-tracking is used for controlling the pointer and the selection process [2.2]. It utilizes the Fitt's law [2.3] for the modelling and the detection of the gaze.

R A Colburn et al. proposed an idea of combining the eye-gaze-tracking with the computer-generated avatar mediated conversation (similar to teleconferencing) [2.4] in order to make the conversation more real. The motion of the avatar's eye is controlled with the user's gaze. This model also utilized the glint for detecting the gaze.

Takehiko Ohno developed an eye gaze HCI whose purpose is the selection of different elements in the interface [2.5]. The gaze tracking was fulfilled by utilizing an environment that consists of an EMR-NC, a non-attachment eye mark recorder, and a Sun Sparc 10 (SS10) workstation.

Dong Hyun Yoo et al. introduced a new idea for the gaze detection [2.6] which utilizes 5 IR LEDs and a camera. In this method, the image of the iris center is always inside a pentagon formed by the gleams of these light sources in the iris. This iris centre estimation was used for tracking the gaze.

Jian-Gang Wang and Eric Sung proposed an idea of user gaze estimation which developed the 'one-circle' algorithm [2.7] [2.12]. The gaze is determined by capturing the image of a single eye and detecting the iris direction.

Robert J. K. Jacob and Keith S. Karn developed an HCI model which utilizes the eye motion for controlling the interface, which aimed for the disabled or arm-occupied purposes[2.8]. Takehiko Ohno et al. proposed a gaze controlled HCI [2.9], which cooporated the blink detection along with. This proposed system utilized three cameras, in which one was having an IR array for the calibration of the user's gaze.

Arnon Amir et al. presented a hardware embedded system for eye detection, implemented only using simple logic gates and the image processing system using a CMOS

digital imaging sensor and an FPGA [2.10]. This utilized the captured frame-subtraction eye-detection technique.

Myung Jin Chung et al. in the year 2004 proposed an eye-gaze estimation system [2.11]. Unlike other systems, this system enables the large head movements, with a cost of multiple light sources, cameras, complicated computations. The glints formed by various light sources and the dark and bright pupil image capturing and analysis methods are used for the gaze estimation.

Carlos H. Morimoto et al. presented a review of different eye gaze tracking methods and their scope in using in general computer applications [2.13]. The paper gives a detailed information about different trackers like Intrusive eye gaze trackers, Camera-based eye gaze trackers, their Calibration and head motion capabilities, the Pupil–corneal reflection technique, etc.

Yuki Oyabu et al. proposed a novel eye input device [2.14] using only eye movement without the calibration for correcting the gaze direction, with the help of cursor control method using the length and direction of the eye movement vector connecting from the reference point to the center position of pupil that eliminates the calibration for the gaze direction correction and the head movement control.

P M Corcoran et al. developed a system utilizing Real-Time Face & Eye Gaze Tracking for 3-D Gaming Design [2.15]. The Viola-Jones Algorithm is used here for the tracking the face and its features. It does not require any wearable attachments, supplementary lighting, nor rely on the use of eye-glint phenomena but only employs a single user-facing camera.

Seung-Jin Baek et al. proposed an Eyeball Model-based Iris Center Localization method for Visible Image-based Eye-Gaze Tracking Systems [2.16]. Here the iris centre is done by taking in to account that its shape varies as the gaze changes. They

registered its different shapes by making a prototype of the eye ball. Then the real-time captured images are compared with these to estimate the gaze direction.

Uma Sambrekar and Dipali Ramdasi proposed an HCI utilizing the eye motion tracking, in which the keyboard type-in process and opening of different applications are controlled by the users' gaze [2.17]. It utilizes the Viola-Jones Algorithm, Hough transform, Blob analysis and Blink detection methods for the implementation of the complete system.

An HCI controlled by the user's gaze is proposed here in this thesis. This is a modified version of [2.17] having more number of keys in the interface, which can be selected and get typed by the user's head motion and gaze direction. The system is made use of the Viola-Jones Algorithm in face & eye tracking, Circular Hough Transform for the Iris Localization and the connected feature detection in binary images (Blob analysis) for the glint detection, utilizing all these pieces of information in the gaze detection. This is the most economical system implemented in this business with only the cost of a camera to afford.

Chapter 3
SYSTEM OUTLINE

The general structure of the proposed Human-Computer-Interface system is shown below in figure 3.1. The input to the system is the live captured image of the user.

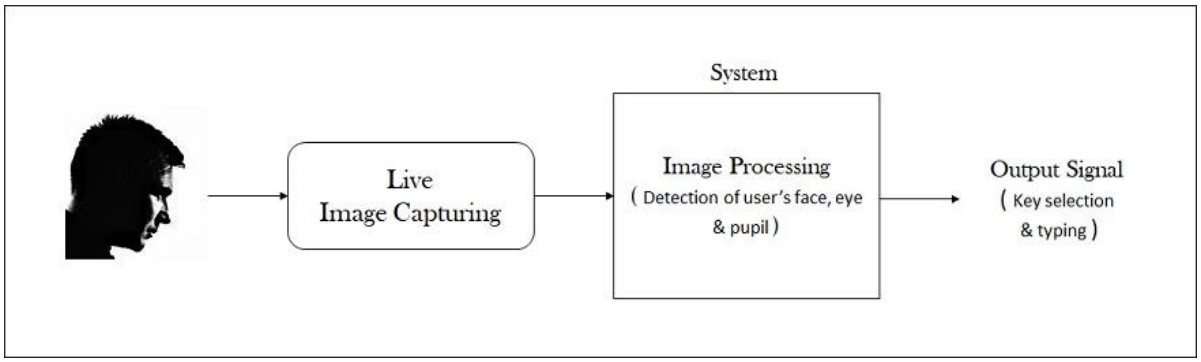

Figure 3.1. General structure of the System

The real-time image capturing of the user is done by a 720p HD webcam connected to the computer and the laptop camera, depending upon the desired outputs. This image capturing is done at a rate of 4-5 frames per second. These captured frames got transferred, using a Universal Serial Bus (USB) connection, to the system, where the image processing stage takes place.

The image-processing stage is completely implemented in MATLAB software. The output of this stage is actually the coordinates of the detected face or eye-pupil, which helps in locating the user's gaze and thereby tracing the keys which the user wants to select and get typed in.

How the system works: A virtual keypad and a textbox are available in the interface shown in the computer monitor. The user can select the desired keys in two ways. In the **first one**, the user has to move his head, and a pointer moves over the virtual keyboard according to the user's head's position. The user can select a key by placing the pointer over the desired key, by his/her head movement, and then keeping the pointer there for 5 seconds to type that letter in the text box. A timer is shown above the keypad to show the duration in

which the pointer is over a key. While the pointer moves between the keys, the timer gets reset every time.

In the **second one**, the user doesn't have to move his/her head, but to keep it steady and look at different keys to select the desired keys. While looking at a particular letter, the colour of that letter in keypad gets changed, in order to specify the selected key. The user has to stare at a letter for 5 seconds, for the letter to get typed in, same as in the first method. The iris position and the spot of the reflected light (gleam) changes as the user's direction of gaze changes, and these two positions are used to recognise the selected keys.

This is how the proposed system works. The system architecture, various algorithms and tools made use in the system implementation, the working environment details and everything else regarding this project are provided in the coming chapters.

Chapter 4
THEORETICAL MODELLING

So many algorithms have been used in realising/designing this HCI. As it discussed in the previous chapter (System Outline), this project has mainly 3 stages. **1.** Real-time video capture, **2.** Image processing & **3.** Output (the key type in). Here, in this chapter, the theory and the logics behind the various algorithms, and various software tools used in each stage are explained.

4.1 REAL TIME VIDEO CAPTURE

The live video of the user is taken using a camera connected to the system. All the operations performed by the camera, including the camera resolution, is controlled by different tools in MATLAB. There are mainly 2 tools in MATLAB, in order to obtain images or videos using a webcam.

1. `webcam` tool in the Webcam Image Acquisition Toolbox.

2. `videoinput` function in the Image Acquisition Toolbox.

Using the `snapshot` function in MATLAB, a real-time image frame from the laptop camera or any USB Video Class (UVC) compatible camera connected to the system can be obtained. For this, a camera object has to be made first in the code using the `webcam` tool and using this object, the web-cam features can be controlled. In order to capture live video or to capture many live frames as the time goes on, in the execution of a single code, the `snapshot` function can be run inside a loop (like while loop, for loop etc.,) in the code. By programming this loop we can control the frame rate of the image acquisition, and the conditions to begin and finish the image capturing can also be programmed. The additional processes to be done, if any, to each frame can easily be coded inside this loop in the program.

It is easy to save the current frame, compare the compare frame with previous frames without saving the frames using this function inside the loop method.

The `videoinput` function is used to make a video input interface, and this video object is used to manage the camera features. This is similar to the `webcam` tool explained above. The frame rate of the video capture can be set using this tool directly. As in the `webcam` tool, here also, live snaps can be taken, and using a loop in the coding, continuous capture is possible.

After acquiring live image frames, a video file (avi format) comprising of these frames can be created, if necessary, using the `addframe` function in MATLAB. The function is used to add the image frames to a video file (of format avi) using an already created an avi video input object (`aviobj`). While creating the `aviobj` the frame rate at which the video be played can also be managed.

4.2 IMAGE PROCESSING

This is the core part of this whole system. All the important algorithms used in this project are made use in this stage. These algorithms can be mainly classified into 2 sections, based on the part in which these algorithms have been utilised in the project. And those two parts are **1.** Pre-processing of the image & **2.** Tracking the Gaze Direction.

In the Pre-processing stage, the input image frames (live) are modified and corrected in order to make each frame ready/suitable for the second, Tracking stage. The output from the tracking stage is the coordinates that estimated from the tracked data from each input frame, which represent the user's direction of gaze.

4.2.1 FRAME PRE-PROCESSING

The input to this stage is the image of the user, which is taken by the camera. This captured image is an RGB frame. The RGB frame consists of the colours, which are

actually acquired by the controlled mixing of the three primary colours: red, green and blue. In RGB modelling, each colour has its own RGB value. For example, the value of red colour is 255.00.00 in decimal representation. The key objective of this RGB model is to distinguish, interpret and to show various pictures and frames in electronic devices. Thus, the RGB value really represents the pixel value of each pixel in the display of these electronic devices. As the brightness, contrast, sharpness and similar other features of the display monitor of different electronic devices varies, the same RGB value represents different colour for different electronic devices or even the same system over time. That is, the RGB model is machine-dependent.

4.2.1.1 GRAYSCALE

A grayscale image is comprised of the pixels having different shades of gray. It is similar to the black n white representation of an RGB image. Each pixel in the gray scale represents the brightness information of that pixel. That is, the pixel having the greatest intensity is represented in white, while the zero intensity will be a black pixel. Each pixel in grayscale is bearing an 8-bit information equivalent to its luminance. This is the grayscale value of the pixel and it ranges from 0 to 255, in which 0 is black and 255 represents the white pixels, all the other shades of gray have the values between these 0 and 255. Even though the grayscale image looks similar to the black and white images, they are not the same. In black and white images, all the pixels which are not white, are black. As there are only 2 colour tones in black and white images, it can be represented in 2 bits, that is, each pixel in it is having a 2-bit value, while to represent 8-bits are needed for grayscale pixels. The figure below distinguishes between these.

In the grayscale of the first image, the outer border and the inner bird seem to be having the same shade of gray, even though they are totally different colours in the RGB image. The grayscale gives the information about the luminance of each pixel. That is, if two different colours have the same brightness, they both look similar in their corresponding grayscales.

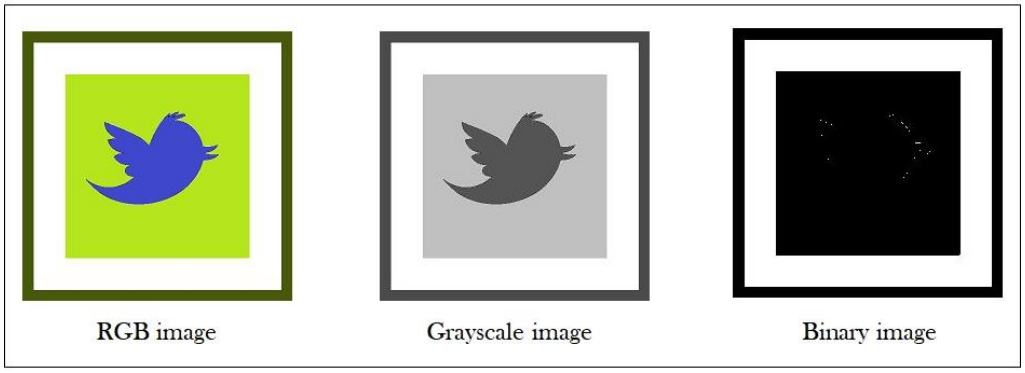

Figure 4.1. RGB – Grayscale – Binary conversion

RGB to Grayscale conversion: While the conversion, the pixels lose the data about colours, and only the intensity information is left. Each pixel in RGB image is comprised of red, green, and blue colours. That is, every pixel have 3 discrete intensity values, each representing each primary. Figure 3.3 shows the 3 different colour channel images and their corresponding grayscale images of the colour image.

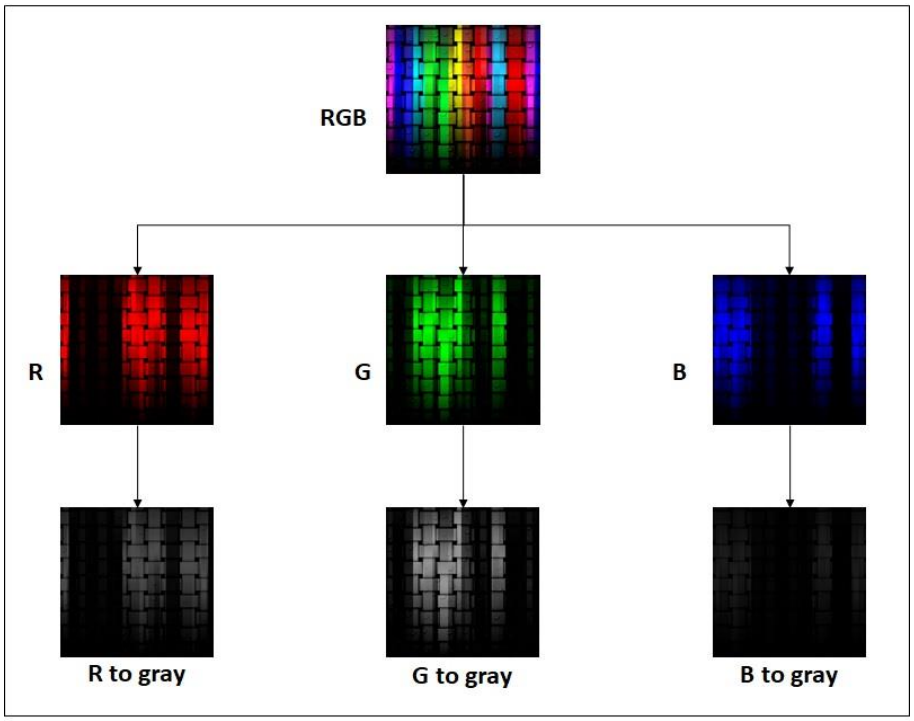

Figure 4.2. Channel splitting of RGB

Thus, a single RGB image has 3 discrete grayscale values. These 3 discrete intensity information have to be mixed to form a single value, while the conversion from RGB to Grayscale. One simple method is to take the average of the three. But, this way is not

preferable, as the sensitivity of our eyesight to different colours are different. So, most of the programs use a weighted average of these 3 values to find the final grayscale value of each pixel [4.1] and these weightages should add up to one [4.1],[4.2].

$$\begin{aligned} G \quad &= \quad \omega_R L_R + \omega_G L_G + \omega_B L_B \\ given \quad &\quad \omega_R + \omega_G + \omega_B = 1, \\ &\quad \omega_R \geq 0,\ \omega_G \geq 0,\ \omega_B \geq 0, \end{aligned} \tag{1}$$

where $\omega_R, \omega_G, \omega_B$ are weightage coefficients of primary colour channels and $L_R,\ L_G,\ L_B$ are the luminance components of the primary channels. The values of the weighting components, in MATLAB while using `rgbtogray()` [4.2] are

$$\omega_R = \frac{7.47}{25},\ \omega_G = \frac{14.67}{25},\ \omega_B = \frac{2.86}{25} \tag{2}$$

Normally gray to RGB conversion is not possible. But, with the grayscale values of all the three primary channels, it is possible to retrieve the RGB image from the Grayscale image.

4.2.1.2 RGB to BINARY CONVERSION

Usually, grayscale images are known as Black and white images, but that name best suits here, for the binary images. Each pixel can possess a maximum of 2 values, representing either a bright or a dark pixel. A binary image can be defined as a modified version of grayscale. Suppose, a threshold value is assigned for the grayscale values of a grayscale image. By representing all the pixels having grayscale values more than the threshold as white and the others as black, or vice versa, it is the binary equivalent of the image [4.3]. Every pixel in grayscale is having an 8-bit data, likewise, in binary, each pixel possess a 1-bit data, either zero or one. Also, the binary of an image is not unique. By changing the threshold, discrete binary images of the same image is possible. Figure 3.4 shows the binary images, having 3 different threshold values, of a grayscale image.

It is not possible to retrieve the grayscale image from the binary image usually. But, as in the grayscale, it is partially possible to retrieve if the binary images with various thresholds are available.

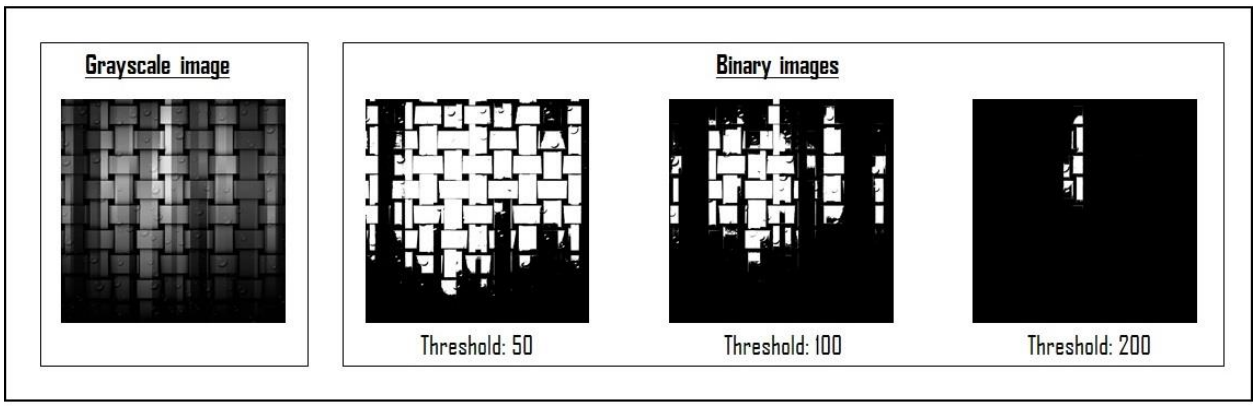

Figure 4.3. Gray to Binary conversion

The main purpose of the binary conversion is to distinguish different objects in the image from the background by utilising the intensity variations in different surfaces. By making use of edge-detection algorithm on a binary image, the shape of different objects in the image can be possibly detected. Various Edge-detection algorithms convert the rgb image to binary first for better results. In order to detect the gaze direction, the gleam inside the iris should be detected, which is done by the use of the binary image of the iris.

4.2.1.3 GAUSSIAN SMOOTHING / GAUSSIAN FILTERING

The filtering of an image results in another image with some modifications in the original image. Compared to the first image, either some features may be added or removed or both, in the resulting image. Various filters are used for improving different features like sharpness, contrast, noise removal etc. of an image. As the name indicates, Gaussian Filtering is a kind of filtering, which uses the possibilities of Gaussian function in modifying an image. It is used to remove/control the noise content in the image. So, in a way, this filtering lowers the image details. It simply blurs an image, which allows only the low-frequency contents. Thus this is a non-uniform linear low-pass filter. The Gaussian function in 1-D is:

$$G(a) = \frac{1}{\sqrt{2\pi d^2}}\, e^{-\frac{a^2}{2d^2}} \qquad (3)$$

and in 2-D

$$G(a,b) = \frac{1}{\sqrt{2\pi d^2}}\, e^{-\frac{a^2+b^2}{2d^2}} \qquad (4)$$

where a, b : distance from the origin in x and y-axis
d : standard deviation of the Gaussian distribution.

A Gaussian distribution with mean 0 and d = 1.2 is given in figure 3.5:

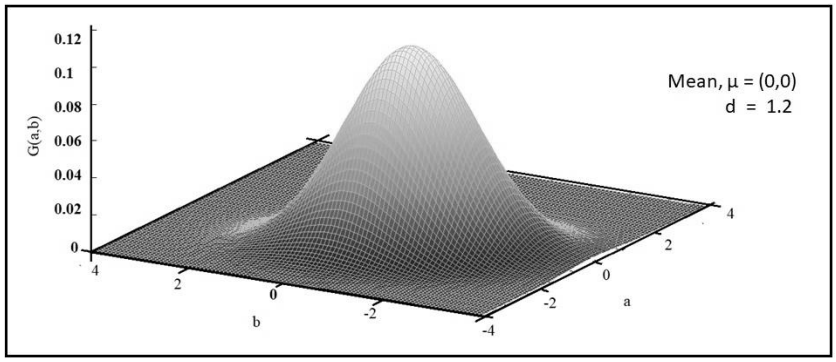

Figure 4.4. Gaussian distribution

The 2-D representation of this equation results is a plane of concentric circles having a normal distribution from the centre. The Gaussian filtering utilises this distribution in forming a kernel matrix, which is convolved with the original image to form the modified blurred image. The figure below shows a kernel matrix approximation of the normal distribution having d = 1,

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figure 4.5. Gaussian filter kernel matrix with d = 1

From the kernel, it is clear that the filtering results in an image whose pixel values are the weighted mean of the 25 neighbouring pixels in the original image. The pixel

in which the convolution is done has the highest weight and as the distance from this pixel increases, the weightage is further reducing. Figure 3.7 shows the Gaussian filtered results, with two different standard deviations.

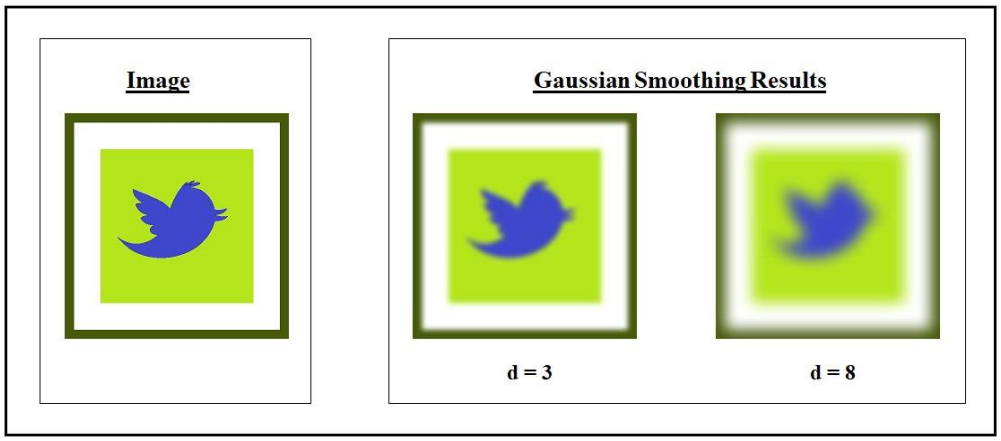

Figure 4.6. Gaussian filtering with d=3 and d =8

4.2.2 TRACKING

The input to this stage is the preprocessed image of the live user and the output of this stage is the detected gaze direction of the user, in a format suitable for the next stage, where the type-in process takes place. The tracking stage is done in three stages: **1.** Face Detection, **2.** Eye Detection **3.** Pupil Detection & **4.** Gleam Detection. The algorithms those have utilised in each stage are defined below.

4.2.2.1 FACE DETECTION

Face Detection is always an interesting and not-so-easy task from long back itself. Many theories and algorithms have been proposed through all these years. Some could detect the face from an image, some could do it from videos etc... According to Ming-Hsuan Yang *et al.* [4.4][4.5], different techniques in detecting the face from an image are categorised into 4,

1. **Knowledge Based.** The approaches come under this follows certain rules in the face detection. These rules utilise the usual concepts of different parts and the features the face comprises of. The rules are simply **1.** The eye detection rules, **2.** The nostrils

detection rules & **3.** The mouth detection rules [4.6]. Here, it searches for the basic face components like eyes, nose, mouth etc. plus, it eliminates the portion in the image for detection where the mouth part positioned near to eyes than nose, or the nose comes inside the mouth etc…

2. **Feature-Consistent.** This makes use of certain consistent structural characteristics that the faces possess, those persist even when the environment, posture, brightness etc. changes. Certain classifiers are trained with these consistent features in this method. These help in distinguishing the face area from the non-facial areas.
3. **Template Matching.** Various conventional facial prototypes are retained here for the detection of the face or certain characteristics. The correlation of these prototypes and the input data are calculated for the efficient identification process.
4. **Appearance-Based.** In this method also, several prototypes are used for the detection. But these prototypes are not predefined, but the method gets trained itself from the sample face models.

Most of the above described detection methods, detects from an input image and their applications in the real-time detection are limited. The algorithm suggested by Paul Viola & Michael Jones [4.7] [4.8], is considered as the basic fundamental detection method, which delivered the first ever substantial results in live face- detection [4.9].

4.2.2.1.1 VIOLA-JONES ALGORITHM

The detection rate of the Viola-Jones algorithm is so high, such that it can be used for object/face detection in real-time. And that is the main focus of this algorithm over the others on the same job. This framework introduced 3 main concepts **1.** Integral Image, **2.** Adaboost based Training, & **3.** Classifier Cascading

4.2.2.1.1.1 Integral Image

An integral image at any point can be formed as the summation of pixels above and to the left of that point (pixel) including that point. The integral image at (a,b) can be

found out [4.7] [4.8] by

$$ii(a,b) = \sum_{a' \le a,\ b' \le b} i(a',b') \quad (5)$$

where, $ii(a,b)$:integral image

$i(a,b)$:original image

and by making use of the set of recurrences given below, this can be calculated in a single pass across the primary image.

$$s(a,b) = s(a,b-1) + i(a,b) \quad (6)$$

$$ii(a,b) = ii(a-1,b) + s(a,b) \quad (7)$$

where $s(a,b)$: cumulative row sum

$s(a,-1) = 0$

$ii(-1,b) = 0.$

The figure 4.7 shows an example for finding the integral image of the input image.

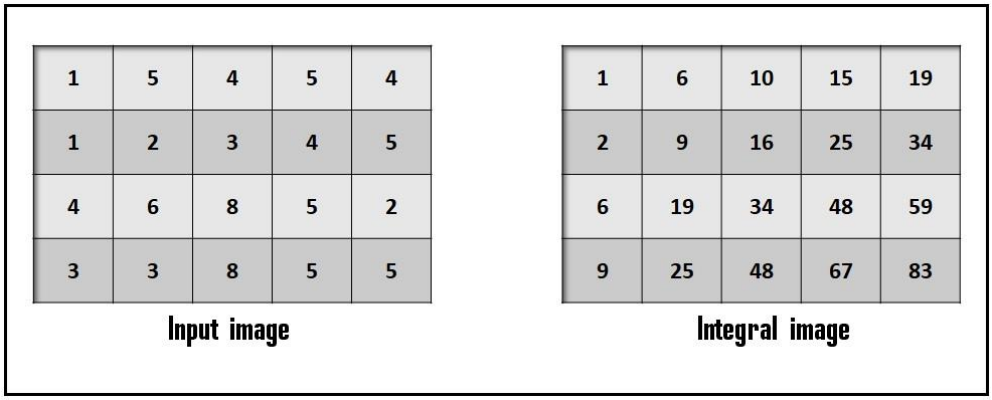

Figure 4.7. Integral image computation

Here, the first image is the input image and the second one is its corresponding integral image (showing the pixel values). The value 67 at the point (4,4) in the integral image is obtained by adding the pixel values (1+5+4+5+1+2+3+ 4+4+6+8+5+3+3+8+5) above and left to that point in the input image. In order to understand the importance of the integral image, the process of detection in this algorithm should be known.

Instead of using pixels, this algorithm makes use of the value of simple features for the detection purpose. Compared to the other method for training, this method's aim is to cut the intra-class alteration while raising the inter-class variability [4.10] and thereby shaping

a simpler classification. The features used here are similar to the Haar wavelets [4.11]. This algorithm utilizes mainly 3 such features for the training and detection (Figure 4.8).

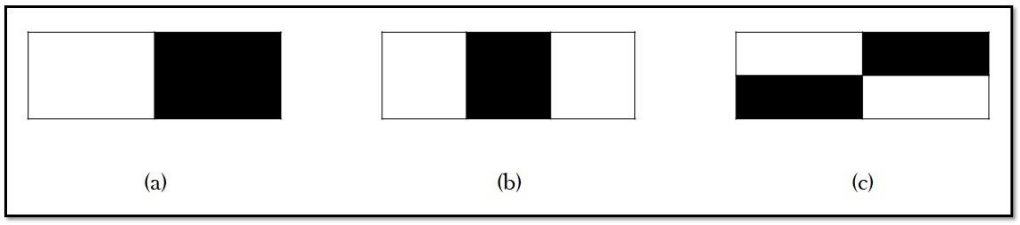

Figure 4.8. Haar-Like Patterns

For each feature given in the figure, the value is calculated as,

(a) Feature value = $(W_p - B_p)$ (8)

(b) $(W_{1p} + W_{2p} - B_p)$ (9)

(c) $((W_{1p} + W_{2p})-(B_{1p} + B_{2p}))$ (10)

where W_p : pixels under white rectangular box
B_p : pixels under black rectangular box

These features in different resolutions are placed over the input images (fig 4.9), to find the corresponding feature values. The value is calculated by finding the difference between the sums of the pixels under the white cell and that of the black cell. Other pixels aren't used in calculating the feature.

Figure 4.9. Haar-Like Pattern calculation

The summation of a set of pixels comes under a rectangular region has to be found out in order to make use of the features, in the algorithm and as the feature resolution is variable, the number of terms in each summation are not fixed. It can be as small as a pixel

as well as a huge number. Therefore the time taken for this calculation won't be a constant, but varies with the feature and the input image, if we are using the input image directly. This will affect the total detection time also. The integral image is introduced to handle this situation. The Figure 4.10 below shows how the integral image makes the detection time a constant.

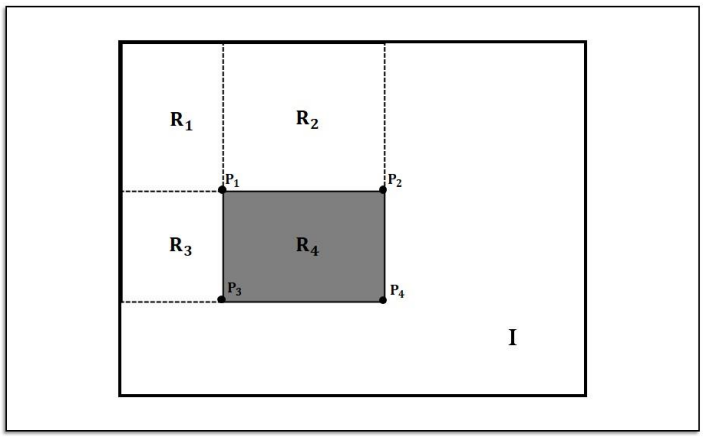

Figure 4.10. Feature calculation using integral image

Suppose I is an integral image, and the feature value of the rectangular region R_4 of the input image has to be calculated. The value at point P_1 in I is the sum of the pixels in R_1. Similarly, at P_2 is $(R_1 + R_2)$, P_3 is $(R_1 + R_3)$ & P_4 is $(R_1 + R_2 + R_3 + R_4)$. So, the final result is $(P_1 + P_4)$ - $(P_2 + P_3)$. That is, irrespective of the number of pixels under the feature, the number of terms in calculating the feature value is always 4, while using the integral image.

4.2.2.1.1.2 Classifier Training

On detecting the object in an image, the base window size is 24x24 pixels. That is, the detection starts with a 24x24 portion of the input image and as the detection goes on, the window of the same size pass over the complete image. So, for each sub window, over 160,000 features has to be calculated. This is much higher than the total number of pixels, which makes the detection efficient but much time-consuming.

So, here, instead of taking all the features, an efficient classifier is generated with less number of features, and used the Adaboost Algorithm for feature selection [4.12]

[4.13]. This is a weak learning algorithm. This picks a feature which can distinguish among the given image models effectively compared to the others, then boost its thresholds in further iterations of training. Thus for detection of different objects, several separate weak features are selected. As shown in the figure 4.9, two separate features for eyes, and nose on the sub windows are used for the detection of face.

4.2.2.1.1.3 Classifier Cascading

These selected weak features are then cascaded to form a robust classifier as given in the figure 4.11 [4.7] [4.8]. Each boosted weak classifiers in each stages have individual features with separate thresholds, such that, a positive result on a weak classifier may be a negative while passing through another. The classifier in each stage gets activated only when the previous classifier has a positive result. If the result of a particular classifier isn't positive, that sub-frame will be immediately discarded.

The figure 4.11 shows a graphical representation of the final detection. Here, the weak classifiers are numbered 1 to 4. The rejection of the sub window occurs, whenever the threshold selection criteria of any of the 1 to 4 classifiers is not positive. In other words, for a sub window to be detected positive, all the classifier result should be positive.

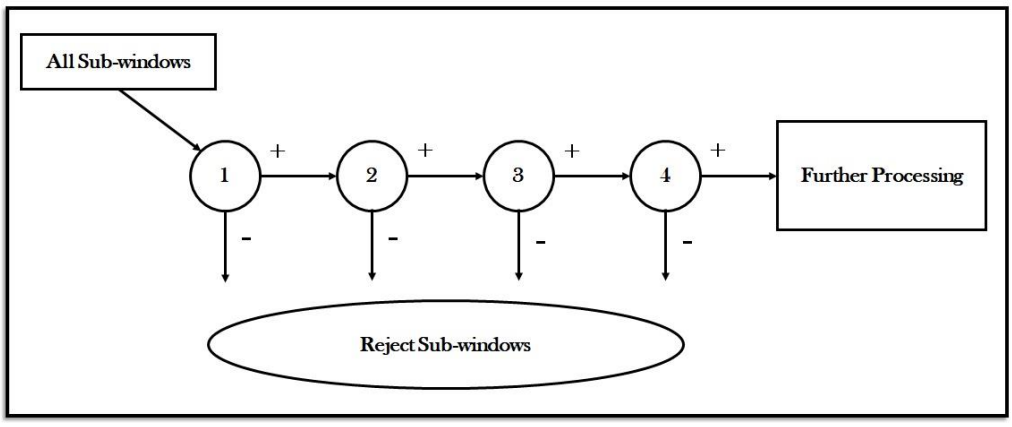

Figure 4.11. Classifier Cascaded detection

Each classifier will give best results, when the sub window has the similar pixel luminance with that of the feature associated with it. This is demonstrated in the figure 4.12

below. The first row shows 2 classifiers, having similar but not the same features. Among these,

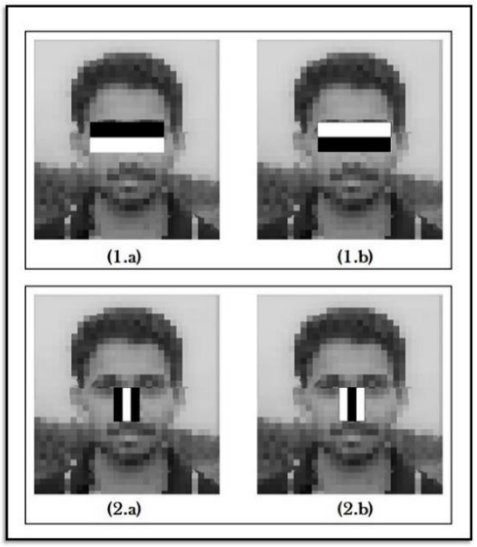

Figure 4.12. Selsction of features

the first classifier result will be positive, while the other result will be negative. The reason is, both the classifiers are trying to detect eye in the sub window. The first classifiers feature has the similar pixel luminance as of that part of the image. It has black rectangle over the eye portion and the white rectangle over the cheeks. So the feature value of the first classifier will be higher compared to the second one, which having the feature not matching with the image. In a similar way, the classifier in 2.a results in positive result, while 2.b results in negative.

4.2.2.2 EYE DETECTION

This part is similar to the previous face-detection part. Here the main aim is to detect the eyes of the user. After the detection, it has to modify the output image so as it suits for the next iris detection. The different algorithms used are as follows.

4.2.2.2.1 VIOLA-JONES ALGORITHM

The eye-detection also comes under the object-detection image processing. As in face-detection, Viola-Jones Algorithm is used for the eye-detection also. Here, the features used to detect the face and the eye are not the same. The features given in fig 4.8 are not

sufficient for the perfect detection of the eyes. Hence, in the same Viola-Jones Algorithm, for the detection of eyes, another set of features [4.14] (Fig 4.13) are utilized. These features are similar to the previous features, but best matching with the human eye structure.

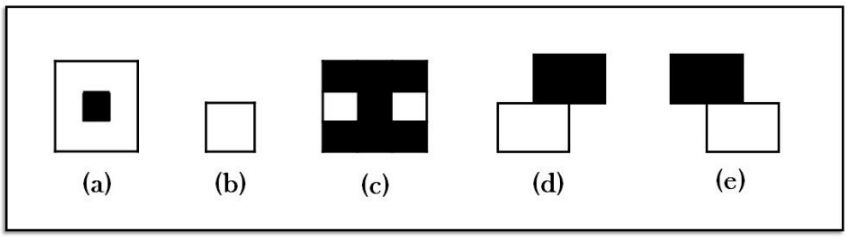

Figure 4.13. Haar-like patterns for Eye-Detection

All the other parts in the eye detection is as same as in the face detection. That is, as per this algorithm, it has to select the classifiers, boost them, cascade the classifiers according to their defined thresholds, pass the image sub-windows through these classifiers and those sub windows don't fail in any of the cascaded classifiers are detected as the eye portion in the image.

4.2.2.2.2 REGION OF INTEREST

In Image-Processing, the Region of Interest (RoI) is a part of the input data, in which further processes has to be executed, while the remaining part is of no interest in the process. It can be defined as a subset of the original data. The application of ROI improves the execution time in coding, by avoiding the processing of the unnecessary data. For this elimination process, a binary-mask is first generated. The dimensions of this mask should match with the dimensions of the original input image. As this mask is binary, it has only 2 pixel values 1 & 0. The region corresponding to the part of the image to be eliminated from processing are given 0s, and others as 1s.

And here the ROI is used in the eye-detection of the user. The input date here is the captured image and the Region of Interest is the detected sub window, from the previous Face-detection step (Figure 4.14). In other words, the code has to search for the eyes only in the face.

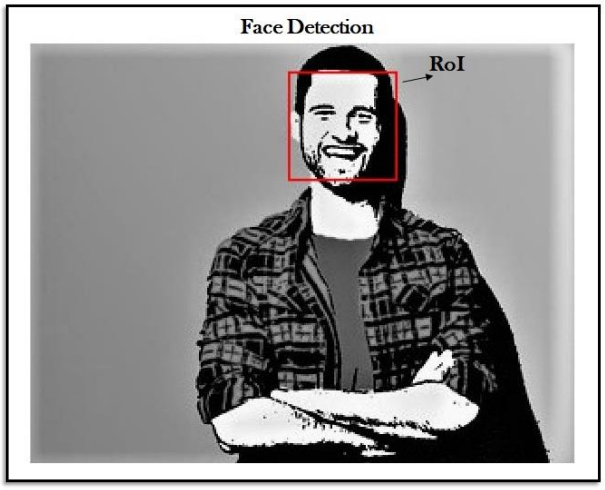

Figure 4.14. An example for ROI

If the Region of Interest is in rectangular shape, the processing part is comparatively easy. But when it comes to arbitrary shapes are not as simple as what is given above. Several methods for dealing with random shaped ROI [4.15] [4.16] [4.17] has been proposed later.

4.2.2.2.3 POINT TRACKER OBJECT

In MATLAB, this object traces a group of points in a frame. It is done by utilizing the Kanade Lucas Tomasi (KLT) feature tracking algorithm [4.18] [4.19]. This is made use in the object tracking process. It tracks the group of points those having similar characteristics like luminance, pixel values etc. among the continuous frames. And by tracking such points, it helps in tracking similar objects. The algorithm utilizes the spatial intensity variance of each frames in order to locate the equivalent pair of points among the frames [4.20]. As the time goes on while tracking, some tracked features may be lost, since brightness change, or camera instability etc. In such cases, in order to track features on lengthy videos, or live videos, these features has to be regained continuously.

4.2.2.2.3.1 Maximum Bi-Directional Error

This is one of the properties of the point tracker. While initializing the tracker, we have to provide this value. This is the maximum directional error threshold in either

directions, greater than which the tracker lose the points between the frames. That is, for tracking a point in continuous frames, the displacement between the same points in the continuous frames should be less than this error.

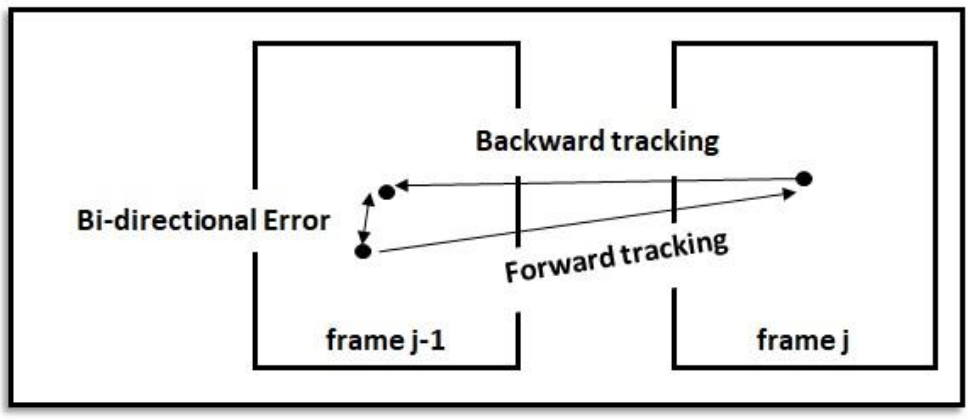

Figure 4.15. Bi-Directional Error [4.21]

The algorithm tracks back the point in the current frame to the previous frame, to compute the error, and if the error is less than the predefined, the tracking of that point continuous to the next frame.

4.2.2.2.3.2 Corner Points

Finding the corner points of an object is as important as finding the feature points in a frame using the KLT Algorithm. The main reason why, the points tracking is used in the face/eye detection is, detecting face/eye using Viola-Jones Algorithm in every frame in real-time (with a frame rate of >5 frames per second) is more time consuming than finding feature points using the KLT algorithm. It's not like Viola-Jones is slow, but KLT is faster. So an alternative way for using face detection in every frame is, to use the Viola-Jones Algorithm in the first frame. Then use KLT for finding the feature points, and the corner points of the detected face region, and then track these points in the subsequent frames. Whenever the number of points tracked go below 10, that is, it can't track the points effectively in the next frame, use face-detection again in the next frame and then use point tracker in successive frames.

The corner points of the detected face regions (RoI) is detected using the `detectMinEigenFeatures` function in MATLAB. It generates an object that contains the corner points of the RoI. This function makes use of the minimum eigenvalue algorithm proposed by Jianbo and Carlo in 1994 [4.18].

4.2.2.2.3.3 Matched Points' Transformation

Suppose a point in a frame is matched with another point in the following frame, and this point has to be highlighted or shown in these frames. If the points are not on the same location on both frames, may be because of a camera tilt or something, it doesn't seem like the points are matched/same. In order to make it feel that both the points are same in these frames, there should be a smooth transform within these points between the frames. `estimateGeometricTransform` function (Computer Vision System Toolbox) in MATLAB is used for this. This function generates a 2-D transform object. This helps in mapping the matched points in the 2 frames [4.22] [4.23].

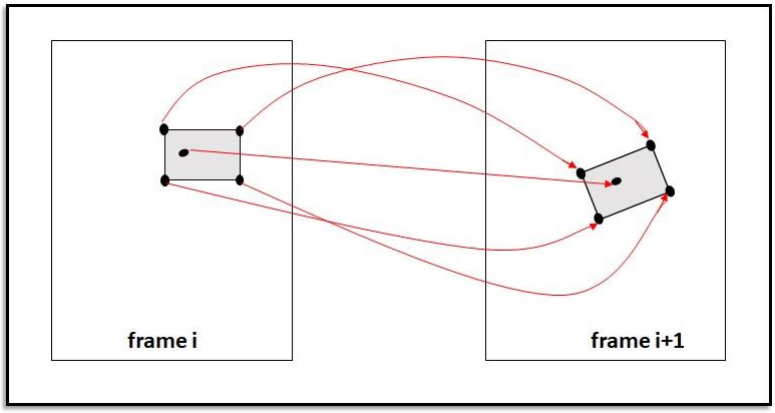

Figure 4.16. Matched points

If any points inside RoI in the first frame got out of the RoI in the next frame, then transformation for those points are not generated. That is, the transformation are generated only for the inlier points, but not the outliers. `transformPointsForward` function [4.24] in MATLAB applies the transformation generated by the `estimateGeometricTransform` function, between the 2 points.

4.2.2.3 IRIS DETECTION

Iris is the dark circular region in the human eye, and the pupil is the darker small circle in the middle of the iris, as shown in the figure 4.17.

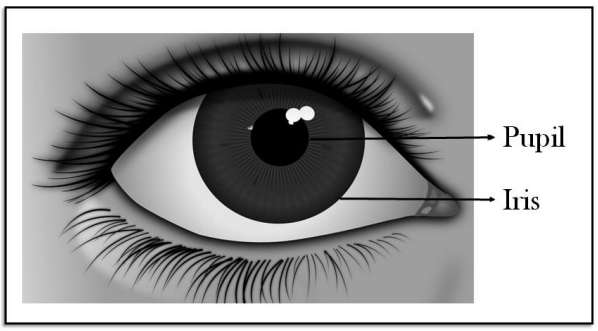

Figure 4.17. Iris

As the user looks at different directions, the part of the eye that moves along with that is the iris along with the pupil. So after the detection of the user's eye the system has to locate the iris and record its movement in order to find the direction to which the user stares. The easiest way to locate the iris is to use an algorithm to detect the circular shaped objects in an image. One of the best algorithm to do this job is the Hough Transform Algorithm. The input to the Hough transform algorithm is the edge detected image of the user's eye.

4.2.2.3.1 EDGE DETECTION

The best way to find the shape of the objects present in an image is to find the objects' boundaries in the image. There will be a discontinuity in the object boundaries with respect to the surroundings. These discontinuities are imprinted in the image as intense intensity variations in the neighboring pixels. So, in order to detect the object boundaries, it is to find the abrupt alterations in the luminance content of the pixels in the image. Not only the edges, but the important properties like colour change, lighting information etc. can also be detected using the edge detection.

In Image processing, different filters are there to modify or alter any of the image characteristics. Usually, it is done by altering the image pixel values with the weighted

sum of the corresponding pixels and its surroundings. In another way, it is the convolution of each pixels with specific matrices for specific purposes. And these matrix are called kernels. For each image filtering processes, various kernels have been proposed over time.

The major types of edge detection methods, in image processing are [4.25] are

1. Robert, **2.** Prewitt, **3.** Sobel & **4.** Canny.

4.2.2.3.1.1 Robert Edge Detection

This is a first order gradient operator, which compares the difference gradient with the threshold in finding the edges. This was first proposed by L G Roberts in 1963. This operator has 2 kernels, each for horizontal and vertical detection track. The horizontal and vertical masks are given in figure 4.18. After finding the derivative approximates in each directions, the gradient magnitudes are computed for comparison with the threshold.

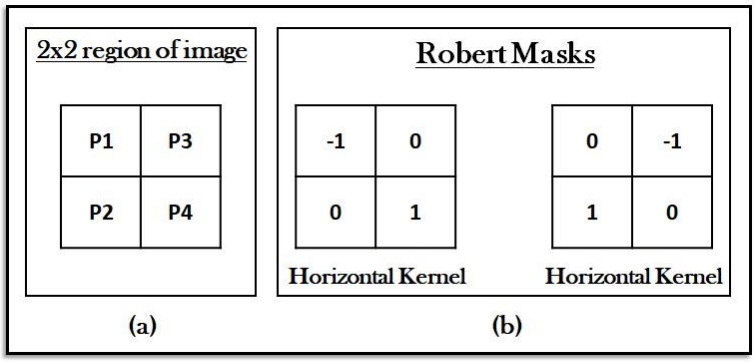

Figure 4.18. Robert Kernels

Upon the application of these operators on each pixels, the derivative approximates on either directions are given by,

$$g_x = (P_4 - P_1) \quad (11)$$

$$g_y = (P_3 - P_2) \quad (12)$$

where P_1 is the pixel where the operators are applied

P_2 to P_4 are neighbouring pixels

g_x & g_y are horizontal & vertical derivative approximates

And the gradient magnitude, |G| is found out by,

$$|G| = \sqrt{{g_x}^2 + {g_y}^2} \quad (13)$$

This |G| is used to determine whether the pixel is a part of the edge by comparing it with a predefined threshold.

4.2.2.3.1.2 Prewitt Edge Detection

Tis is also a kind of discrete gradient operator method for edge detection. In Robert detection, the kernels were of 2x2 dimension, while here it is 3x3. The Prewitt masks are given below (Fig 4.19).

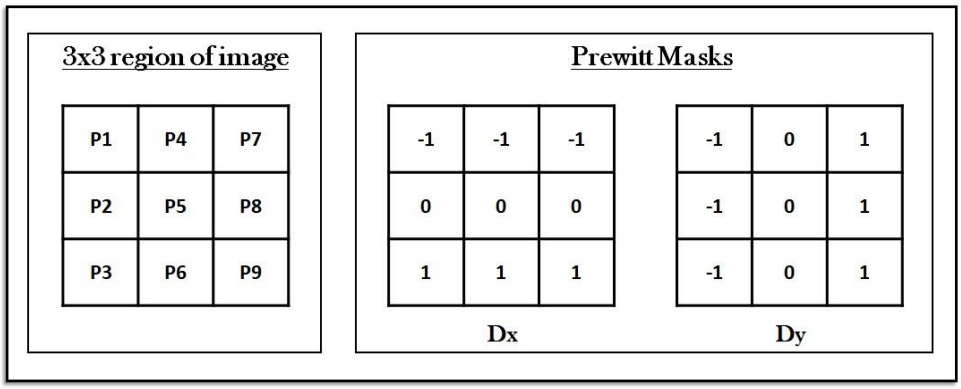

Figure 4.19. Prewitt Kernels

$$g_x = ((P_3 + P_6 + P_9) - (P_1 + P_4 + P_7)) \quad (14)$$

$$g_y = ((P_7 + P_8 + P_9) - (P_1 + P_2 + P_3)) \quad (15)$$

where P_5 is the pixel which operators are applied

The remaining steps are same as in the Robert detection, to find the gradient magnitude and the thresholding.

4.2.2.3.1.3 Sobel Edge Detection

The kernels used here are similar to that of the 3x3 Prewitt kernels. But here, the masks are modelled such that, the pixels in the boundary regions are provided with higher weightage than the neighbouring ones.

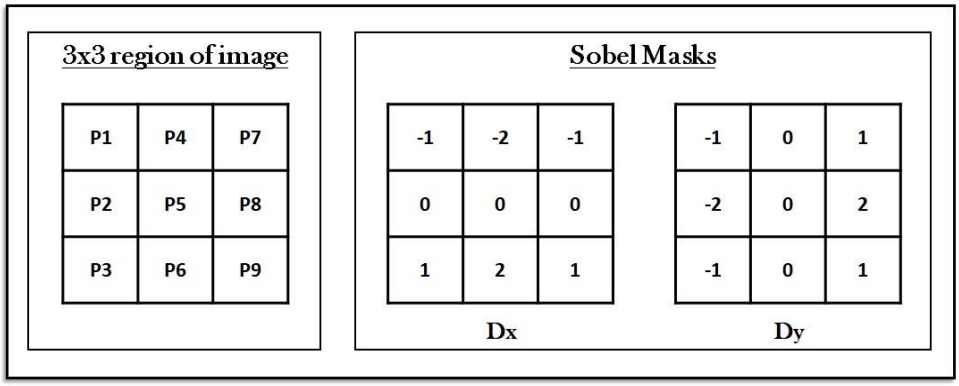

Figure 4.20. Sobel Kernels

$$g_x = ((P_3 + 2P_6 + P_9) - (P_1 + 2P_4 + P_7)) \quad (16)$$

$$g_y = ((P_7 + 2P_8 + P_9) - (P_1 + 2P_2 + P_3)) \quad (17)$$

The above discussed 3 Edge-Detection Methods are Gradient based detection methods. The main problem associated with this method is, they are more susceptible to noise [4.26]. The operator masks and its elements are fixed and it can't be modified according to the input.

4.2.2.3.1.4 Canny Edge Detection

Unlike the methods described above, Canny Edge Detection is less sensitive to the noise contents in the image. This method was introduced by John Canny in the year 1986 [4.27]. through his paper, he boosted the different detection methods present then. A set of standards were kept by him while doing this. The important one is to curtail the error rate. That is, no omission of edges, and non-inclusion of non-edges. The next one is to make the detected edges and the real edges in the image, close to each other. And the last one is not to detect the same edge twice [4.28].

Canny comes under Gaussian Edge Detection, because of the above set of standards. The various stages in the detection are [4.26].

1. Generate an appropriate kernel for the execution of Gaussian smoothing in the input image. This kernel is convolved with the image. The sensitivity-to-noise minimizes with the size of the kernel.
2. Using the Sobel kernels (Fig 4.20), compute both the derivative approximates (g_x & g_y) and then the edge strength |G|.
3. Using g_x & g_y, find the direction of the edge, Θ.

$$\theta = \begin{cases} tan^{-1}\left(\frac{g_y}{g_x}\right) & if\ g_x \neq 0 \\ 0 & if\ g_x = g_y = 0 \\ 90^o & if\ g_x = 0, g_y \neq 0 \end{cases} \quad (18)$$

4. After finding θ, compare it with a predefined range of angles shown in fig 4.21. That is, if $0 \leq \theta \leq 22.5^o\ or\ 157.5^o \leq \theta \leq 180^o$, then θ is assigned to 0^o, or, if $22.5^o \leq \theta \leq 67.5^o$, then $\theta = 45^o$, or if $67.5^o \leq \theta \leq 112.5^o$, then $\theta = 90^o$, or if $112.5^o \leq \theta \leq 157.5^o$, then $\theta = 180^o$.

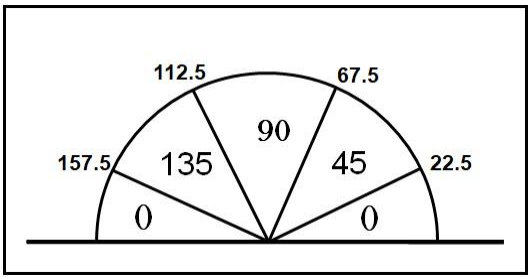

Figure 4.21. Direction Assignment

5. Next is non-maximal suppression. Here, it follows the edge pixels in its direction, and reset those which are not a part of the edge, to zero. This narrows the edge thickness.
6. Double thresholding is applied. The pixels having value greater than the first threshold are marked as edge and any pixels attached to this marked pixel having a value greater than the second threshold are also considered as a part of the edge in the image outcome.

The Canny Edge Detection has the advantage of better SNR than the other methods. It is flexible compared to other methods, as it can be applied to various noise intensities. Unlike the others, the constraints and the thresholds of Canny are adaptable to various situations.

4.2.2.3.2 HOUGH TRANSFORM

This algorithm is used here to locate the iris position in the detected eye image. The input to this stage is the edge detected image. Hough transform helps in spotting different curves in the image and here it is used to locate the circle having radius in a predefined range. This algorithm detects the user-specified curves, even if there is a break in the curve or the curve is not even complete [4.29].

For the detection of any curve, its attributes should be predefined to the algorithm. So the equation of the circle and the radius (or radius range) are provided.

$$(x - a_0)^2 + (y - b_0)^2 = r^2, \tag{19}$$

which describes the circle having the points $x \; and \; y$ with centre $(a_0 \, , b_0)$ and having the radius r. Some detected edges in the inputs constitute these $x \; and \; y$. All the input image edges are mapped to a 3-D space known as accumulator space, or Hough space [4.31]. This is a parameter space [4.30] with parameters (a_0, b_0, r). If an edge point is a part of a circle, then the locus for the parameters of that circle is a right circular cone [4.32]. The figure 4.22 shows the mapping of the edge pixels to the 3D-parametric space, *A*. In fig 4.22 (c), the cones representing different edge points transect each other at a point. The base circles of the cones at that point are known as circles of votes. Then those edge points in the image is a part of a circle. The parameter r in the Hough space, *A*, at the point of intersection is the radius, and the parameters (a_0, b_0) is the centre point of that circle. There may be more than one r=constant plane in this 3d space where the cones intersect.

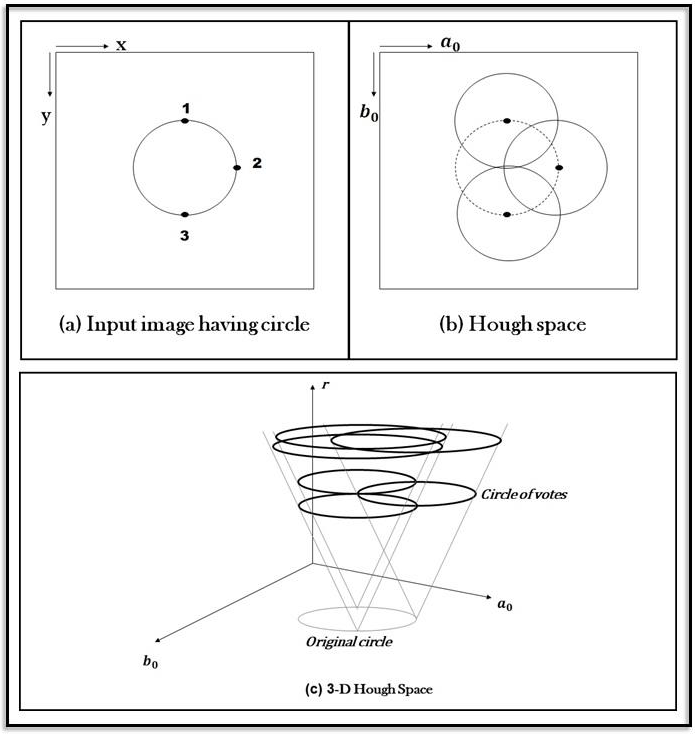

Figure 4.22. Hough Transform

The figure 4.22 (b) shows the circles of votes interpolated with the circle to be detected from the input image. The algorithm chose the parameter having max(A) (maximum number of votes in the space A) as the detected circle's parameters [4.33], if the parameter *r* is in the range of radius specified by the user.

4.2.2.4 GLEAM/GLINT DETECTION

The gleam is the white portion visible inside the iris of the human eye, as shown in fig 4.23. This portion is formed due to the small part of the incident light gets reflected by the iris. This gleam plays a vital role in locating the users gaze direction. The detection method used in this stage is Blob Analysis.

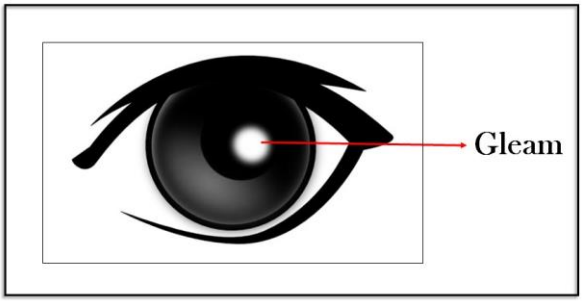

Figure 4.23. Gleam/Glint

4.2.2.4.1 BLOB DETECTION

The input to the blob detection stage is a binary image. That is, the input has only two pixel values, either 0 or 1. Now, what is a blob? It is a region with at least one local extremum and extend until it merges with another blob [4.34]. Thus the best way in detecting the blob is to use a binary image.

This method finds the portion in the image having the most number of connected pixels having the same value, usually 1. That is, the output of this method has the information about the biggest white portion (extremum with biggest spread) in the binary input. It first detects a local maxima (or minima) in the input image. Then it extend its searching scope to all the eight neighboring pixels and goes on. It stops its search when all the connected pixels are 0s (or 1s). The figure 4.24 shows an example of the blob searching algorithm. Suppose it is the input binary images with pixels 0 and 1. Let the pixel (3,2) is detected first (white). Then as per the algorithm, the searching extends to its 8 neighbouring pixels ((2,1),(2,2),(2,3),(3,1),(3,3),(4,1),(4,2),(4,3)), where 6 of them are again detected as 1. Then the searching spreads to the 8 neighbouring pixels of each of these 6 pixels. Only for the pixel (4,3) again a white neighbouring pixel (5,3) is detected. As this pixel has no other connected white pixel, the searching process for this blob stops there and it searches for another blob. And the searching starts again on the pixel (3,5) and goes on.

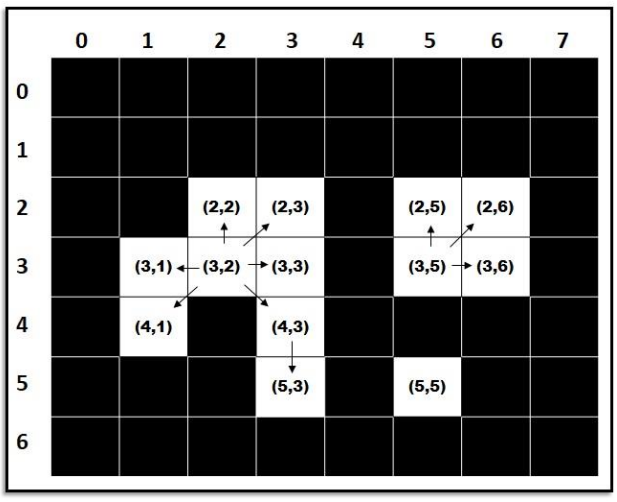

Figure 4.24. Blob Analysis

The centroid coordinates and the area of each blob or the biggest blob are the outcome of this detection method. That is, the blob analysis/detection method gives the parameters of the brightest parts in an image.

So far, all the algorithms and tools used in this project and the logics and theories behind them have been explained in this chapter. Now, the different stages where these algorithms are made use of, and the outcome of each stages will be discussed in the next chapter.

Chapter 5
SYSTEM ARCHITECTURE

The detailed functioning of the system proposed has already discussed in the chapter 3. Here, the complete system architecture of the different stages and the input output features of each stages are discussed. As defined in the third chapter, this project has mainly 3 stages. **1. The input image capturing stage, 2. The image processing stage & 3. The output key selection stage.**

5.1 THE INPUT STAGE

As discussed earlier, the input stage is the real-time image capturing of the user, and the system camera & a 720p USB camera are made use for this purpose. The frame rate at which the camera capture the image is 4-5 frames per second. There are various factors which decide the number of frames per second or the frame rate in which the camera captures the user's image.

1. The processes performed over each frame in the image processing stage. As the number of processes increase, the frame rate gets reduced as some time will be dedicated for all these processes to be done. Also, the time for different functions to get performed is different in MATLAB. For example, it takes more time to write an image file to a location than to read the same image from the same location.

2. Increasing the number of frames actually slows down the total system speed as the system has to process each and every image frames separately. i.e., as the number of images got input to the image processing system increases and processing this increased number of frames takes more time. So the optimum number of frames input per second for the system to perform all the processes smoothly is 3 to 6.

In MATLAB, one can actually set the frequency at which a process runs. It has a `pause` function [5.1], which adds a pause of a predefined amount of time in between the execution of the program and thereby controls the time of execution of a process. MATLAB also has a class `robotics.Rate` [5.2], which also controls the execution of a loop in the program. But, the addition of these functions further increases the execution time, which is already raised due to a large number of processes to be done. So, it is better not to add an additional pause during the program execution.

Now, the reason why different cameras are used for the different results is, in the first method, the pointer is moving with the head movement while it is the eye-iris movement which controls the second output method. So the camera positions and the clarities are different for different purposes. For the second method, the camera should be placed close to the face to track the iris movement, and should have a perfect clarity image capturing. In the first method, the movements to be captured are wider than the second one. So the camera should be a little far from the user's face, and also the image clarity can be compromised.

In front of the user, the interface (system monitor) will be displaying 3 main sub-windows, in both the methods. In the first section, there is a keyboard. The live image of the user and a textbox for showing the typed letters are there in the next 2 windows. The difference in the 2 output methods (the head controlled & the eye controlled) are, the keyboards displaying in the interface. In the first method, where the movement of the head is used for key selection, the keyboard has all the letters plus a space key and a dot, while, the second one has a keyboard having a limited number of keys, to be clear, only 6 letters. In this method, the number of keys depends upon the quality of the camera. As the resolution of the camera increases, more number of keys can be added. This is because, out of the total resolution of 1280x960 pixels, the portion of the iris comprises of only of 32x32 pixels, inside which the gleam variations occur for about 22x14 pixels. That is, the variations of only 0.025 percent of the total frame resolution have to be recorded to control the key selection process.

Also, even if we keep our eyes steady staring at a particular direction, there will be a small variation of two to three pixels in the position of the gleam with respect to that of the iris, which is more than 14% in both the directions. So technically each key is having an average resolution of 7 x 7 pixels with a tolerance of ± 3 pixels in both x and y-axis. So, in order to increase the number of keys in the interface window, the quality of the camera capturing the real-time images of the user, should be enhanced.

The output of this stage is the live images of the user, which are input to the next image processing stage. The virtual keypads used in both the methods and the outputs (of input stage) of each method are shown in the figures below.

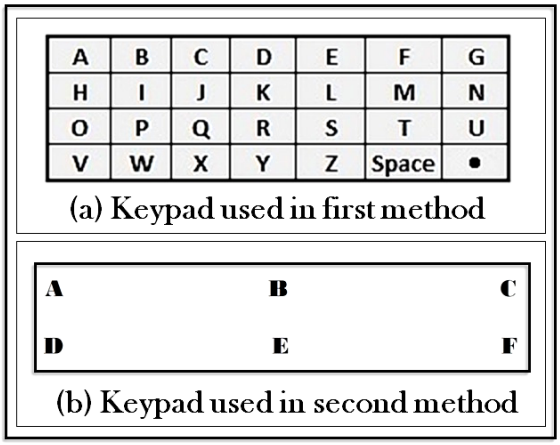

Figure 5.1. Virtual keypads used in the interfaces

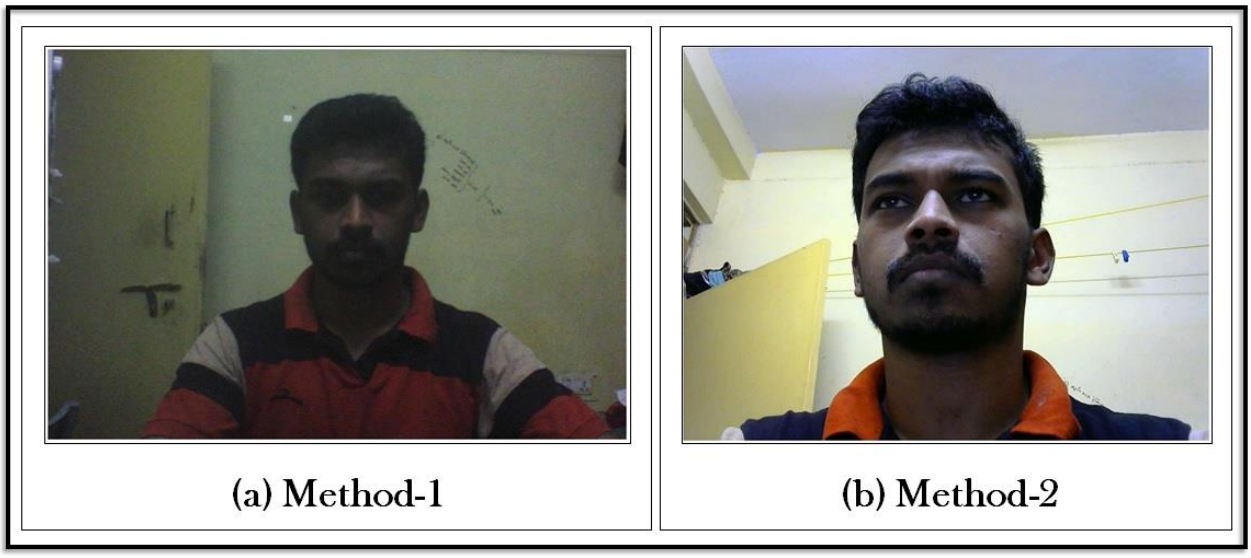

Figure 5.2. Output (captured live images) of both methods

5.2 THE IMAGE PROCESSING STAGE

Most of the algorithms and the tools described in chapter 4 are made use in this stage. The input to this stage is the captured image directly from the camera. The output of this stage are the coordinates of the detected face or the iris. The following sections describe the separate image processing steps used in implementing both the methods.

5.2.1 METHOD 1. HEAD-MOVEMENT CONTROLLED HCI

This method is the simplest of both. The block diagram of the image processing stage of this method is given below in fig 5.3

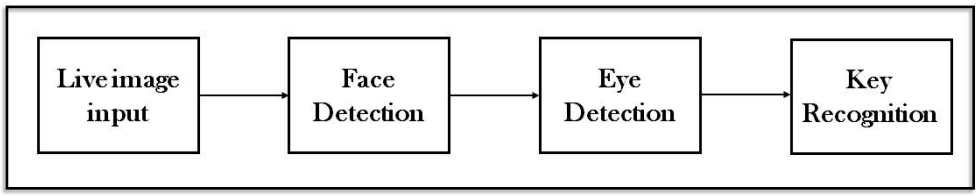

Figure 5.3. Block diagram of the image processing stage in method 1

This method starts with the live image received from the system camera (Fig 5.2(a)) as the input. Here no pre-processing steps are done before the face detection process. The Face-Detection algorithm (Viola-Jones) is applied directly on the first input frame, using the `vision.CascadeObjectDetector` System object [5.3] in the Computer Vision System Toolbox in MATLAB. The output is a 1x4 matrix having the details (coordinates, height and width) of the detected objects. The detected face in the input image is shown in the fig 5.4. The rectangle region of the detected face part is highlighted in the image using the `insertShape` function [5.4] in the Computer Vision System Toolbox.

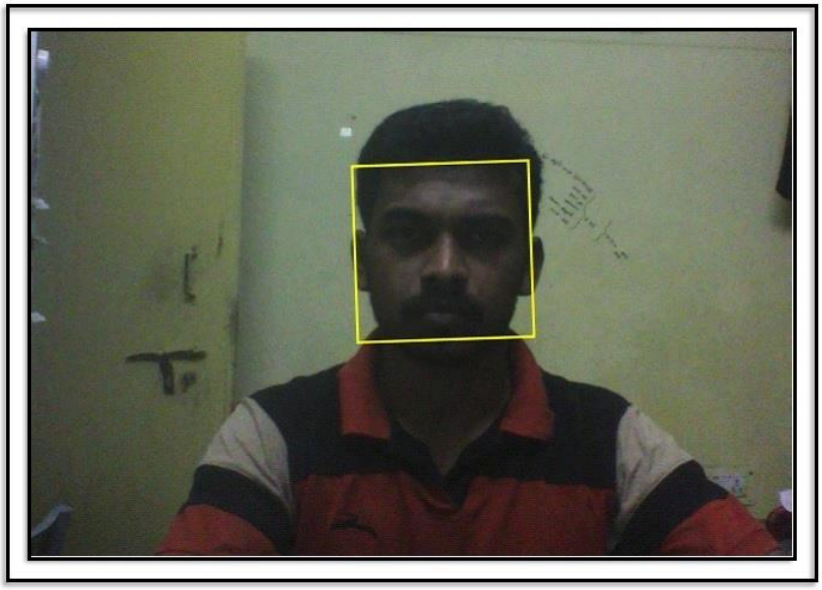

Figure 5.4. Detected Face in the input image

The next step is to detect the eye of the user. Using the pre-detected face portion as the Region of Interest, the Viola-Jones Algorithm is used again, with the appropriate features, for the detection of the eye. The figure 5.5 illustrates the detected eye (using the `insertShape` function) with the pre-detected face region as the RoI.

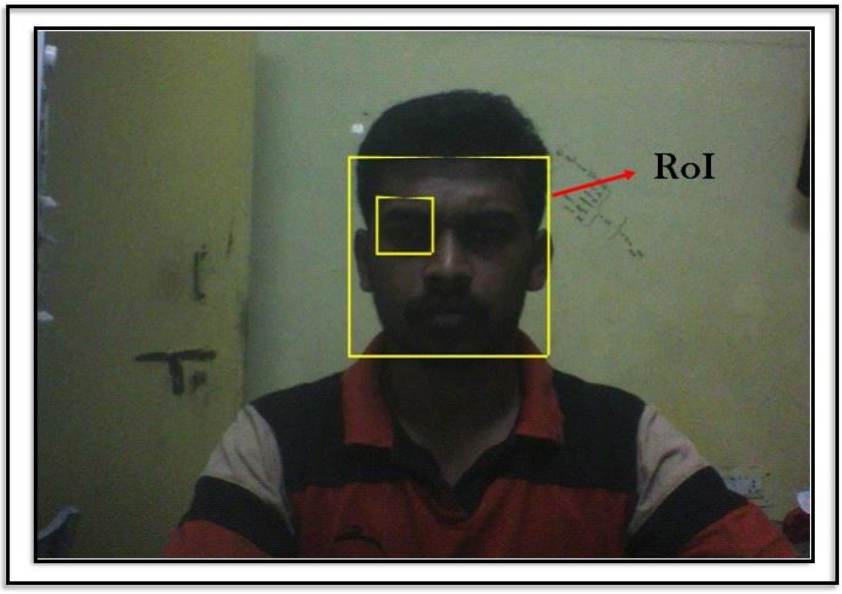

Figure 5.5. Detected Eye inside the RoI

Now, as the eye of the user is detected, the details of this detected eye has to be forwarded to the next section for the output stage.

5.2.2 METHOD 2. EYE-MOTION CONTROLLED HCI

This is the second type of HCI implementing in this project. As defined early, the user can do the type-in process only using his/her eyes' motion, without ever moving his/her head. The structural diagram of the image processing stage of this method is given in fig 5.6.

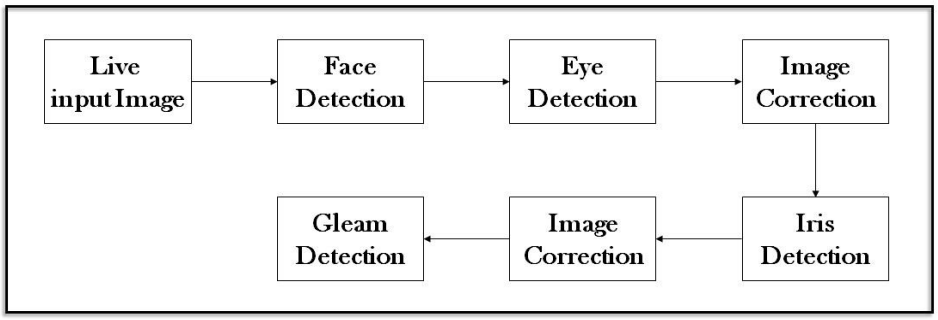

Figure 5.6. Block diagram of the image processing stage in method 2

The input is the same live user image as in the first method, but taken from a different camera. Here a 720p USB camera connected to the system is used. The first step here is the same face-detection using the Viola-Jones Algorithm.

5.2.2.1 EYE DETECTION

In this stage, the eye is to be detected using the pre-detected face as RoI. In this method, the camera is kept closer to the user's eye at an angle (not straight). So, using the detected face as the RoI for the next eye detection stage, has some limitations here. As the camera is not straight, there is a chance for some other face features to be detected as the eye, in the eye detection stage [5.5]. This false detection is shown in the figure 5.7.

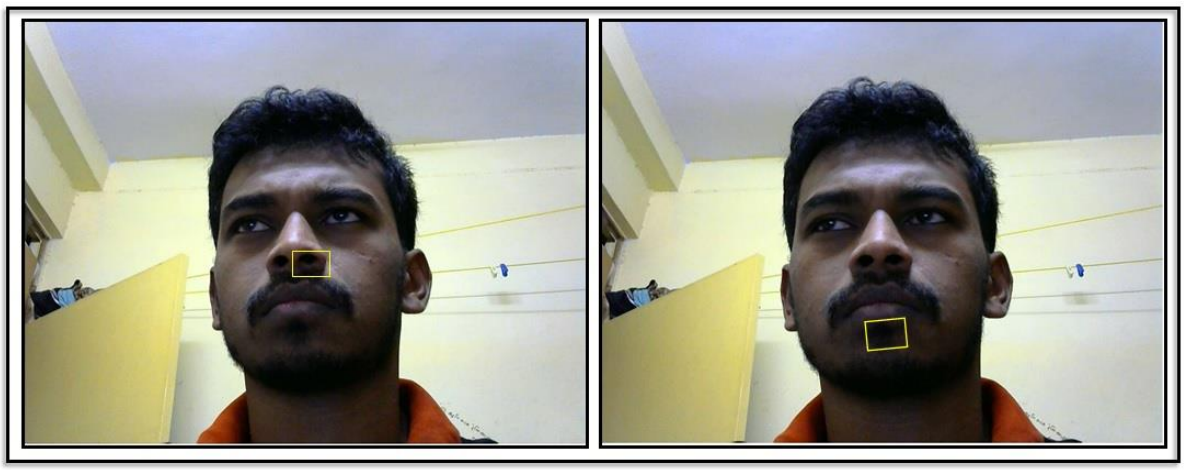

Figure 5.7. Wrong Detections due to the camera not being straight

One practical solution for this is, instead of using RoI, just crop the image such that the wrong features would be absent in the input image for eye detection as shown in figure 5.8.

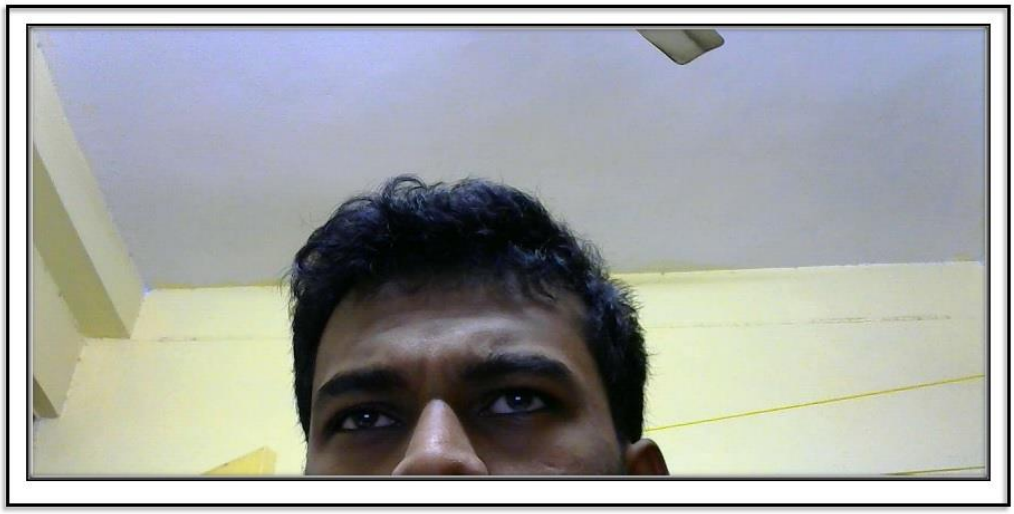

Figure 5.8. Cropped input image

One problem still persists here. Due to the angle created by the camera, the position of the iris and the gleam of both the eyes wouldn't be the same. And as the complete image of the face is not available too makes the detection of a particular eye (say left eye) every time wouldn't be possible. That is, every time the program runs, there is a chance for the detection of alternate eyes. This alternate detections are illustrated in the fig 5.9.

This is a big problem here, as the image is not straight, but at an angle makes the position of iris and the gleam for both the eyes in the image different. And as the program uses those positions to determine the user's gaze, the detected eye should not be changed. In

Figure 5.9. Detection of alternative eyes

other words, the program required for different (left and right) eyes are different. So, again cropping has to be done in the already cropped image (fig 5.8), so that every time only one eye is available for detection (fig 5.10). The cropping should be such that the position (coordinates) of the detected eye in the cropped eye should be the same in the initial input image too.

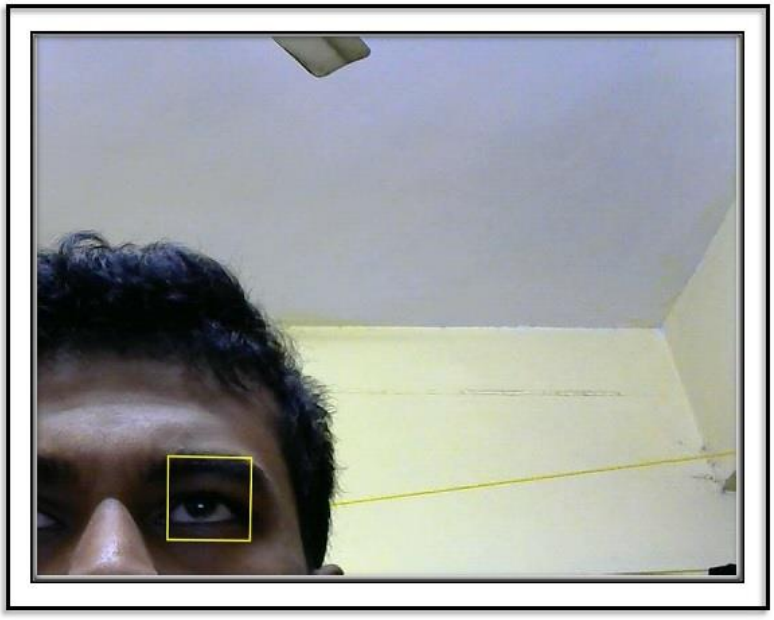

Figure 5.10. Cropped image input for eye detection

This cropping idea may seems silly, but considering the practical profits, it is working well, better than the RoI concept. This cropping is done only to detect the eye position in the input image and after the detection on the first frame, the position of the eye continues to be localized using the help of point tracking method (KLT feature tracking method).

5.2.2.2 IMAGE CORRECTION

This step is to modify the detected eye part, so as it best suits for the next stage, iris detection. The diagram below (fig 5.11) shows the image correction steps.

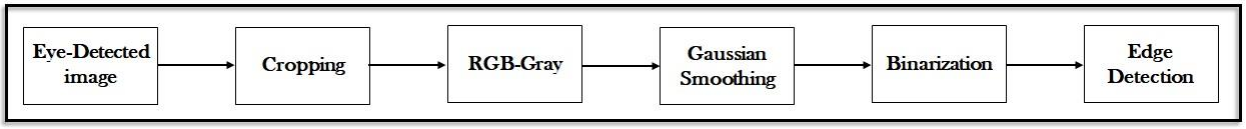

Figure 5.11. Image Correction steps

The eye-detected RGB image of the user is input to this stage first. This image is then cropped with the eye-detection output details, in order to crop out only the eye portion in the image. Then this cropped rgb image is then converted to grayscale image. The model of the cropped-gray converted image is shown in figure 5.12. Then this image is smoothed by utilizing the Gaussian smoothing method, which is further converted to binary image. The reason why the Gaussian smoothing is performing is, it make the rough edges in the binary image smooth, which makes narrow neat edges in the next step. The figure 5.13 illustrates the difference in the edge-detected images with and without the Gaussian smoothing. This slight variation counts, in the next iris detection step.

Figure 5.12. Image after cropping and RGB-Gray conversion

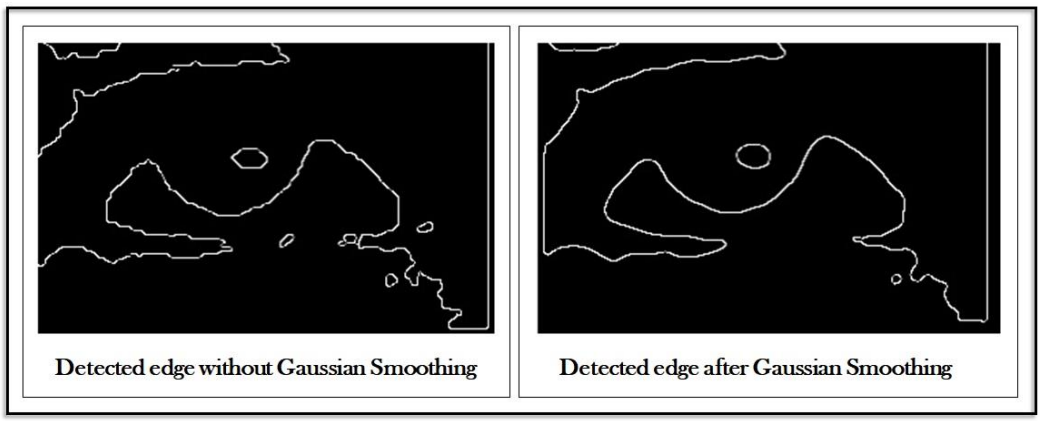

Figure 5.13. Edge detection with and without Gaussian smoothing

This edge-detection is the last step in this image correction stage. This edge detected image is input to the next stage.

5.2.2.3 IRIS DETECTION

The purpose of this step, as the name indicates, is to detect the iris from the edge-detected eye image input. This iris detection has two major uses. One is to find the iris centre and the next is to crop the eye image for the next Gleam detection step. Circular Hough Transform is used here to detect the circle of predefined radius range in the edge detected image. The radius of the circle (iris) depends on the camera properties. That is, the radius of the iris in the captured image varies with the camera resolution. The resolution of the image capturing used here is 1280xp60 pixels. The radius of the iris in the image is ranging from 16 to 19 pixels, and that is the radius range for the circle to be detected by the Hough transform from the edge detected input image. The figure 5.14 shows the iris-detected result of the Hough transform. Thus the iris centre coordinates are obtained, as the detected circle centre coordinates.

5.2.2.4 IMAGE CORRECTION

In this stage, it's not the iris detected image (5.14) which is getting corrected, but the binary image of the detected eye using the detected iris parameters. Using the iris center coordinates and the radius, the binary eye image is cropped here. So, in the resultant image, only the binary

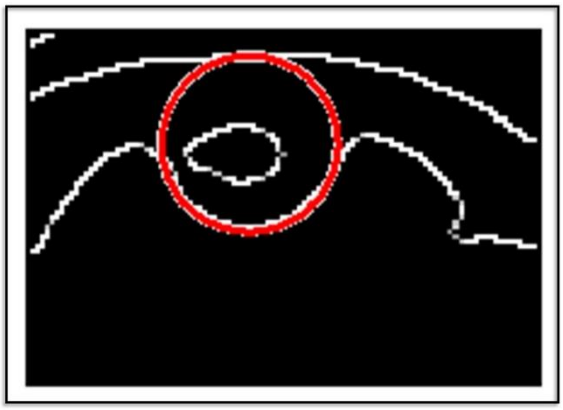

Figure 5.14. Iris-detected image

iris image will be there. The figure 5.15 shows the resultant image after the image correction.

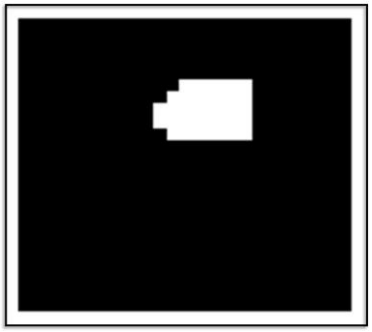

Figure 5.15. Cropped binary iris image

The binary eye image is cropped in to a square binary image, with the image centre as the iris centre, and the sides as twice the iris radius. The binarization of the eye image is done with a threshold, such that only the gleam inside the iris will have pixel value '1' (white pixel), and the others '0' (black). So, in the figure 5.15, the white pixels present represents the gleam inside the iris.

5.2.2.5 GLEAM DETECTION

As, the input to this stage is the binary cropped image of the iris (fig 5.15), it is an easy task to locate the gleam. Blob analysis is used here to detect the gleam, which it detects the white pixels in the binary image, and returns the blob-centroid coordinates. Thus the glint position is located. Now this blob coordinates and the iris centre coordinates are used to determine the gaze direction of the user.

5.3 THE OUTPUT STAGE

The output stages of both the methods (the Head-movement controlled HCI & the Eye-motion controlled HCI) are technically the same. It is the key-type in of the selected keys from the virtual keypads (fig 5.1) shown in both the interfaces. The differences in the keypads make the outputs different.

5.3.1 METHOD 1. HEAD-MOVEMENT CONTROLLED HCI

Here the input is the coordinate of the real-time position of the eye of the user. This coordinates determines the key, the user has to select and type. The keypad in this stage (fig 5.1 (a)) has 28 keys (26 English letters, a space bar and a dot (.)). Each key has a specific threshold in x and y direction and a specific range in both the directions.

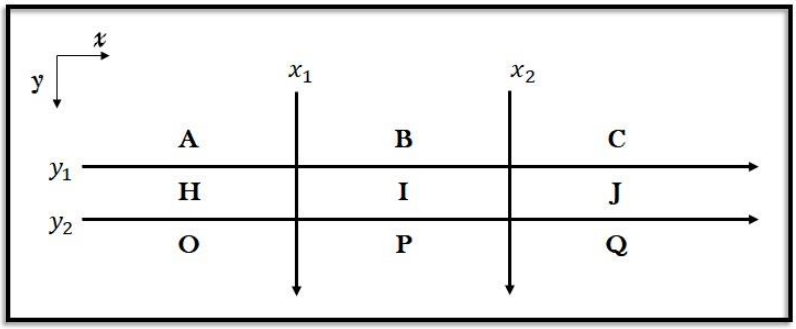

Figure 5.16. Key selection criteria (method 1)

The figure 5.16 illustrates the key selection criteria. Each letter has a specific range in both the directions. For example, suppose the detected eye's centre coordinate has the value, say x, in the x direction and the value in y direction as, 'y'. If ($x \leq x_1$) & ($y \geq y_2$), then the selected letter is 'o'. A pointer is placed in the interface, sweeping over the keypad, so that it is located over the selected keys. Whenever this pointer is over each keys, a counter will be running and when the selected key changes, the counter gets reset and starts counting from 0. Now, the selected key is typed-in, in the textbox available in the interface, if the pointer is placed over a key for 5 seconds, that is, if a key is selected for 5 seconds continuously.

5.3.2 METHOD 2. EYE-MOTION CONTROLLED HCI

The inputs to this stage are the iris radius and the gleam centroid coordinates. These input information are compared each other to determine the user's gaze direction. Here the ratio of the x and y coordinates of the centroid position separately to the radius of the iris together determines the selected key. Each key has a specified range for these two values. So the resultant values are compared with these specified range and to whichever key the result belongs, that particular letter is selected at that time. And similarly to the previous method,

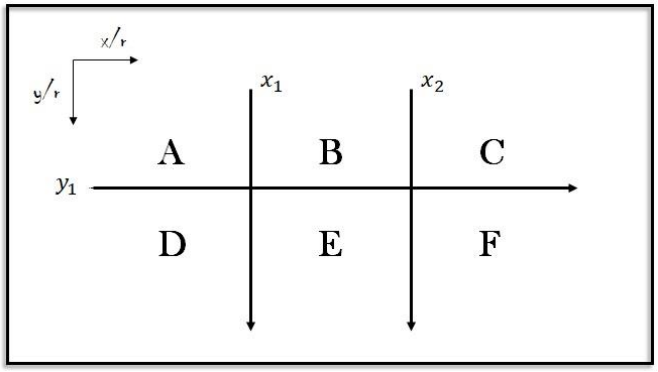

Figure 5.17. Key selection criteria (method 2)

the user has to stare at a particular letter for 5 seconds, for that letter to get typed-in in the text box. The figure 5.17 shows the key selection criteria. The only difference this has from the previous method is that, here the values of the ratios x/r & y/r are the deciding parameters. That is, for an input gleam parameter, if the ratio of the horizontal gleam position to the radius of the iris is less than x_1 and the ratio of the vertical gleam position to the radius of the iris is less than y_1, then the selected letter (the letter to which the user is looking at) is 'A'. Similarly the other letters are selected and typed-in.

Chapter 6
EXPERIMENT RESULTS

Two types of Human Control Interactions, the Head-movement controlled HCI and the Eye-motion controlled HCI have been discussed so far. Both the HCIs are implemented in MATLAB, and are tested in different environments. The interaction windows of both the methods are given in the figures 6.1 and 6.2.

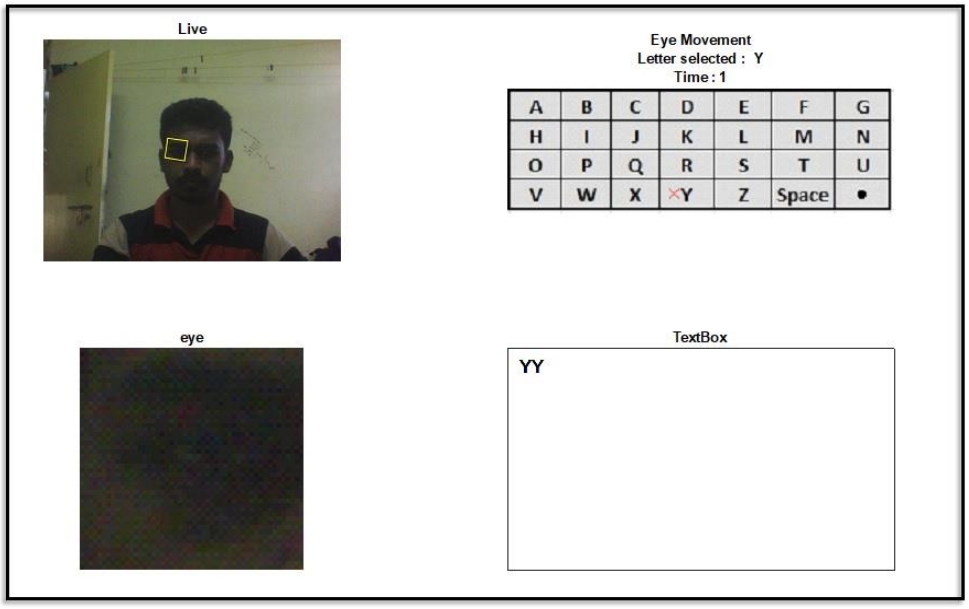

Figure 6.1. Interface of the Head-movement controlled HCI

There are 4 sections in this interface. The first (top left) window shows the real-time image of the user, in which the right eye is detected. The second window (top right) displays the virtual keypad with 28 keys. The bottom left window shows the detected eye portion of the user. And the last window is the text box, in which the selected keys are getting typed-in.

Even though the camera used in this method is low quality than the second method, this method gives the best result of the two. This method generates false results only when two people are present in the field. Other than this situation, if the lighting conditions are good, this method gives almost 100% results.

In the Eye-motion controlled HCI, the working environment influence the results more than in the Head-movement controlled HCI. The environment means the surrounding of the user. A second person in the frame affects the results. The lighting conditions should also be satisfactory for attaining best results. As the lighting conditions change, the threshold of the binarization processes performed at two places have to be changed accordingly. The figure 6.2 shows the HCI interface for this method.

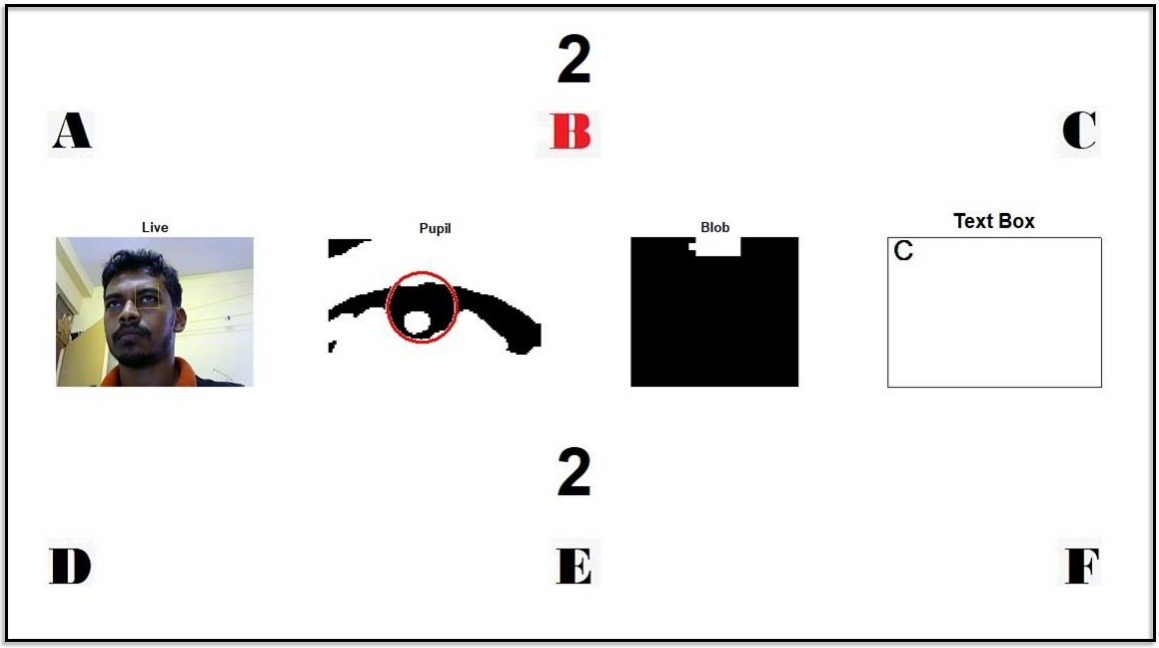

Figure 6.2. Interface of the Eye-motion controlled HCI

As it seems, this interface window has 3 rows. The first and the third row displays the letters (6 letters) to which the user has to stare at. Just above these 2 rows, a timer is running (showing '2' in fig 6.2) (in seconds). When the timer reaches 5 (seconds), the letter selected at that time will be typed-in in the text box given at the second row, last column. The selected letters (the letters at which the user is staring at) change its colour from black to red. In the second row, the first box displays the live image of the user, with the eye detected. The second one is the binary image of the detected eye with the iris detected. And the third one shows the binary image of the detected iris, with the gleam in white pixels.

Chapter 7
CONCLUSION & FUTURE WORK

7.1 CONCLUSION

This thesis proposed 2 types of HCI for those who can't use their hands for using the keyboard, the Head-movement controlled HCI and the Eye-motion controlled HCI. Both the methods made use of the Viola-Jones Algorithm for the face and eye detection. The first method mainly used these 2 detection methods only, while the eye-motion has controlled HCI utilized Circular Hough Transform for the iris detection, blob analysis for the gleam detection along with the Viola-Jones.

The head-movement controlled one is giving satisfactory results in almost every environment. While the second one's results are more dependent to the working conditions. It gives the best results when the lighting conditions are satisfactory. In bad light conditions, either the image binarization parts have to be modified, otherwise false detections are happening sometimes, which lead to the wrong key selection.

Even though, the results of the eye-controlled HCI are just satisfactory, considering the practical applications, this method has more significance than the head-controlled one. Moving the head every time to do different tasks is not practically feasible. On the other hand, the second one is more economical comparing to the other eye-detection methods available in the business.

7.2 FUTURE WORKS

Comparing the feasibility of the 2 methods proposed, the Eye-motion controlled HCI has a vast scope over the first one. In this thesis, the direction at which the user is staring at the screen, is successfully detected. By using good quality camera for live image

capturing, this detection can be done more precisely such that the gaze to every positions on the screen can be distinctively detected. One of the limitations of this method here in this thesis is its sensitivity to the lighting conditions. By using a light sensor, and programing the code to utilize separate thresholds, limits and tolerances for the image processing steps for separate lighting conditions (sensor readings) can solve this problem to an extent.

This same idea of precise gaze detection can be used on all applications where the mouse is used. Instead of using the mouse, the user can sweep the pointer all over the screen by just changing his/her gaze direction. By utilizing the blink-detection or speech recognition, the clicking, double clicking and other similar jobs done by the mouse can be swapped.

Chapter 8
REFERENCES

[1] Duchowski, Andrew T. "Eye tracking methodology." *Theory and practice* 328 (2007).

[2.1] Morimoto, Carlos Hitoshi, David Koons, A. Amit, Myron Flickner, and Shumin Zhai. "Keeping an eye for HCI." In *Computer Graphics and Image Processing, 1999. Proceedings. XII Brazilian Symposium on*, pp. 171-176. IEEE, 1999.

[2.2] Miniotas, Darius. "Application of Fitts' law to eye gaze interaction." In *CHI'00 extended abstracts on human factors in computing systems*, pp. 339-340. ACM, 2000.

[2.3] MacKenzie, I. Scott. "Fitts' law as a research and design tool in human-computer interaction." *Human-computer interaction* 7, no. 1 (1992): 91-139.

[2.4] Colburn, Alex, Michael F. Cohen, and Steven Drucker. "The role of eye gaze in avatar mediated conversational interfaces." *Sketches and Applications, Siggraph'00* (2000).

[2.5] Ohno, Takehiko. "Features of eye gaze interface for selection tasks." In *Computer Human Interaction, 1998. Proceedings. 3rd Asia Pacific*, pp. 176-181. IEEE, 1998.

[2.6] Yoo, Dong Hyun, Jae Heon Kim, Bang Rae Lee, and Myoung Jin Chung. "Non-contact eye gaze tracking system by mapping of corneal reflections." In *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, pp. 101-106. IEEE, 2002.

[2.7] Wang, Jian-Gang, and Eric Sung. "Study on eye gaze estimation." *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 32, no. 3 (2002): 332-350.

[2.8] Jacob, R. J., and Keith S. Karn. "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises." *Mind* 2, no. 3 (2003): 4.

[2.9] Ohno, Takehiko, Naoki Mukawa, and Shinjiro Kawato. "Just blink your eyes: A head-free gaze tracking system." In *CHI'03 extended abstracts on Human factors in computing systems*, pp. 950-957. ACM, 2003.

[2.10] Amir, Arnon, Lior Zimet, Alberto Sangiovanni-Vincentelli, and Sean Kao. "An embedded system for an eye-detection sensor." *Computer Vision and Image Understanding* 98, no. 1 (2005): 104-123.

[2.11] Yoo, Dong Hyun, and Myung Jin Chung. "A novel non-intrusive eye gaze estimation using cross-ratio under large head motion." *Computer Vision and Image Understanding* 98, no. 1 (2005): 25-51.

[2.12] Wang, Jian-Gang, Eric Sung, and Ronda Venkateswarlu. "Estimating the eye gaze from one eye." *Computer Vision and Image Understanding* 98, no. 1 (2005): 83-103.

[2.13] Morimoto, Carlos H., and Marcio RM Mimica. "Eye gaze tracking techniques for interactive applications." *Computer vision and image understanding* 98, no. 1 (2005): 4-24.

[2.14] Oyabu, Yuki, Hironobu Takano, and Kiyomi Nakamura. "Development of the eye input device using eye movement obtained by measuring the center position of the pupil." In *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 2948-2952. IEEE, 2012.

[2.15] Corcoran, Peter M., Florin Nanu, Stefan Petrescu, and Petronel Bigioi. "Real-time eye gaze tracking for gaming design and consumer electronics systems." *IEEE Transactions on Consumer Electronics* 58, no. 2 (2012).

[2.16] Baek, Seung-Jin, Kang-A. Choi, Chunfei Ma, Young-Hyun Kim, and Sung-Jea Ko. "Eyeball model-based iris center localization for visible image-based eye-gaze tracking systems." *IEEE Transactions on Consumer Electronics* 59, no. 2 (2013): 415-421.

[2.17] Sambrekar, Uma, and Dipali Ramdasi. "Human computer interaction for disabled using eye motion tracking." In *Information Processing (ICIP), 2015 International Conference on*, pp. 745-750. IEEE, 2015.

[4.1] Hsin, Chengho, Hoai-Nam Le, and Shaw-Jyh Shin. "Color to grayscale transform preserving natural order of hues." In *Electrical Engineering and Informatics (ICEEI), 2011 International Conference on*, pp. 1-6. IEEE, 2011.

[4.2] Song, Yibing, Linchao Bao, Xiaobin Xu, and Qingxiong Yang. "Decolorization: Is rgb2gray () out?." In *SIGGRAPH Asia 2013 Technical Briefs*, p. 15. ACM, 2013.

[4.3] Mishra, Debashis, Utpal Chandra De, Isita Bose, and Bishwojyoti Pradhan. "Fish school search approach to find optimized thresholds in gray-scale image." In *Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on*, pp. 1-4. IEEE, 2014.

[4.4] Yang, Ming-Hsuan, David J. Kriegman, and Narendra Ahuja. "Detecting faces in images: A survey." *IEEE Transactions on pattern analysis and machine intelligence* 24, no. 1 (2002): 34-58.

[4.5] Chauhan, Mayank, and Mukesh Sakle. "Study & Analysis of Different Face Detection Techniques." *International Journal of Computer Science and Information Technologies* 5, no. 2 (2014): 1615-1618.

[4.6] Kotropoulos, Constantine, and Ioannis Pitas. "Rule-based face detection in frontal views." In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 4, pp. 2537-2540. IEEE, 1997.

[4.7] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, pp. I-I. IEEE, 2001.

[4.8] Viola, Paul, and Michael J. Jones. "Robust real-time face detection." *International journal of computer vision* 57, no. 2 (2004): 137-154.

[4.9] Wang, Yi-Qing. "An analysis of the Viola-Jones face detection algorithm." *Image Processing On Line* 4 (2014): 128-148.

[4.10] Lienhart, Rainer, and Jochen Maydt. "An extended set of haar-like features for rapid object detection." In *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1, pp. I-I. IEEE, 2002.

[4.11] Papageorgiou, Constantine P., Michael Oren, and Tomaso Poggio. "A general framework for object detection." In *Computer vision, 1998. sixth international conference on*, pp. 555-562. IEEE, 1998.

[4.12] Freund, Yoav, and Robert E. Schapire. "A desicion-theoretic generalization of on-line learning and an application to boosting." In *European conference on computational learning theory*, pp. 23-37. Springer, Berlin, Heidelberg, 1995.

[4.13] Freund, Yoav. "Boosting a weak learning algorithm by majority." *Information and computation* 121, no. 2 (1995): 256-285.

[4.14] Lu, Huchuan, Wei Zhang, and Deli Yang. "Eye detection based on rectangle features and pixel-pattern-based texture features." In *Intelligent Signal Processing and Communication Systems, 2007. ISPACS 2007. International Symposium on*, pp. 746-749. IEEE, 2007.

[4.15] Nister, David, and Charilaos Christopoulos. "Lossless region of interest with a naturally progressive still image coding algorithm." In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, pp. 856-860. IEEE, 1998.

[4.16] Christopoulos, Charilaos, Joel Askelof, and Mathias Larsson. "Efficient methods for encoding regions of interest in the upcoming JPEG2000 still image coding standard." *IEEE Signal Processing Letters* 7, no. 9 (2000): 247-249.

[4.17] Askelöf, Joel, Mathias Larsson Carlander, and Charilaos Christopoulos. "Region of interest coding in JPEG 2000." *Signal Processing: Image Communication* 17, no. 1 (2002): 105-111.

[4.18] Shi, Jianbo. "Good features to track." In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593-600. IEEE, 1994.

[4.19] Tomasi, Carlo, and Takeo Kanade. "Detection and tracking of point features." (1991).

[4.20] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674-679.

[4.21] *https://in.mathworks.com/help/vision/ref/vision.pointtracker-class.html*

[4.22] http://in.mathworks.com/help/vision/ref/estimategeometrictransform.html

[4.23] Torr, Philip HS, and Andrew Zisserman. "MLESAC: A new robust estimator with application to estimating image geometry." *Computer Vision and Image Understanding* 78, no. 1 (2000): 138-156.

[4.24]https://in.mathworks.com/help/images/ref/projective2d.transformpointsforward.html

[4.25] Chaple, Girish N., R. D. Daruwala, and Manoj S. Gofane. "Comparisions of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA." In *Technologies for Sustainable Development (ICTSD), 2015 International Conference on*, pp. 1-4. IEEE, 2015.

[4.26] Shrivakshan, G. T., and C. Chandrasekar. "A comparison of various edge detection techniques used in image processing." *IJCSI International Journal of Computer Science Issues* 9, no. 5 (2012): 272-276.

[4.27] Canny, John. "A computational approach to edge detection." *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986): 679-698.

[4.28] Maini, Raman, and Himanshu Aggarwal. "Study and comparison of various image edge detection techniques." *International journal of image processing (IJIP)* 3, no. 1 (2009): 1-11.

[4.29] Khairosfaizal, WMK Wan Mohd, and A. J. Nor'aini. "Eyes detection in facial images using circular hough transform." In *Signal Processing & Its Applications, 2009. CSPA 2009. 5th International Colloquium on*, pp. 238-242. IEEE, 2009.

[4.30] Duda, Richard O., and Peter E. Hart. "Use of the Hough transformation to detect lines and curves in pictures." *Communications of the ACM* 15, no. 1 (1972): 11-15.

[4.31] Yuen, H. K., John Princen, John Illingworth, and Josef Kittler. "Comparative study of Hough transform methods for circle finding." *Image and vision computing* 8, no. 1 (1990): 71-77.

[4.32] Ballard, Dana H. "Generalizing the Hough transform to detect arbitrary shapes." *Pattern recognition* 13, no. 2 (1981): 111-122.

[4.33] Tian, Qi-Chuan, Quan Pan, Yong-Mei Cheng, and Quan-Xue Gao. "Fast algorithm and application of hough transform in iris segmentation." In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, vol. 7, pp. 3977-3980. IEEE, 2004.

[4.34] Lindeberg, Tony. "Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention." *International Journal of Computer Vision* 11, no. 3 (1993): 283-318.

[5.1] *http://in.mathworks.com/help/matlab/ref/pause.html*

[5.2] *https://in.mathworks.com/help/robotics/ref/robotics.rate-class.html*

[5.3] *https://in.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html*

[5.4] *https://in.mathworks.com/help/vision/ref/insertshape.html*

[5.5] Lopar, Markan, and Slobodan Ribarić. "An Overview and Evaluation of Various Face and Eyes Detection Algorithms for Driver Fatigue Monitoring Systems." *arXiv preprint arXiv:1310.0317* (2013).

Thesis

ORIGINALITY REPORT

%3	%1	%3	%1
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Sambrekar, Uma, and Dipali Ramdasi. "Human computer interaction for disabled using eye motion tracking", 2015 International Conference on Information Processing (ICIP), 2015. Publication	%1
2	Oyabu, Yuki, Hironobu Takano, and Kiyomi Nakamura. "Development of the eye input device using eye movement obtained by measuring the center position of the pupil", 2012 IEEE International Conference on Systems Man and Cybernetics (SMC), 2012. Publication	<%1
3	milos.stojmenovic.com Internet Source	<%1
4	Corcoran, Peter, Florin Nanu, Stefan Petrescu, and Petronel Bigioi. "Real-time eye gaze tracking for gaming design and consumer electronics systems", IEEE Transactions on Consumer Electronics, 2012. Publication	<%1

5	Submitted to University of South Florida Student Paper	<%1
6	www.brl.ntt.co.jp Internet Source	<%1
7	robotics.ee.uwa.edu.au Internet Source	<%1
8	Huang, Long Jun, Qing Hua Liu, Jie Tang, and Ping Li. "Scratch line detection and restoration based on Sobel operator", International Journal of Grid and Utility Computing, 2015. Publication	<%1
9	research.ijcaonline.org Internet Source	<%1
10	Submitted to ABV-Indian Institute of Information Technology and Management Gwalior Student Paper	<%1
11	Wei-Kai Liao. "Weighted fractal image coding", 2007 IEEE International Conference on Systems Man and Cybernetics, 10/2007 Publication	<%1
12	web.it.kth.se Internet Source	<%1
13	"Facial feature point location and tracking method based on improved Viola-Jones	<%1

algorithm and Kalman filter prediction mechanism", Environment Energy and Applied Technology, 2015.
Publication

14	www.ucsp.edu.pe Internet Source	<%1
15	Sambrekar, Uma, and Dipali Ramdasi. "Estimation of gaze for human computer interaction", 2015 International Conference on Industrial Instrumentation and Control (ICIC), 2015. Publication	<%1
16	Sacchi, C.. "Advanced image-processing tools for counting people in tourist site-monitoring applications", Signal Processing, 200105 Publication	<%1
17	Inari, T.. "Optical inspection system for the inner surface of a pipe using detection of circular images projected by a laser source", Measurement, 199404 Publication	<%1
18	O. Yamaguchi. "Face Recognition Using the Classified Appearance-based Quotient Image", 7th International Conference on Automatic Face and Gesture Recognition (FGR06), 2006 Publication	<%1

Baek, Seung-Jin, Kang-A Choi, Chunfei Ma,

19	Young-Hyun Kim, and Sung-Jea Ko. "Eyeball model-based iris center localization for visible image-based eye-gaze tracking systems", IEEE Transactions on Consumer Electronics, 2013. Publication	<%1
20	Amir, A.. "An embedded system for an eye-detection sensor", Computer Vision and Image Understanding, 200504 Publication	<%1
21	www.bmva.org Internet Source	<%1
22	Al-Rahayfeh, Amer, and Miad Faezipour. "Enhanced combined eye gaze direction classification and head flexion detection system", IEEE Long Island Systems Applications and Technology (LISAT) Conference 2014, 2014. Publication	<%1

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		