A Robust Cursor Activity Control with Iris Gesture and Blink Detection Technique

by

Md. Rayhan Al Islam
15101063
Maliha Rahman
15101105
Md. Rezyuan
15101114

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
April 2019

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Md. Rayhan Al Islam
15101063

Maliha Rahman
15101105

Md. Rezyuan
15101114

Approval

The thesis titled "A Robust Cursor Activity Control with Iris Gesture and Blink Detection Technique" submitted by

1. Md. Rayhan Al Islam (15101063)
2. Maliha Rahman (15101105)
3. Md. Rezyuan (15101114)

Of Spring, 2019 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on April 25, 2019.

Examining Committee:

Supervisor:
(Member)

Dr. Jia Uddin
Associate Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Md. Abdul Mottalib, PhD
Professor
Department of Computer Science and Engineering
Brac University

Abstract

Using computers for people with very limited or no arm movements is a very tough task and most of the time impossible. In this paper, we are proposing an iris based cursor control system developed specifically for physically challenged people. Eye gaze localization has been a very popular research field for several years. Tobi Eyes, Gaze Pointer Face Detection with viola-jones are the advanced works in this field but both have certain limitations such as the infrared ray issues and accuracy deviation because of unstable frames. In our work, we have developed a model to overcome these barriers of the state of the art systems. Along with the software model we are also proposing a hardware model in order to improve the accuracy of previously mentioned difficulties. In our system, we used Hough circle transform for localizing the eye gaze. The developed system is tested by multiple users and we achieved around 89.8% accuracy. This model will help the disabled peoples to get themselves out from their limited world. Furthermore, the functionality of using the on-screen keyboard will help them to interact very easily. This system will ensure maximum user comfort with minimum setup requirements.

Keywords: Cursor Control; Image Processing; Blink Detection; Iris Gesture; Hough Circle Transform; Window to View port Transformation; Haarcascade Eye Trees

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis have been completed without any major interruption. Secondly, We would like to express our sincere gratitude to our supervisor Dr. Jia Uddin and co-supervisor Rubayat Ahmed Khan for their highest attention and valuable time. We would also like to thank them for giving us the opportunity to work on this topic and guiding us all the way through the process. Moreover, we would like to thank BRAC University Computer Science and Engineering department for providing us with the facilities to conduct this research. And finally to our parents with their kind support and prayer we are now on the verge of our graduation.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

An eye control based computer system works on determining the cursor position depending on the focal point of two eyes. One of the major eye region is the Iris. This thin layer actually regulates a human eye color. This portion also helps to ascertain the diameter of the pupil. Eye lens passes lights and on the retina's focal point, the objects are structured and sent to the nerves in our brain. Usually, the main focal point is the center of the retina depending on a perfect eye sight. In Figure 1.1 the fragmentation of eyes are shown.

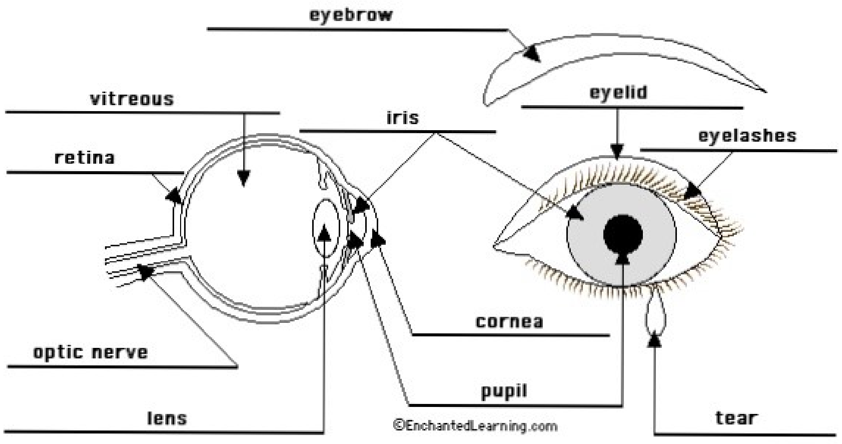


Figure 1.1: Eye fragments

Accordingly, this center point is also the center of the iris. So, deciding the iris position or the retina position will work on deciding in which point the person is focused on. Therefore, the position of iris or pupil is used in all of the systems to place the cursor position.

1.1 Motivation

As of late there has been a growing interest in improving natural interaction between human and computer. Several studies for human-computer interaction in universal computing are introduced [1]. It is a functioning examination field for some specialists working on disable people who cannot move anything except their eyes. A statistics from 2018 shows in Figure 1.2, there is almost 5.4 million paralyzed people all over the world [2]. If their knowledge is connected to a computer usage, they can create a massive turn over in the world.

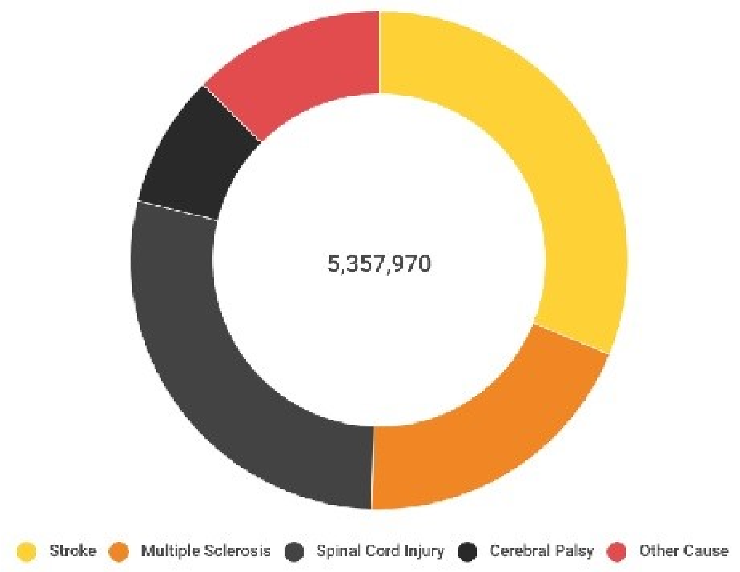


Figure 1.2: Statistics of Disabled People

From the mentioned figure we can see a major amount of people in the world are suffering from paralysis. This physical limitation is becoming a burden for their regular life. Most of these peoples' have knowledge and creativity but they cannot express. A boundary creates between their knowledge and their physical limitations.

1.2 Objective

This research aims in developing a system that can aid the physically challenged by allowing them to interact with a computer system using only their eyes. The vision-based interface technique extracts motion information without any high cost equipment's from an input video image. Thus, vision-based approach is taken to account as an effective technique to develop human computer interface systems. Eye movements can be captured and used as control signals to enable people to interact with interfaces directly without the need for mouse or keyboard input [3]. Moreover, computers can be used by persons with disabilities for communication, environmental control, source of information and entertainment. Their thoughts and ideas can be shared through our system. Peoples without hands or legs or other disabilities will be facilitated through our system and with their eye gesture they will be able to do all the required computer works. They will be able to share their thoughts and ideas through our system.

1.3 Thesis Orientation

The rest of the thesis is explained as follows-

- Chapter 2 includes literature review with necessary references

- Chapter 3 contains algorithms and system implementation
- Chapter 4 shows result analysis
- Chapter 5 includes conclusion and future work

Chapter 2

Background Study

2.1 Literature Review

To adapt with the approaches that has been attempted till now which are related to the topic we have studied several research paper, journals and articles. As technology is improving day by day it has become one kind of mandatory for all to use computer. Though there are many people who are physically challenged but they have potential and demand to work using computer. That's how scholars have come up with the idea to retina based cursor control for solving the issues. There are several ways to make this possible but the common things that come with these ideas are: image pre-processing, segmentation, feature extraction, classification and evaluation. Working long time sitting in front of a computer causes musculoskeletal imbalance which is related to the physical factors and also psychological factors [4]. Even if, the user has placed the computer to the wrong place where he or she has to extend hands or shoulder to reach to the input devices of computer such as mouse, keyboard etc. causes intensive pain in shoulder, and headache however it can also be responsible for toughness in the neck and shoulder [5].
Moreover, any repetitive actions like this can cause harm permanently to health is known as repetitive strain injury (RSI). Whenever using input devices of computer like mouse can cause RSI on wrist.There are many solutions has been added where many of them are related with external input devices to get rid of above mentioned diseases. The problem is all of the solution has been related with using external hardware devices like head mounted heavy devices or glasses for getting the gaze information [6]. ITU gaze tracer uses Open CV which is developed by a research group from IT University of Copenhagen which requires additional costly hardware setup as well as head movements for tracking [7]. But there are people who are not able to move head. Later on, another system of gaze tracking was developed by a researcher named Ohno, et al. This system requires three cameras for tacking. Among them two cameras are positioned at the top and the third one is placed at the bottom which operates using infrared spectrum [7]. However, many research have already been showed that IR is dangerous as causing various kinds of diseases in skin and eye. All those devices are also costly which is beyond imagination for general people to use.
Another system was proposed in 2014 where the cursor movement was also based on users gaze information [8]. There some controlling keys were showed on a screen and after calibration one can combine speech, controlling his atmosphere, type as

well as can operate telephone, computers external devices like mouse, run computer programs and accessing the internet and mail system. Thus the system was based on image processing using conversion of gray scale, edge detection, using threshold values and calculation of iris focus point displacement vector [8]. After that, scholars stepped one step ahead for the improvement of human computer interaction (HCI). A paper was published where the system was able enough to control the cursor automatically causing the focus point of eyesight and perform mouse-operation by performing blinking action [9]. The main purpose of the system was eye based human computer interaction (HCI) which gives real time tracking of eyes and eye-gaze information. There are many algorithms for eye detection and in there regression, Bayesian and discriminative approaches were used. For eye tracking, limbus, pupil tracking, electrooculography and saccade as well as face detection algorithm were performed. Those systems were more complicated to detect the eye portion properly even when the terms of efficiency come it could not be able to take the position.
In terms of face detection, an article published in bio medical soft computing and human science, we have seen that researchers are using viola Jones algorithm. According to the article, the viola Jones algorithm is working by generating the integral image. Haar extractors were calculated from the integral image by adding number [10]. The actual detection happens inside a window where the min and max size of the window can be chosen and for each window sliding size can also be chosen corresponding to the size. In each step the window can slide both vertically and horizontally. A set of N- face recognition filters was applied on each window. Whenever a filter gives positive result there is no need of rest of the filters but if a filter fails to detect face then rest starts working. The main problem of viola Jones is, at the very beginning it extracts white band then dark band and then it start matching patterns related to face as well as it's working parameter depends on environment, its mechanism becomes complicated to perform in a live video stream where the environment is a bit complex.
Another drawback of the previous work was complexity in terms of setup of face-camera pose (head orientation), resolution of image, illumination and motion dynamics etc. One more thing is needed to include that the real image used many heuristic methods where intensity are becoming more sensitive than the real time performance. Another paper was found where it describes the various ways for tracking eye retina [9]. Accordingly it discussed various ways for blink detection and face detection. They summarized face detection in two categories which are feature based method and image based method. In feature based method features are detected at the very beginning like nose, eye and mouth. After that it takes decision based on the detected features if all those are going to represent the human face or not from the given image. Next, image based technique is one kind of template matching which can be simply used for face detection with efficient accuracy. Many researchers have been detected eye blink with the help of Open CV from python and dlib library. Calculation of eye aspect ratio (EAR) based upon facial landmark is an elegant solution and easy to perform. Facial landmarks can be used to localize important regions from face like nose, mouth, eyelash and eye and from there special areas can be extracted using the indexes. In terms of blink the region of interest (ROI) is the eye and from the paper it has been stated that six co-ordinates represent each eye, so the indexes of those co-ordinates are needed to detect each eye. After that, with the help of an equation eye aspect ratio has been

found and it represents a constant value. When the eyes were open the value of the metric was a pretty stable value. The value drops when the eye is closed that means a blink has occurred.
In another paper a noticeable thing has been marked which is preprocessing of image or frames for smoothing of contours where the aim is simplicity for successive contour detection [11]. Unlike image restoration algorithm, the purposes of filters are not only removing noise but also texture since the latter tends to create problem for contour detection. There are local and global contour preserving filters which depend on variation of methods and non-linear diffusion. The authors have mentioned for removing noise low pass filtering is one kind of best solution in terms of accuracy. Some edges and contour have higher frequency and this becomes problematic too to process it. Further many techniques have been proposed to overcome it known as local adaptive smoothing algorithm where it computes the average weight of neighborhood pixels of gray scale depends on various ways of local pattern configuration.
In reviewing paper, another filtering method has come to light which is kalman filter. In a paper, describing noise removal mentioned about kalman filter which measure noises, inaccuracies and then produces a value which is near to the actual value and associated calculated values [12]. It is an algorithm which shows its performance by making optimal use of imprecise data of linear system even with the Gaussian errors and continuously updates the best prediction for system's current state. Kalman filters work effectively in reducing noise from an image with maintaining underlying structure of an image. However, it shows efficiency in noise removing for helping parallel programs to control the detection and movement of cursor.
Our proposed system is based on a simple web cam which is not that much costly and an infrared free cam. It relieves users from carrying extra heavy devices as well as increases the accuracy in contrast to others. However, the main problem was working with low resolution of images and in dark lights. To overcome all those things, we analyzed a couple of solutions and from there the best techniques have been applied. To get the desired information from the input we have used advanced image processing techniques and our own developed methodologies. After that, one thing come which says how accurately the system supports human computer interaction (HCI). We have increased our thought one step ahead and accordingly we tried to make smooth movement of the cursor. In the contrast to many processing techniques the movement of cursor in our system shows the better performance. The calibration process in our system is going to help the user in terms of view port transformation. Moreover, we have designed our system keeping in mind how we people interact with computer in day to day life. The navigation of computer has been developed from the movement information of the eyes. Those information has helped a lot for cursor movements in a more authentic way. Again the average filtering techniques shows better performance in the interpretation for reducing noise and smoothing the detection of retina. Our own heuristic calculation for blink detection allowed the system to control the cursor with better efficiency.

Chapter 3

Algorithms and System Implementation

3.1 System Overview

The block diagram in Figure 3.1 is demonstrating the overall work flow of the proposed system. We are dividing the whole system into several parts which are – video capturing, frame pre-processing with focal point detection and cursor movement. After capturing the video from camera we detected the iris from eye and find the focus point, which is used to move the cursor in the screen. The blink detection system is running in parallel to detect blink and run the click function.

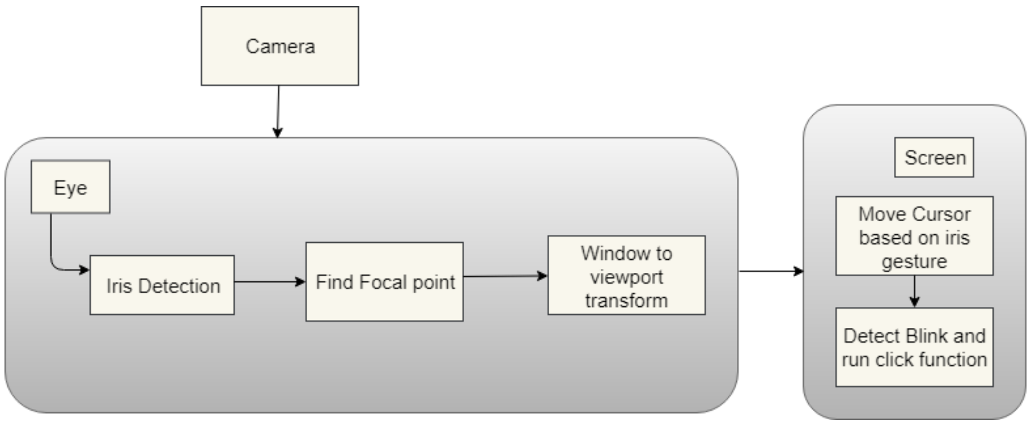

Camera
Eye
Iris Detection
Find Focal point
Window to viewport transform
Screen
Move Cursor based on iris gesture
Detect Blink and run click function

Figure 3.1: System Block Diagram

3.2 Experimental Setup

In terms of improving system accuracy, hardware model plays a vital role. Therefore, after reviewing the previous works we have found that most of the work's limitation is unstable frames. Thus, if we settle down the frame and make it fixed, there will be huge accuracy improvement in detecting eyes. So, we are using a cap and we attached a Logitech C270 for the web cam. The camera is placed at the extended portion of the cap. This helps us to keep our frame stable and fixed. Though the user moves his head, the frame will remain same.

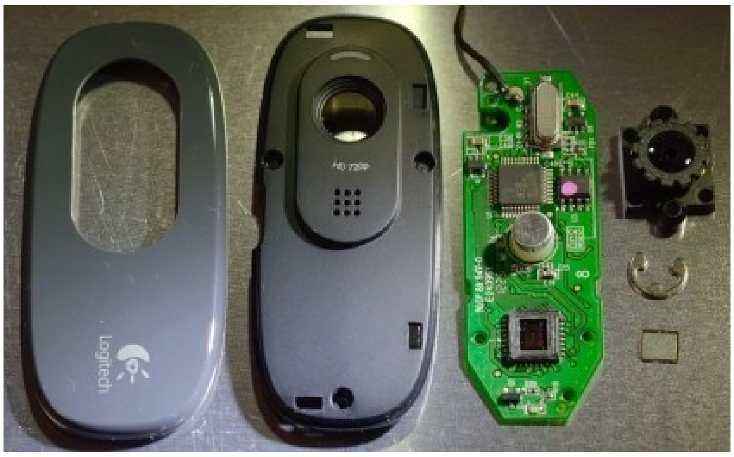


Figure 3.2: Web cam (Logitech C270)

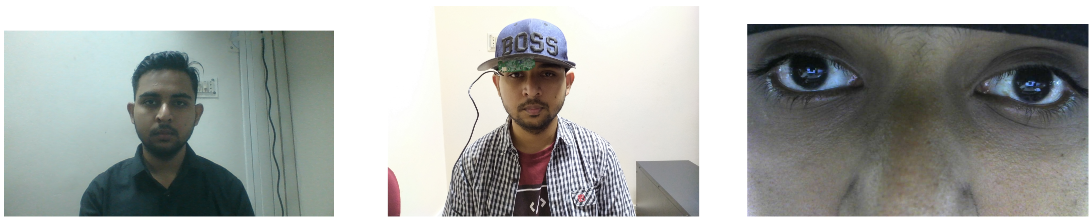

(a) User Web Cam View (b) Hardware Model (c) Hardware Model View

Figure 3.3: Comparison Between Different View of Web cam

We thought about the user comfort so we removed the plastic case of the camera and positioned only the logic board at the extended portion of the cap displayed in Figure 3.2. Thus, the camera weight became negligible and the system is very comfortable to manage for the user.
We tried to implement the system using integrated web cam of the laptop and an extra web cam over monitor in the desktop shown in Figure 3.3a but we found that, through that model the accuracy decreases drastically as the video will capture the whole face first and then eye. So we have planned to have the cam as close as possible to user eyes so that we can only focus on eyes rather than whole face. In Figure 3.3b our integrated hardware model is shown. Lastly, in Figure 3.3c the hardware model view is displayed.

3.3 Frame Pre-processing

In the pre-processing part which is shown in Figure 3.4, we have cropped the actual frame of our actual region of interest. Then the frame was converted to HSV. We set a lower and lower and upper threshold value to detect the iris region as it is comparatively darker than other pixels. Furthermore, the noise was reduced using Blur function. After that, the detection process starts.

3.3.1 Framing and determining Region of Interest (ROI)

OpenCV is defined with enormous powerful video editing functions. In current scenario, scanning of images, face recognition can be easily done with OpenCV.

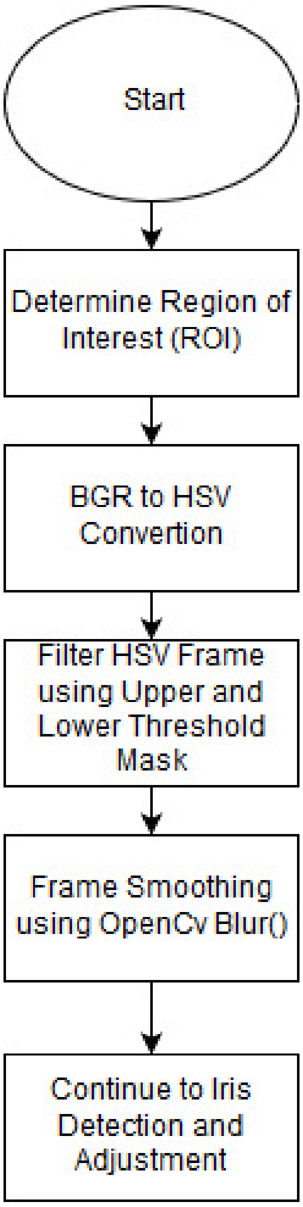
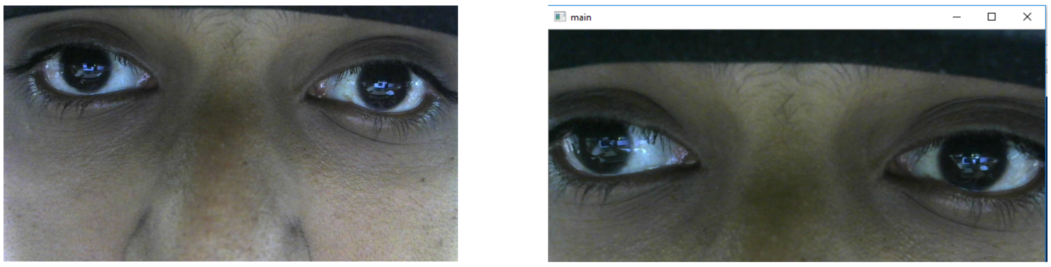

Start
Determine Region of Interest (ROI)
BGR to HSV Convertion
Filter HSV Frame using Upper and Lower Threshold Mask
Frame Smoothing using OpenCv Blur()
Continue to Iris Detection and Adjustment

Figure 3.4: Frame Processing



(a) Actual Camera Frame (b) Region Of Interest (ROI)

Figure 3.5: Comparison Between Camera Frame and Region Of Interest (ROI)

Basically, it takes video or recorder clips as input divide it into frames. After that, any operations can be performed on those saved frames. Some functions related to computer vision are well popular and can be easily handled with the help of this. Most often we need to capture live video with the help of camera where camera can be used from laptop or external camera can also be used. According to our research we are capturing live video stream with the help of web cam. The size of frame can vary for different devices. For our perspective the size of frame is 640480. This system provides quite simple interface to do this. For capturing video using web cam, the video capture process from OpenCV can be used [13]. However, we can also use the primary camera that is connected with the computer. Accordingly, we can also other cameras if they are connected with the device so that better interaction is possible in between computer and human. After capturing video it will capture frame to frame. Basically what is means is going to break the video into frame. We are going to get pictures from that for further processing. From the figure mentioned below we can see how the frames look after using the method. If we need pause in between every frame in the video there we can add some delay. Here are some parameters that can be used to make the best use of the function. Whenever zero is passed in the parameter the video is going to pause for an infinite amount of time. That's why we are adding values in the parameter which is greater than zero.
After that it will capture frame to frame. After reading the video it can be displayed frame by frame shown in Figure 3.5a.
Here comes one more interesting thing which is very useful for our research that is sometimes we do not need the whole frame. As we are going to detect the iris at the very first step so it is necessary for us to identify region of interest which is known as ROI [14]. That means we can adjust the amount of area we want to use from the capture video. It can be said in other word, re-sizing the frames of the video. Let's take an example that is if we say roi = frame [220: 300, 240: 450] it reflects we are going to take the amount of area that is shown in the array. From the argument we can demonstrate it as the left part before comma is for Y-axis and the rest of the thing is for X-axis. If we change the values of 220 to 100 or 0, the change we will see is in Y-axis. That means frame will increase along with Y-axis. However, the same thing happens for the right side also. We can change the value of 240 to any range according to our criteria. But one thing should be remembered the range should not exceed the limit of the frame which means the range must be in between the size of the frame. That's how we are going to use the concept of ROI in our research. For us we are going to extend it in a limit so that we can get the detected object per our interest. So the ROI for us is going to be roi = frame [0: 300, 0: 640]. From that

we will get the region where we want to implement further functions for processing it. In Figure 3.5b we can see how it re-sizes the frame for our region of interest. After all of these one thing need to remember that is releasing the capture. This can be done by a single line of code using Open CV that is cap.release() function. As we all know for every key in the keyboard there is an ASCII value and with the help of ASCII value we can be solved capture releasing and destroying all windows. 27 is the ASCII value for 'esc' key in keyboard. So if the value of key is 27 that means if we press 'esc' key we can release the capture of frames and as well as destroying all the windows.

3.3.2 BGR to HSV Conversion

HSV (hue, saturation, value) is a different presentation of RGB color model in computer graphics in human vision which reflects color making attributes. Each hue are re-arranged in a radial slice where the range is from black to white and the range covers both from the bottom to top. Here both parameters carry specific meaning, as HSV models how various colors combine together accordingly saturation rearrange different shades from bright colored. Lastly, value dimension assemble all the combination of color with the range as mentioned earlier. For HSV the range of hue is [0,179], saturation ranges from [0,255] and the range of value is [0,255]. Those scales are varying in different software so if comparing this in OpenCV, the ranges must be normalized properly [15]. The first step for detecting eyes for our research is to convert the input frame to HSV color [16], [17]. As the HSV model is known as a cylindrical presentation of the standard RGB model. In order to convert frame from BGR to HSV, each and every pixel of the frame is subjected to the following transformation [18].The max and min values for R, G, and B values are C_MAX and C_MIN which should be calculated and their difference M should also be calculated. Equation 3.1 shows the calculation of hue:

$$H = \left\{ \begin{array}{ll} 0, & \text{C\_MAX} = 0 \\ 60\times\frac{G-B}{M}, & C\_MAX = R \\ 60\times\frac{B-R}{M} + 120, & C\_MAX = G \\ 60\times\frac{R-G}{M} + 240, & C\_MAX = B \end{array} \right\} \tag{3.1}$$

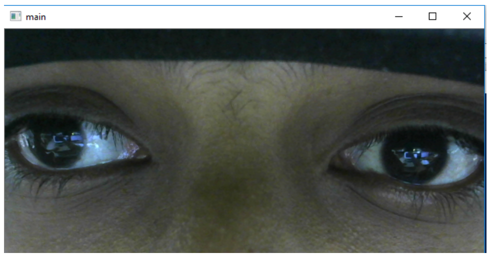
Equation 3.2 shows the calculation of value:

$$V = C\_MIN \tag{3.2}$$

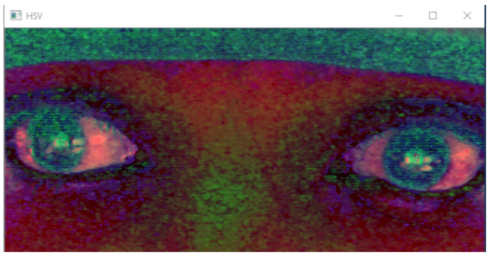
Equation 3.3 illustrates the calculation of saturation:

$$S = \left\{ \begin{array}{ll} 1\times\frac{M}{C\_MAX}, & C\_MAX = 0 \\ 0, & \text{C\_MAX} = 0 \end{array} \right\} \tag{3.3}$$

So, if we calculate the values for each and every pixel we are getting our input frame in HSV model. Now we know how this color space actually works so let's take the perspective of our research where we want to detect black colored object so we have to go through in some process. First of all, in HSV it is always easier to represent a color than BGR model. So we have to take the input frame from web cam and then it should be converted to HSV from BGR color-space. There is a threshold range for detecting the black color. In python the conversion can be one by writing a single line as the operation are performed by default.



(a) Before BGR to HSV Conversion



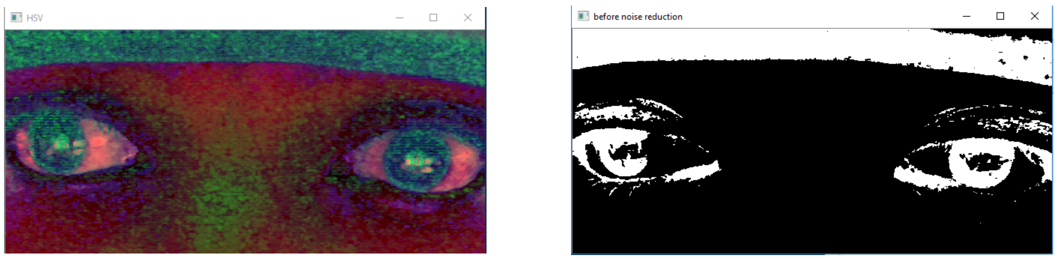
(b) After BGR to HSV Conversion

Figure 3.6: Before and After Result of BGR to HSV Conversion

With the help of the proposed process, it will convert HSV model into BGR color space. As well as, here are some parameters and the first one is frame which is basically the actual which is going to be converted. Any image or any video clips can also be taken in that parameter. Later on, the next parameter is describing that the frame basically the source is going to be converted into HSV from BGR color space. We have shown in Figure 3.6a the previous visualization of the frame before using BGR to HSV conversion and in Figure 3.6b the after view of the conversion.

3.3.3 Masking

Masking in OpenCV is known as filtering. Basically, this is one kind of filtering technique which is also known as spatial filtering. So, masking is a filtering operation which is applied on an image. Here to describe the process of masking involves setting some pixels value to "0" or in some other values related to the "background". Masking can be performed on an image in two different ways [19]. The two processes are using an image as mask or using ROIs as mask where in terms of ROIs, masking can be performed on the whole ROI or on the slices of ROIs. It depends on the desired outcome. To apply masking in an image means using the filtering mask in each pixel of that image. In each pixel (x, y) the outcome of the mask is shown by a predefined operational relationship. All the values of the filters are set on a standard by default [20]. Here, we are performing masking on ROI. Whenever, performing masking based on ROI there are two ways which are soft masking and hard masking. In soft masking the resulting intensity depends on how many pixels are lying under the masking area. If only a small amount of pixels are under the ROI then the outcome will become close to the background values and if most of the pixels are under ROI then the result will become be unaffected.
Here, the masking filter is going to be used for sharpness and noise reduction. For the better outcome of the research it is needed to remove the noise from the frame for large extraction of object. Then the topic comes is edge detection. For detecting edges sharpness is to be increased. For sharpness masking can also be performed. After having the gray scale image masking is going to be applied. The black portion will have values of "0" and the white portion will get "255". In the research, first of all we are converting the frame to gray scale then we are applying masking technique which is one form of bit wise and operation. In Figure 3.7a the state of the frame before doing masking and in Figure 3.7b the state after performing masking is shown.After that, sharpness will be increased as well as noise is going to be removed and then we are performing further functionality for actual outcome of



(a) Before Performing Masking (b) After Performing Masking

Figure 3.7: Applying Masking

the research.

3.3.4 Gaussian Blur

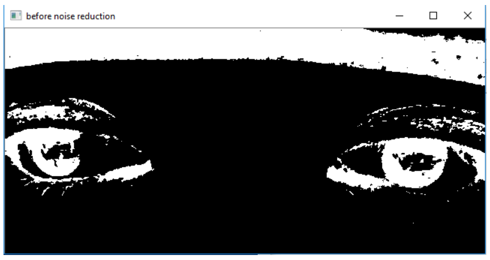
Image filtering is undoubtedly an important image processing tool where fast and effective response is required. Moreover, when the kernel increases we need an efficient filtering method so that the process does not becomes slow in computation. Gaussian blur is an adaptive method for blurring image using Gaussian function in image processing. This is basically used for reducing noise and details from an image which is mostly used in graphics software. The technique of this effect is a smooth blurring where the image is viewing throughout the translucent screen which is different from shadow of an object under usual illumination. Mathematically applying Gaussian filtering is same as smoothing the image with Gaussian function which reduces the higher frequency of an image as this can be defined as a low pass filter [21]. This filtering method can use in both one and also in two dimensions where for one dimension the function is illustrated in Equation 3.4 .

$$G(X) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{\frac{-x^2}{2\sigma^2}} \tag{3.4}$$

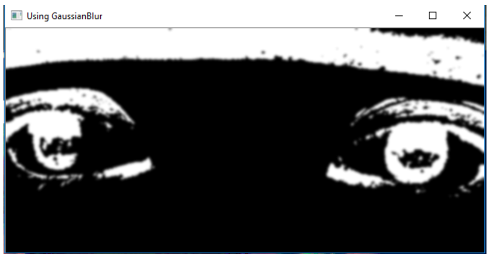
For two dimensions the Equation 3.5 is shown below.

$$G(X,Y) = \frac{1}{\sqrt{2\pi\sigma^2}} \times e^{\frac{-x^2+y^2}{2\sigma^2}} \tag{3.5}$$

Where X is the horizontal distance from the axis and y is the vertical distance and both of them are calculated from the origin accordingly is known as the standard deviation. In two dimensional formula there is going to be a surface where the contours will be concentric circle with Gaussian distribution and point will be calculated from the center. The values getting from the distribution are going to be used for building a convolution matrix that will be applied in actual image. Each pixels of the image will be set to an average value of the neighborhood pixels. The value of the original pixels will be highest and as the value is going to increase so the value of neighborhood pixels will be the smallest one. It results blur which contains boundaries with edges and it is better producing uniform blurring filters [22]. If you apply the multiple successive Gaussian blurs you will get the same result after applying the larger single Gaussian blur where radius is the square root of the sum of the squares of the blur radius. For example, if we apply the successive Gaussian blur with radius 8 and 6, we are going to get actually 10. So if we perform single



(a) Before Using Gaussian Blur



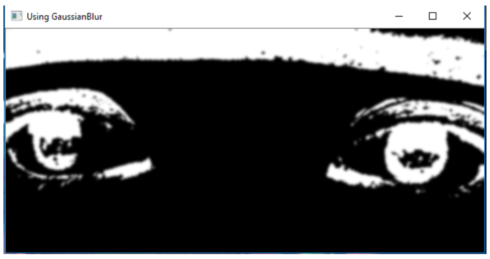
(b) After Using Gaussian Blur

Figure 3.8: Gaussian Blur Filtering

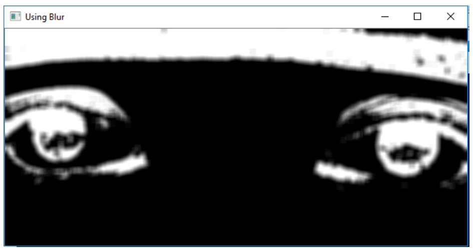
Gaussian blur with radius 10 then the response and calculation time will not be that much longer as it takes in successive Gaussian blur.
For instance, using it in python OpenCV the parameters and the restrictions must be fulfilled. Basically in the method cv2.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]]). Here each of the parameters carry meanings and first of all the src is the input image that we want to process, which is processed independently but it must have the depth in range. Then comes the dst which means the output image. Later on, ksize reflects the size of the kernel, both the height and width values need to be set as well as both can be zero where all the values of Gaussian kernel size is computed from sigma. SigmaX and sigmaY reflects the Gaussian deviation along with x-axis and y-axis. If sigmaY is set to zero then it means it is equal to the values of sigmaX. If both of them are set to zero then those values will be calculated from k-size respectively. Finally, borderType means the pixels extrapolation method which can be set as BORDER DEFAULT. After performing this on the frame we can see result mentioned below in Figure 3.8a we showed the previous version of Gaussian blur filter and in Figure 3.8b the frame after using the filter.

3.3.5 Averaging Blur

There are so many type of blurring methods in image processing tools as mentioned earlier and one of them is average blurring. In applying an averaging blurring filter we consider each pixel of the image one at a time. Here we consider rectangular shape of pixels around the pixel needed to be filtered out. The group of pixels are known as kernel and moves with the pixels that is being filtered [23]. The filter pixels will be in the center always along with the values of kernel and the height also the width of the kernel must be odd. To apply the filter to a selected pixel, the values of kernel are averaged and the averaged value becomes the new value for the filtered pixel. The larger the value of the kernel the image is more accurately blurred in comparison with the lower value of kernel as the larger values are more factored into average. To illustrate the process let's consider a seven by seven segment where we are going to get seven values in both rows and columns. As this is an averaging filter so its gong to sum up the entire row separately. Then it's going to divide the sum with the no of pixels in the kernel and there will be a new value for the center pixel. However, the same process will continue to the next pixel and we can compare it with a while loop as it will stop its function when it will go to the end of pixel of the actual image [23].



(a) After Using Gaussian Blur



(b) After Using Averaging Blur

Figure 3.9: Comparison Between Gaussian Blur and Averaging Blur

Now here comes one more interesting thing about this filter and this is what it will do when we are talking about the pixels which are near to the edge of the image since the kernel of the actual image may be partially off. Let's think about an upper-left corner of an image, how it will process its function. Basically, the default nature of the filter is to fill the all other missing pixels and it's going to do the same thing when it's going to get an off value for a kernel because it is mentioned earlier that there is off value for the pixel. If we can fill the missing pixels in a short time then the response time will be improved accordingly the filter will work more smoothly. There are built in methods in OpenCV so we do not need to perform any mathematical operation here for blurring. This blurring method takes two parameters and in python it is like blurred = cv2.blur (img, (k, k)) [24]. Here the img is the actual image and second we pass a tuple that describe the kernel which is essential for blurring of the image. In Figure 3.9a we showed the frame using Gaussian blur filter and in Figure 3.9b the frame using the averaging blur filter where we can see the averaging blur filter is more smooth than Gaussian blur.
The two parts of the tuple describes the height and the width of the kernel to use in pixels and both of them need to be odd so that the blurred is clearly defined for center pixel. But the thing is both the height and the width of the tuple is not necessarily to be same and there is also a chance for modifying behavior of the filter by adding another parameter which is known as replicate function or wrapping function [25]. This is mainly for the border area of the image.

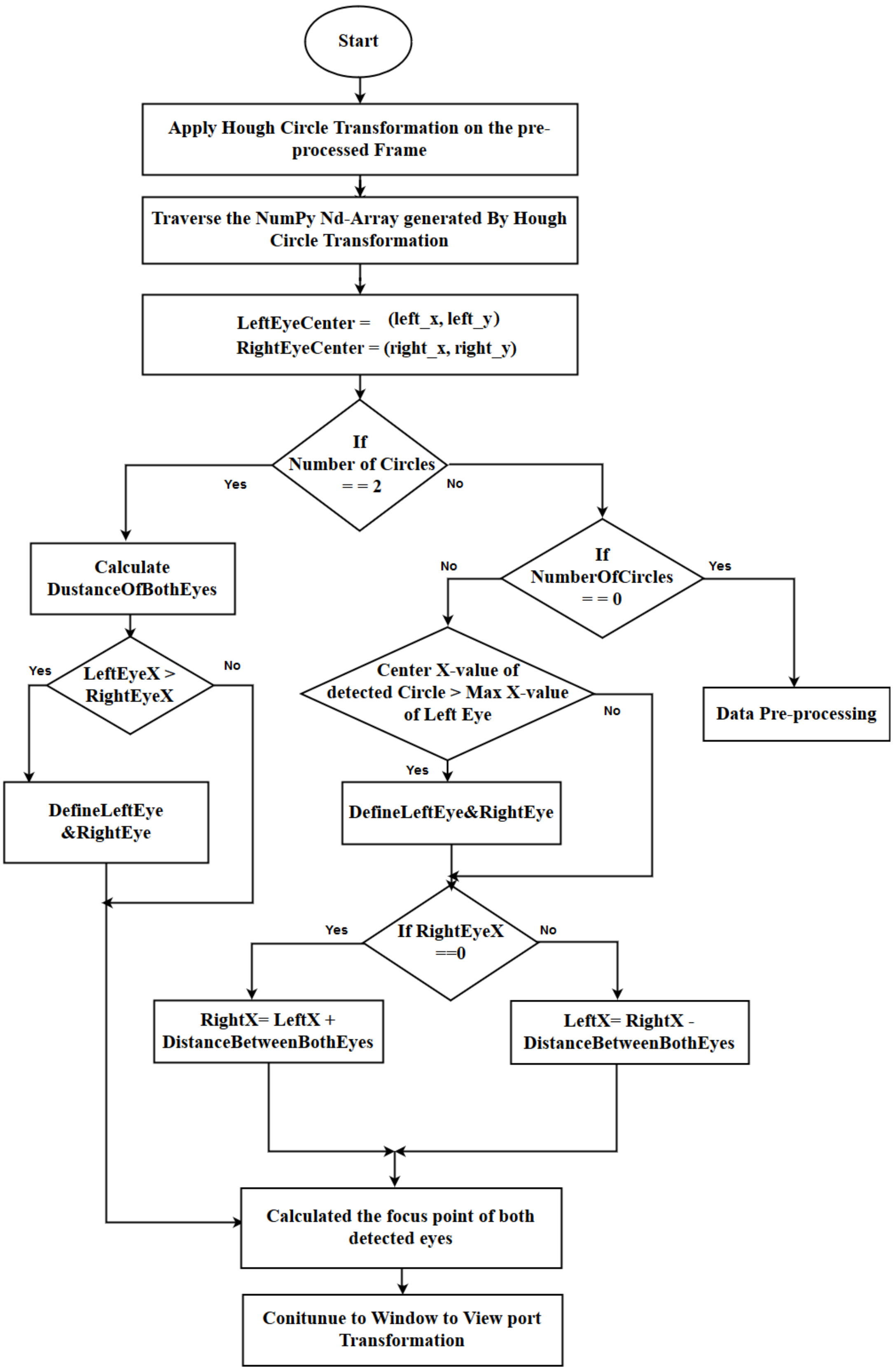
3.4 Algorithms

In the detection process shown in Figure 3.10, we have run the Hough circle method over the smooth image done by Open CV blur function. The Hough circle returns a numpy Nd-array with circle center co-ordinates. Then from the array value we determined the left and right eye. If one eye is not found, our program will adjust other eye position calculating the pupil distance. Afterwards, the iris circle position is converted to actual view port of the system.

3.4.1 Hough Circle

Hough circle transform is a digital image processing technique. It helps to detect circular objects in a digital image. In two dimensional surfaces, circle is described by the Equation 3.6.

$$(x-a)^2 + (y-b)^2 = r^2 \tag{3.6}$$



Start
Apply Hough Circle Transformation on the pre-processed Frame
Traverse the NumPy Nd-Array generated By Hough Circle Transformation
LeftEyeCenter = (left_x, left_y)
RightEyeCenter = (right_x, right_y)
If Number of Circles = = 2
Yes
No
Calculate DustanceOfBothEyes
If NumberOfCircles = = 0
No
Yes
Data Pre-processing
LeftEyeX > RightEyeX
Yes
No
DefineLeftEye &RightEye
Center X-value of detected Circle > Max X-value of Left Eye
No
Yes
DefineLeftEye&RightEye
If RightEyeX ==0
Yes
No
RightX= LeftX + DistanceBetweenBothEyes
LeftX= RightX - DistanceBetweenBothEyes
Calculated the focus point of both detected eyes
Conitunue to Window to View port Transformation

Figure 3.10: Iris Detection and Adjustment

Where (a,b) is the center of the circle, and r is the radius. The total operation is done by voting. At first the image frames are prepared by Gaussian blur and then the canny edge detection is done. After that, there are several checks on detecting and voting the circles. Firstly, all the variables (a,b,r) are set to 0. After that, for each x and y pixel there is a specified parameter for radius of the circles. Under the range of circle radius, a specific theta value is given by the user inside method parameter. Afterwards, the polar co-ordinate of the center calculation is done by Equation 3.7 and Equation 3.8.

$$a = x - r \times \cos(\frac{t \times \pi}{180}) \tag{3.7}$$

$$b = y - r \times \sin(\frac{t \times \pi}{180}) \tag{3.8}$$

The detected circles are voted and the local maximum voted circles of accumulator gives the circle Hough space. On the other hand, the maximum voted circle of accumulator gives the circle. The returned result is in [a,b,r] format, (a,b) is the position of the center of the circle and r is the radius. In Figure 3.11, the mentioned image shows the detected two Hough circles based on the given parameters. The returned numpy nd-array will provide co-ordinates of detected two iris circles.

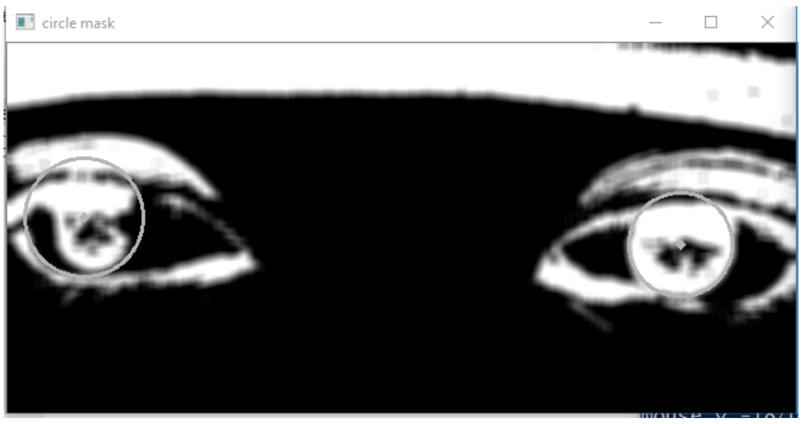


Figure 3.11: Applying Hough Circle Transformation

The hough circle method is: cv.HoughCircles(image, circles, method, dp, minDist, param1 = 100, param2 = 100, minRadius = 0, maxRadius = 0) [26]
This is an OpenCV built-in property. We need to provide the actual parameters and it will return actual circles with positions. In our system, we have masked the region of interest by color filtering. We have set lower and upper threshold value to detect eyes as the iris portion is darker. The dp (inverse ratio of the accumulator resolution to the image resolution) is set to 1. This means the accumulator has the same resolution as the input image. The minDist is set to 370 which is the actual distance between the centers of the detected circles. This is actually the difference of two eyes focal points. The param1 is the parameter for canny edge. High value detects less edge and low value detects more edge. To determine only the rounded iris edge we have placed the value 370. The param2 is the accumulator threshold value of circle center. The smaller it is, the more false circles will be detected. We have placed 20 in our case. For the iris localization, we have set a minimum radius of 40 and maximum of 50. This parameter helps us to skip false or wrong circles in the frame.

3.4.2 Data Processing

After using Hough Circle Algorithm we find the circles value [X Y R] in a NumPy nd-array form. From this array we had to differentiate the x and y position values. For this, at first we converted the array into a list which gives us output in this way –

[[[CircleOneX, CircleOneY, CircleOneR],[CircleTwoX, CircleTwoY, CircleTwoR]]]

Afterwards, we are taking the first item of each index and creating a new array. For example- the new two arrays are-

[CircleOneX, CircleTwoX] and [CircleOneY, CircleTwoY]

At last, we are traversing each array and storing the values as stated-
LeftX = CircleOneX
LeftY = CircleOneY
RightX = CircleTwoX
RightY = CircleTwoY

3.4.3 Iris Adjustment

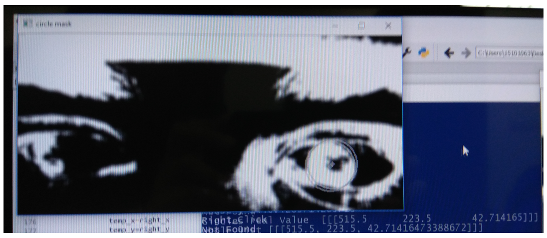
In our system after detecting the circles using Hough Circle algorithm, we get the values of left and right eye. At this time we faced some problem which are it is not fixed that which eye will be detected first and stored in the first position as left eye of the array. We had to differentiate between the eyes values. To solve this problem we proposed a method which will detect which circle is left eye and which one is right eye. Also another problem is, if one eye is detected only, how the cursor will move then. In our system we have developed a process by which these problems can be solved.

Define Left and Right Eye

After processing the detected circles value, we have to check whether the values are saved in the right position. When the circles are detected from the video capture, it is not static that left eye will be perceived first and then right eye. The algorithm detects circles whenever they find the required values. Nevertheless in our system, we need to store the left circles value foremost and then right circle's center position. Therefore, to check if the values are stored in accurate position or not we are using the stated technique. In this procedure, we compare between the values of x-axis of both circles. If the first circle's x-value is less than second circle's x-value than the first circle is stored as LEFT EYE otherwise we need to swap the values as circle's x point is less than first circle's x point.

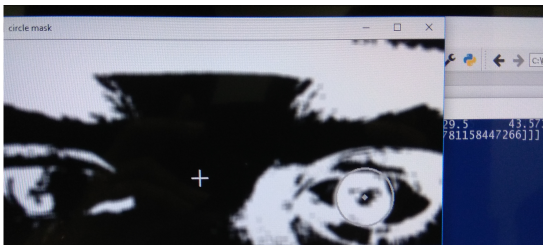
```
If Left_X <Right_X:
DefineLeftEyeRightEye()

DefineLeftEyeRightEye() :
TempX = Right _X
TempY = Right_Y
```



```
176        temp_x=right_x
177        temp_y=right_y
RightClick Value  [[[515.5       223.5        42.714165]]]
Not Found  [[[515.5, 223.5, 42.7141647338867 2]]]
```

(a) One Eye Detection



```
29.5          43.57
781158447266]]]
```

(b) One Eyed Fixation

Figure 3.12: One Eyed Fixation

Right_X = Left_X
Right_Y = Left_Y

The purpose behind this condition is, as the display screen is divided into pixels, the left eye's pixels position must be smaller than the right eye's pixel position, as left eye appear first. Because of this reason, we are only checking the x-axis value of the both centers rather than the y-axis value.

One Eyed Detected

Another problem showed up after detecting the circles is, whenever we are looking at any corner, our one eye is going out of focus. As a result we are getting only one circle from the circle detection algorithm. In addition, only one circle's center value cannot determine the exact mouse position since we are taking the average of the both circles midpoint. If only one circle of detected then the system will assume another circle's center is (0, 0) and resulting a wrong average of both circles, where only one circle is found. For instance, if expected cursor position is near (582,720) then our eye's position would be near left eye = (350,720) and right eye = (815,720). But if only one left circle is found, the cursor position will be at near (175,720). As a result, cursor pointer will be in another direction.
Hence, to solve the problem shown in Figure 3.12a, we have proposed another method which we named as "OneEyedFixation". In this method whenever we find only one circle, the system assumes the other one by calculating the eye distance. Every person both eyes are positioned at a fixed distance which can never be change. So when the program starts, we stores the distance of both eye of the user and to find the accurate value, the user have to look at any point near the middle of the display screen as we need both circles to be detected. After that whenever only one circle is detected, another circles is assumed by positioning the centers at the determined distance. Again there are two possibilities of getting only one circle. We have to check whether the detected circle if left eye or right eye. The reason of this condition is if we found left eye, we can add the determined distance with the left x value and assume that it would the right eye's center x. But if right eye is detected, then the distance need to be decremented from the detected center x as left eye must be in the left side of our detected eye position.

DistanceOfBothEye = RightX - LeftX
RightX = LeftX + DistanceOfBothEye
LeftX = RightX - DistanceOfBothEye

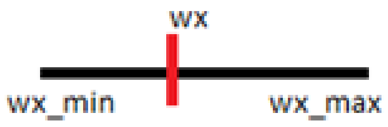
Table 3.1: Coordinates of Window and View Port

Window Coordinates	View port Coordinates
wx_min = Minimum x axis value	vx_min = Minimum x axis value
wy_min = Minimum y-axis value	vy_min = Minimum y axis value
wx_max = Maximum x-axis value	vx_max = Maximum x axis value
wy_max = Maximum y-axis value	vy_max = Maximum y axis value
(wx, wy) = Selected point	(vx, vy) = Mapped value of the point

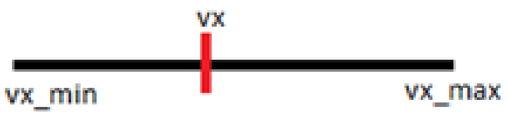
For this, we need to find out first that which eye is detected. To ensure that the circles detected was saved in accurate place we compared the values with pixels. We have fixed the position the ROI frame which is (0, 0) so there is a certain range for left eye to be appeared. As our frame is fixed for any user, we can check for the left eye appearance pixel range. If the values detected is in the range then the detected circle is left eye otherwise it is the right eye. Then again, if the right eye is detected then we have to call our first method to swap the left and right eye value, because as only right eye is detected, the value must have been saved in the first position of the array. By using this two proposed method we can avoid all kind of error position of our cursor and make it move smoothly. After using the proposed method we can see the results in the Figure 3.12b.

3.4.4 Window to View port Transformation

In world coordinate area window is defined as the rectangular space which will be displayed. Window can be smaller or larger than the actual display depending on the requirements. On the other hand, view port is an area where the frame is mapped and will be demonstrated. Window to view port mapping is the process of transforming two-dimensional coordinates into a rectangular screen [27]. The variables we used in the transformation process are familiarized in Table 3.1.
For instance, if we want to locate a coordinate (wx, wy) from window then view port places the points on the display. In order to map the window in the view port some calculations are required as the window may have to increase or decrease the size according to the view port. This transformation develops formulas to find the (vx, vy) and this starts with the values of (wx, wy). If (wx, wy) is 45 percent away from the x-axis of the window port showed in Figure 3.13a then (vx, vy) will also be 45 percent away from x-axis of the view port which is visualized in Figure 3.13b.



(a) A Point (wx) in Window port



(b) Proportional Point (vx) in View port

Figure 3.13: Pixel Transformation

In Figure 3.14 we can see the positions (wx_min, wy_min), (wx_max, wy_max), and (vx_min, vy_min), (vx_max, vy_max), are relative in window and view port. If we locate a point (x, y) in window the relative point in view port will be (u, v). There are three steps in window to view port transformation which are explained below.

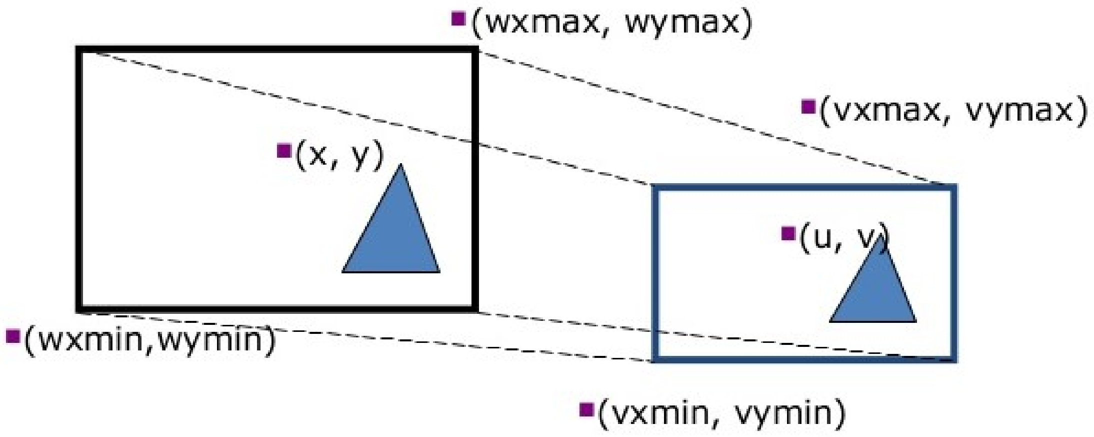


Figure 3.14: Window to View port Transformation

Step 1: First we have to translate the window to origin (0, 0) and then the translation factor will become negative (-Tx, -Ty) as we are moving the values of window to the left. In Fig 3.13 we can see we moved Figure 3.15a to Figure 3.15b. The lower left and upper left corner changes respectively (- wx_min, - wy_min) and (- wx_min, - wy_max). Translation factors are shown in Equation 3.9 and Equation 3.10.

$$Tx = \frac{(wx\_max \times vx\_min) - (wx\_min \times vxmax)}{wx\_max - wx\_min} \tag{3.9}$$

$$Ty = \frac{(wy\_max \times vy\_min) - (wy\_min \times vy\_max)}{wy\_max - wy\_min} \tag{3.10}$$

Step 2: In this step the window size is re scaled to view port size using the scaling factor (Sx, Sy) shown in Figure 3.15c. The scaling factors are illustrated in Equation 3.11 and Equation 3.12.

$$Sx = \frac{(vx\_max - vx\_min)}{(wx\_max - wx\_min)} \tag{3.11}$$

$$Sy = \frac{(vy\_max - vy\_min)}{(wy\_max - wy\_min)} \tag{3.12}$$

Step 3: The third step is about translation. Window is translated to the view port size. If the lower left corner of the window is (0, 0) then third step is not required but if not then the translation factor will be converted into positive (Tx, Ty). In Figure 3.15d the final translation is showed.
After determining the translation and scaling factor we can map any point of the window into our view port by the following Equation 3.13 and Equation 3.14.

$$vx = (Sx \times wx) + Tx \tag{3.13}$$

$$vy = (Sy \times wy) + Ty \tag{3.14}$$

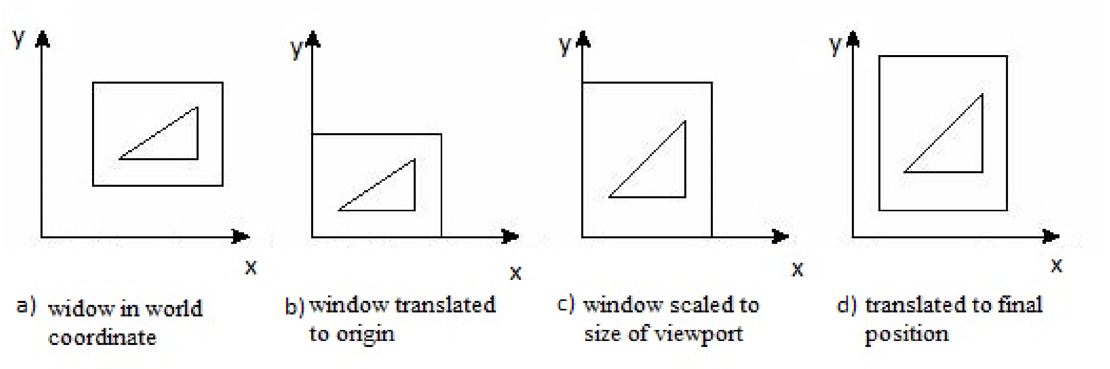

y
x
a) widow in world coordinate
y
x
b) window translated to origin
y
x
c) window scaled to size of viewport
y
x
d) translated to final position

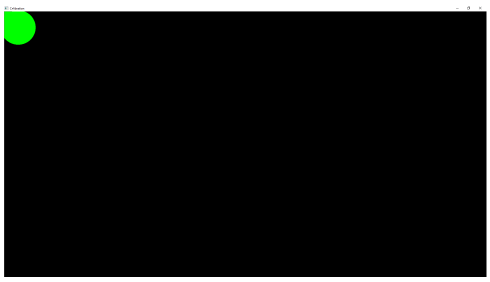
Figure 3.15: Steps of Transformation

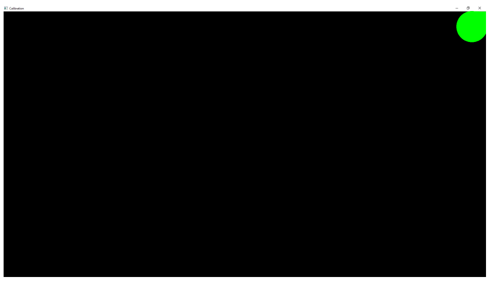
Window Port Determination

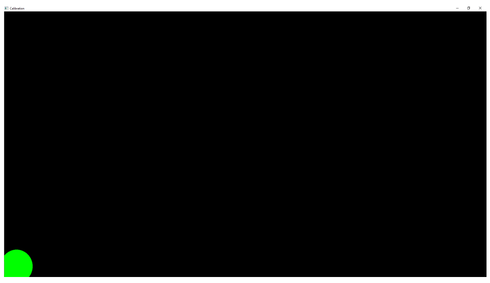
In our system we have used this transformation for position the cursor in the monitor screen. Our cursor position is totally dependent on the iris's midpoint. First, we had to find the range which is covered by our iris. There is a fixed distance which can be covered by the midpoint of our iris. As we are taking the average from both irises, the range or the area of our window would be between our both eyes. To determine the window of our system we used a calibration method to get the exact minimum and maximum point of the covered area by our iris.

Calibration

Calibration is a process of defining some unknown values with the reference of known values. This procedure ensures the accuracy of the dimensions determined by the reference value. At the beginning our program four windows will be displayed for five seconds. These windows show four different points at the corners of the monitor. Each point of the corners is exhibited for the certain time and meanwhile the user look at those points. Here, the upper left corner point in Figure 3.16a determines the minimum value of x-axis and y-axis and the lower right corner point determines the maximum value of x-axis and y-axis. Since the computer screen pixel values origin from the upper left corner (0, 0) and ends at the lower right corner ((1920, 1080) for 1k display).
In the meantime, when the user looks at the four points the average values of both eye irises' midpoint is stored as the window's coordinate. When the user gazes at the upper left point, the average value of the iris's midpoint is stored as the minimum x and minimum y for the window. After that, upper left pointer window destroys and upper right pointer showed in Figure 3.16b window shows. From this window we can find the maximum x-axis value. After the second window destroys within another five seconds, the third window with lower left pointer showed in Figure 3.16c appears which find the value of maximum y-axis.
Subsequently, the fourth window appears with the pointer of lower right showed in Figure 3.16d and stores the value of maximum x and y-axis. The whole process took twenty seconds overall. From this calibration we found the area of our specific window within a short amount of time. Each time the program runs this calibration process took place to ensure the window coordinates, as the window's accurate value results the cursor's smooth movement. Once we get the accurate window coordinate


(a) Upper Left Corner


(b) Upper Right Corner


(c) Lower Left Corner

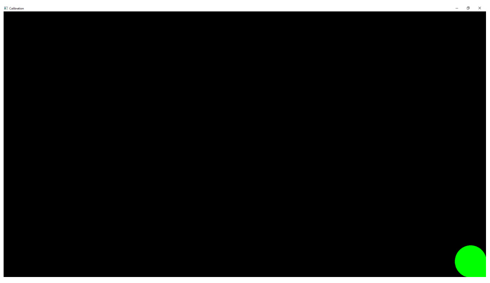

(d) Lower Right Corner

Figure 3.16: Process of Calibration

using this calibration method, no further delay is necessary in our system.

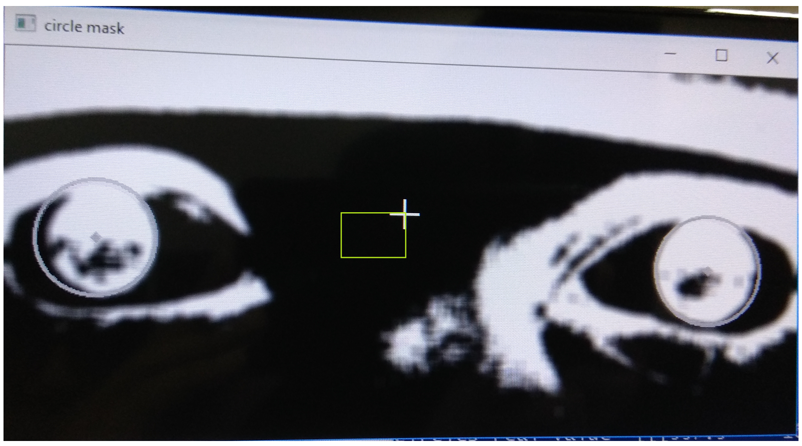


Figure 3.17: Window of the Cursor

In the Figure 3.17, we have shown the actual window size where the eye focal point moves on. As our frame is always fixed the pointer will not go out of screen. The conversion is handled such a way that, the max_x and max_y is always inside this rectangular window. Afterwards, this window is converted to user's required view port and the cursor moves on depending on that view port size.

View Port Determination

View port in our system varies, as user can use different size of displays. There are many display with different resolutions such as- HD (High Definition), FHD 1k (Full High Definition), QHD 2k (Quad High Definition), UHD 4k (Ultra High Definition) and Quad UHD 8k which are showed in Figure 3.18. At first our system determines

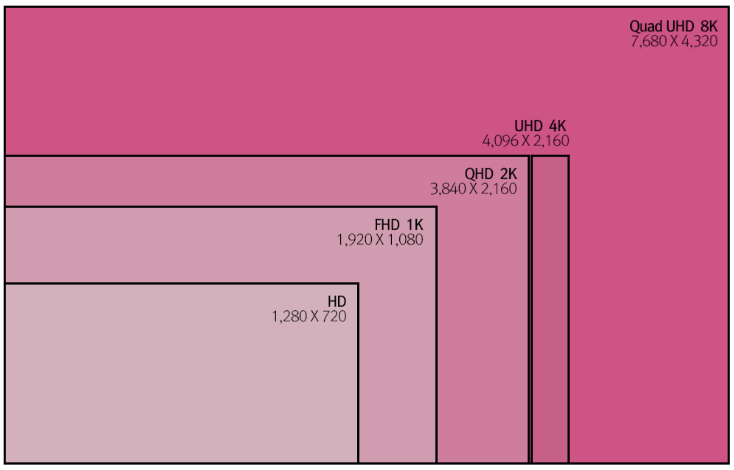


Figure 3.18: Different Resolution of Display

the window size using the calibration method, then it measures the width and height of the device display.
After determining the window and view port of the system, it uses the window to view port transformation equation to map the point of our iris's average midpoint into the display screen where it appears as the cursor movement. For instance, if we look at the middle of the display screen our iris's average middle point will be at a certain position in the determined window. After the transformation equation applied, the very same point will be positioned in the view port's middle point. Moreover, consuming this conversion we can use the system for any display screen without any changes.

3.4.5 Cursor Movement Control

Midpoint line drawing algorithm is used to draw lines between any two pixels in a computer screen. As we need to move the cursor throughout the whole computer screen arbitrarily, we have carefully chosen this algorithm because it is quite simple, only requires integer values and straightforward calculations. In this algorithm there is no multiplications or division which makes it more time efficient [28]. For our system instead of drawing lines we moved the cursor from one point to another. Whenever we will move the cursor pointer to the next position, this algorithm can choose the pixels closest to the actual line with high accuracy, consistency and straightness. After getting the relative center points of left and right eye in the view port we have calculated the FocusPoint(x, y) of our eye using Equation 3.15 and Equation 3.16. The middle point of both centers x-coordinate and y-coordinate is the main focus point. Initially the cursor is on the (0, 0) position. We stored both values of cursor, previous pixel and next pixel to move it smoothly from one place to another. First we had to round up the cursor positions, as the focus point values can be in floating state after calculation and then the cursor moving algorithm starts working.

$$FocusPoint(x) = \left(\frac{LeftEyeX\_Coordinate + RightEyeX\_Coordinate}{2}\right) \tag{3.15}$$

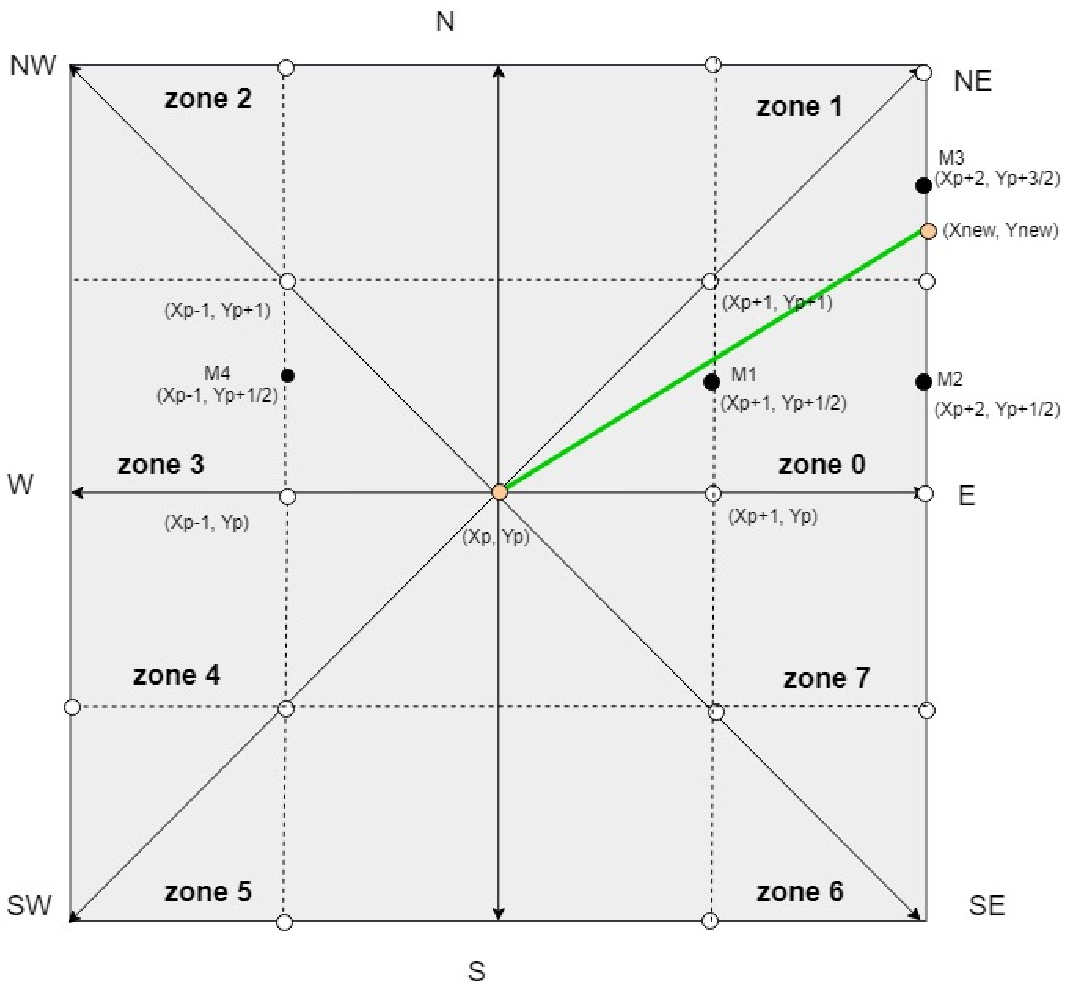


Figure 3.19: Grid visualization of cursor movement

$$FocusPoint(y) = \left(\frac{LeftEyeY\_Coordinate + RightEyeY\_Coordinate}{2}\right) \tag{3.16}$$

At first, the algorithm finds two possible pixels nearby the line from the previous cursor's position. If cursor's previous and next pointers are respectively (Xp, Yp) and (Xnew, Ynew) where Xp is less than Xnew then the two possible next pixels can be E (Xp+1, Yp) and NE (Xp+1, Yp+1), where E and NE represents East and North East. The middle point of E and NE is M1 (Xp+1, Yp+1/2), which is the decision parameter for this algorithm. The equation of the initial decision parameter is Dinit = dy - dx/2.
Here, dx = Xnew – Xp and dy = Ynew – Yp.
If Dinit is greater or equal to zero then chooses NE (Xp+1, Yp+1) otherwise chooses E (Xp+1, Yp). In this way each pixel between the cursor's previous and next point is selected to move the pointer accurately. This is only explained for one zone, which can be called as zone 0. In Figure 3.19 all the eight zones are shown. These zones cover every side of the computer screen. We can move the cursor in each direction accurately. If the (Xnew, Ynew) is less than (Xp, Yp), the cursor will move to backward which means either it is under zone 2 or zone 3. Depending on the zones the midpoint algorithm will work accordingly.
In order to determine the zone we have to compare differences between the x and y-coordinates. For instance if the difference between x-coordinates (Diff_X) and difference between y- coordinates (Diff_Y) are both greater or equal to zero then it is either zone 0 or zone 1. If the Absolute value of Diff_X greater than the absolute value of Diff_Y then it is zone 0. In this way first we find the zones and then based on the decision parameter the cursor move to the exact position pixel by pixel.

3.4.6 Blink Detection for Click Function

The integrated Haar-cascade cannot detect closed eyes. Therefore, we have used the classifier to detect the closed eye and run the click function based on that. For left eye closed, the left click function will work and for right eye closed the right click function will work which is shown in Figure 3.20.
We are using Haar-cascade_eye_tree _eyeglasses classifier to detect eye regions. Haar-cascade is a classifier which is used to detect objects for which it is trained for. This training procedure is done on servers and various environments. The Haar-cascade has a lot of trained sets by which we can detect various objects.
We are using the eyes classifier. This helps us detecting the eyes. One of the feature of this classifier is it cannot detect closed eyes. This feature has become one of our advantages. We are detecting the closed eyes state and using it as a blink detector. When this blink occurs there is a click function called. Haar-cascade Classifier returns the result with an array. We have analyzed the value of the array and found that for each eyes it provide some specific value changes in the array. For example, if we see this two array values [455 157 179 179] and [4 131 198 198], we will see that the first value is decreased which determines the left eye is closed.
By using this change of values, we have measured the eye's current state if it is closed or not. If the left eye is closed we call the left click function and for the right eye closed we call the right click function. In terms of both eyes found, we do the mouse release functions. Figure 3.21a shows the open eye is left side so the click performed would be right click. Similarly in Figure 3.21b shows the detected right

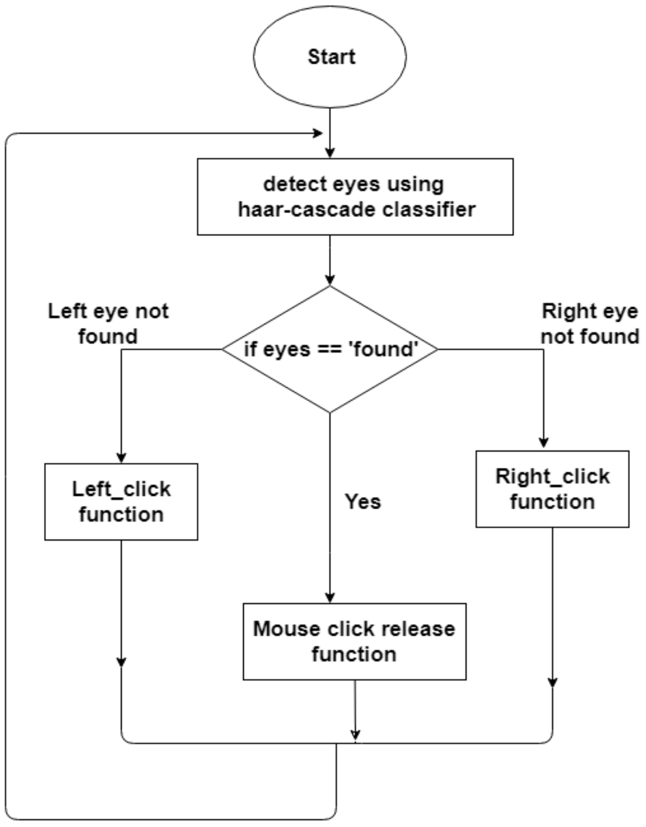
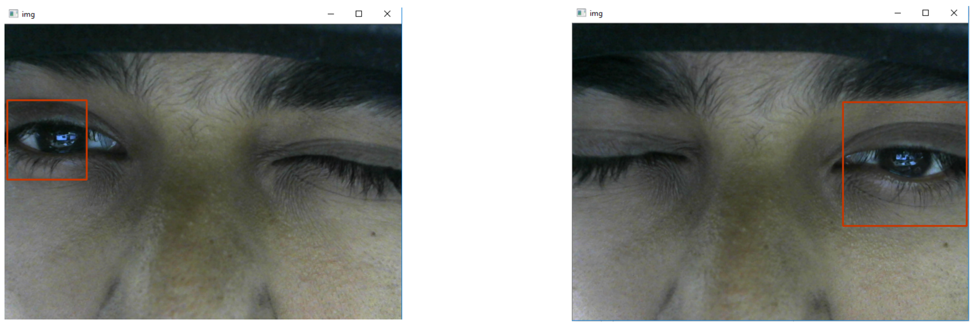


Figure 3.20: Blink Detection

eye which means the function would be left click . So based on this, we decided to measure blink and run click functions.
The reason behind using this library is, this library do not detect the usual blinks. As we know, usual blink takes 100-150 milliseconds [29]. So, in order to get detected the minimum times of closed eyes have to be around 1 second. If the left/ right eye provides the value of closed state for around 1 second then the click function will occur. Otherwise, that will be measure as normal human blink and that will be skipped.



(a) Right Click (b) Left Click

Figure 3.21: Blink Detection for Click Function

3.5 Work Flow

Here, all the above work flows are merged in Figure 3.22. After capturing the video frame, the detection process and the blink detection for click function will work in parallel processes. Therefore, the users will be able to move the cursor along with clicking their desired applications.

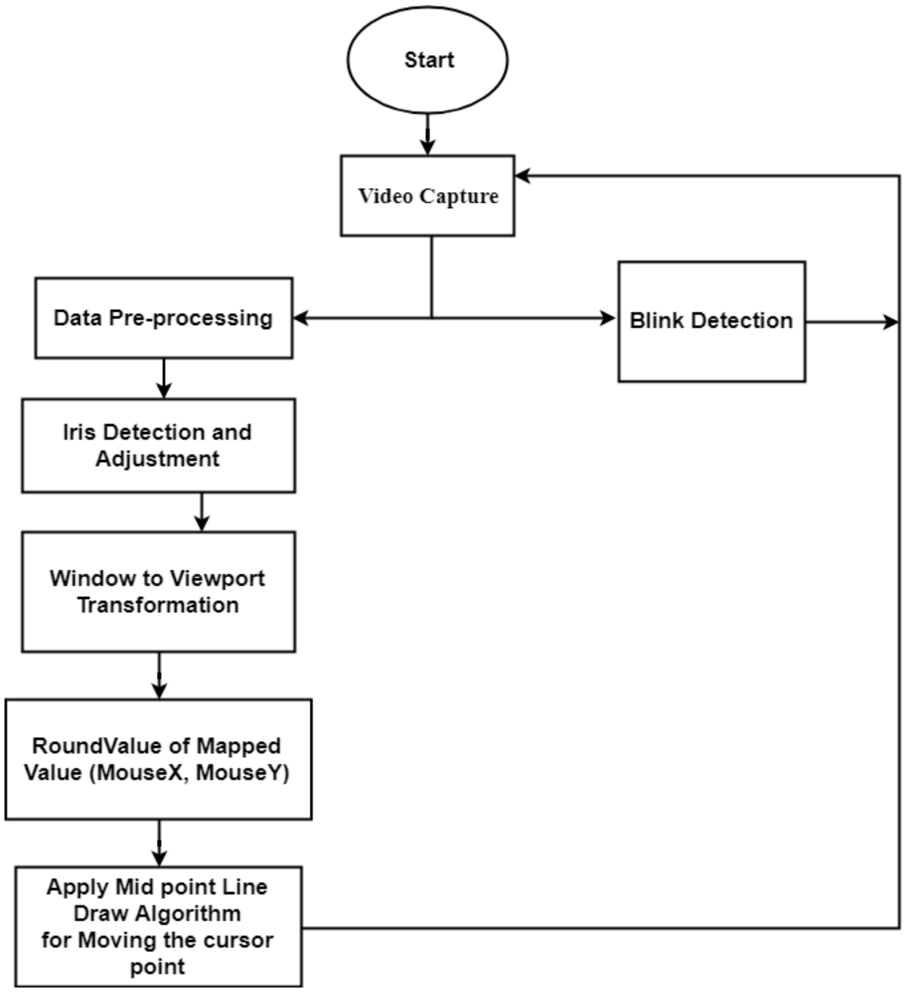

Start
Video Capture
Data Pre-processing
Blink Detection
Iris Detection and Adjustment
Window to Viewport Transformation
RoundValue of Mapped Value (MouseX, MouseY)
Apply Mid point Line Draw Algorithm for Moving the cursor point

Figure 3.22: Complete Work Flow

Chapter 4

Result Analysis

We let some group of people to use our system to perceive the accuracy. Therefore, to analyze the precision of cursor movement and blink detection we fixed some task and calculated the time for each group who would accomplish those tasks. Correspondingly we compared our system with the actual system which is using mouse.

4.1 Evaluate the Accuracy of Cursor Movement

On the way to evaluate the accuracy of cursor movement we selected three types of tasks to perform. These three tasks are – case 1, case 2 and case 3.In case 1, the user have to select a folder from desktop and then in that he has to delete a file. In case 2, the task which is going to be performed is to go to the windows defender and run a quick scan. Lastly, in case 3 the user have to refresh 2 times continuously. We tested these cases on three groups of people- teenage(13-19), adult(20-40) and old (40 and above). In each group we took 10 people for the testing. After that we stored the number of seconds they took while completing the tasks.
In Figure 4.1 we can see that, the same tasks performed by all groups took diverse amount of time. In order to perform case 1, teenager group took around 6.87 seconds, adult group took around 6.5 seconds and finally old group took 7.5 seconds. We can see that, adults took less time than other two groups. Although teenagers were little faster to cope up with the system than old group. There is also the comparison between our system and the actual system. We also performed the tasks by using our hands to see the actual time and compared it with the best value we got from previous evaluation which was stored by adult group.

4.2 Estimate the Accuracy of Cursor Pointer

In this investigation, we tried to show the cursor pointer accurateness. We stored the values of some selected points using the mouse. These points were selected by using some folders kept in the desktop. We noted down the values of the cursor when it select the folders. We took 10 values for this examination. Then we ran our system and tried to select those folders again with our eyes. For instance, if a folder is at the left corner of the monitor screen in desktop, using mouse we select the folder at pixel (18, 1074), And using our system we can select the folder at pixel

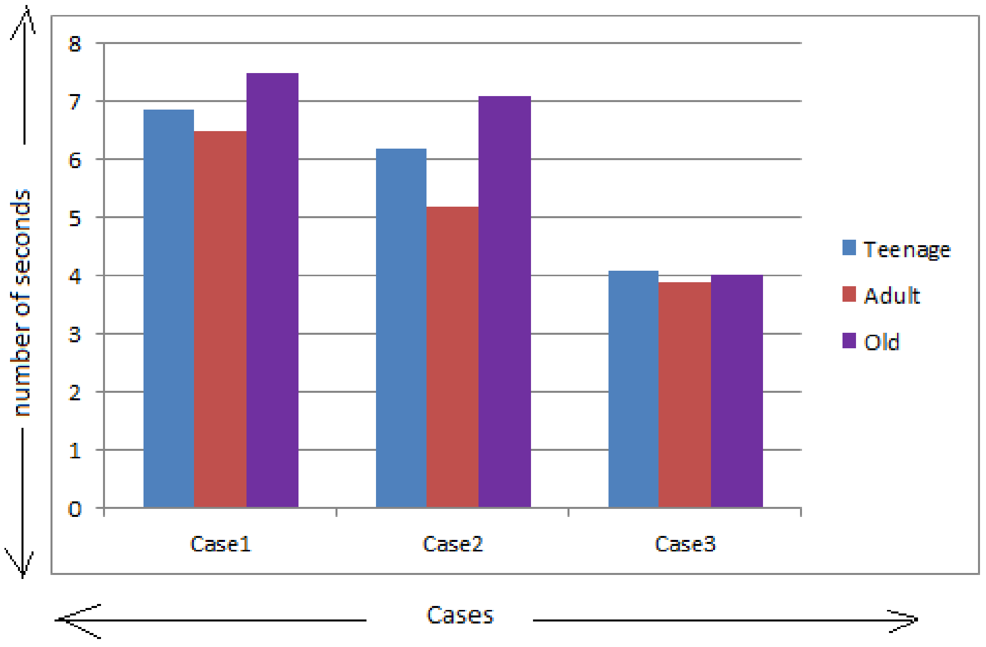


Figure 4.1: Estimate the Accuracy of Cursor Pointer

(17, 1065). The comparison between both the results are shown in Figure 4.2.

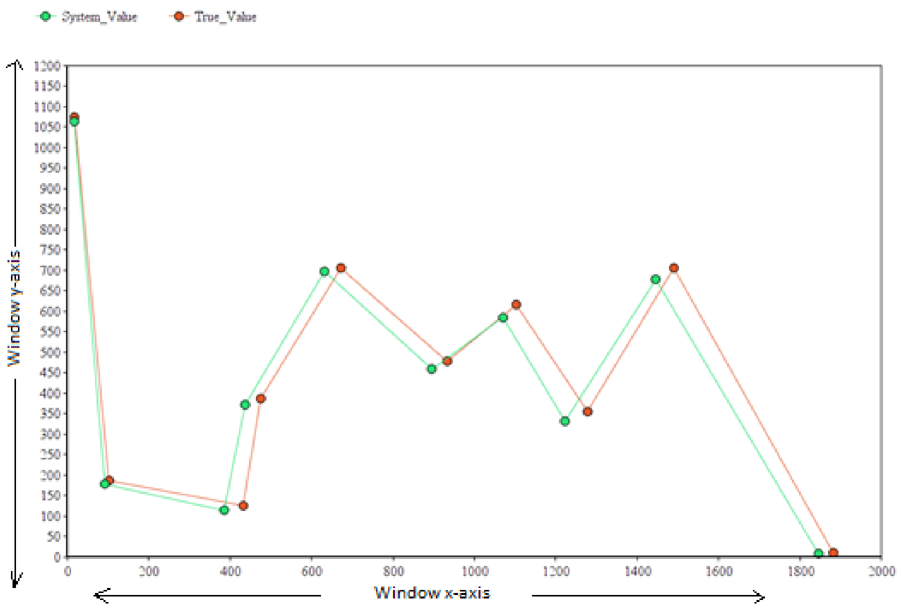


Figure 4.2: Estimate the Accuracy of Cursor Pointer

When we select the folders using our system, it did not give the same values as the mouse but both the results were close. The accuracy of individual x-coordinates and y-coordinates are also presented. In Figure 4.3 the accurateness of x-pointers and in Figure 4.4 the y-pointers are displayed.
Form the collected values of X-coordinates we can calculate the accuracy. The Equation 4.1 we can use to calculate the accuracy is:

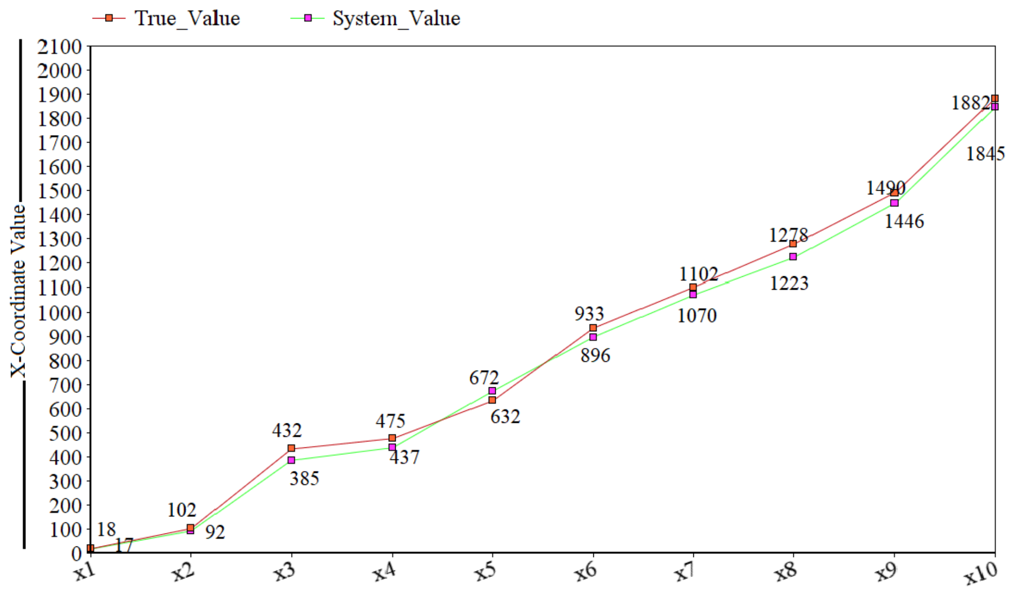


Figure 4.3: Points plotting for X coordinates

$$Accuracy = \frac{true\_value - system\_value}{true\_value} \times 100\% \tag{4.1}$$

After using this equation we are going to get the accuracy for x-axis and after that if we take the average value then we can end up with only one value. The value is indicating the accuracy for x-coordinates. However, after that we are getting 91.7% accuracy for the x-coordinates.

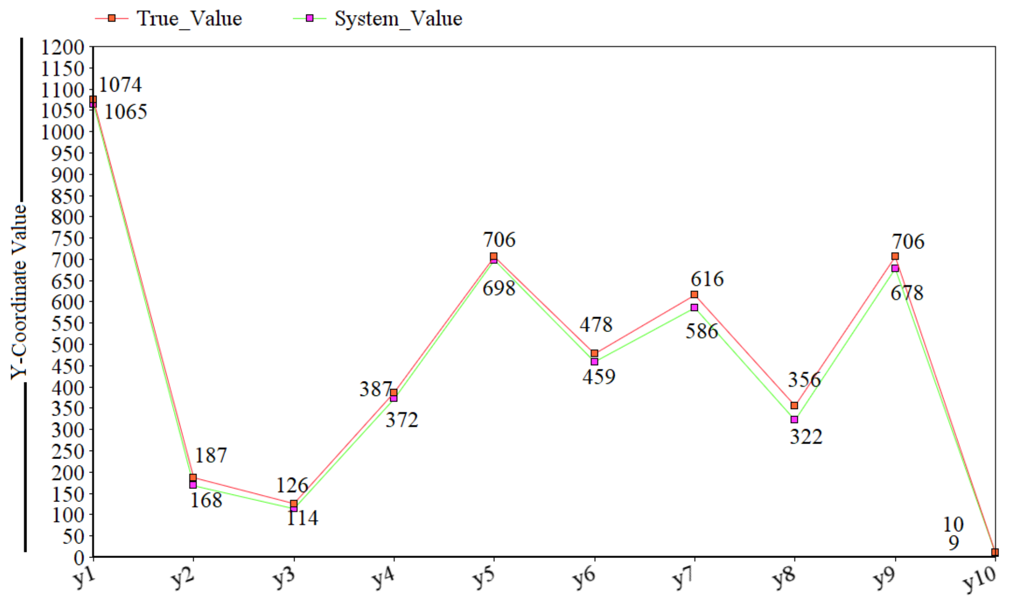


Figure 4.4: Points plotting for Y coordinates

Again for y-coordinates we are using the same equation which has been used for calculating the accuracy of x-coordinates. So, whenever we are going to averaging

Table 4.1: Accuracy of Blinks in terms of Distance

Distance	No. of Clicks	No. of Attempt	Accuracy
2.5 ft	15	18	83.30%
2 ft	15	17	88.23%
1.5 ft	15	17	88.23%
1 ft	15	16	93.75%

it the value of accuracy becomes 90.8%.

4.3 Accuracy of Blinks in terms of Distance

To evaluate the blink accuracy in terms of distance we performed click function staying at difference distances. For instance, the user sat at a distance of 2 ft and used our system. We tried to get 15 perfect clicks and to do so we had to click a little more. At a distance of 2 ft the user clicked 17 times to get perfect 15 clicks. Again, at a distance of 2.5 ft the user had to click 18 times to get the desired numbers of click. The values from this examination is shown in the Table 4.1. In the left most column we have showed distance, in the next column number of clicks performed is displayed and then number of attempts is showed. Lastly, we showed the accuracy at different distances.
As we can see from Table 4.1 in 2.5 ft the blink accuracy is 83.30%, in 2 ft the accuracy is 88.23%, in 1.5 ft the accuracy is 88.23%, in 1 ft the accuracy is 93.75%. As previously we have stated that distance does not matter in our system as the camera is fixed in position. From this evaluation we can state that, no matter where or how the user is sitting from his monitor, the accuracy will not be affected by the distance. It totally depends on the blink detection.

4.4 Resolution Invariant

We are using a calibration for all available screen and available systems. So, whatever the resolution is. it will not affect our system stability. For all kinds of display with various resolutions it will work completely fine. That's why this developed system is resolution invariant. We have worked on a PC in our research lab which has a 1080p monitor. We have placed a folder in a specific portion of the monitor and we tried to select and enter that folder. We measured the required time for that and was around 3.5 seconds. On the other hand, in our home we have 4k and 2k displays on which we have tested the same process and it took around 5 seconds and 4.2 seconds. So, from this test we can say that the system is not resolution specific. It took almost same time for all the operations and it is resolution invariant.

4.5 Comparison with Existing System

As there have been many previous work in this field, we ran a comparison with the previous existing system so show the improvements in our system. In Table 4.2 we showed comparison between Gaze Pointer system and our system. In Gaze Pointer

Table 4.2: Comparison between Existing System and Our System

Functionalities	Gaze Pointer	Our System
Distance flexibility	N/A	Available
OS independent	N/A	Available
Screen size independent	Available	Available
Detection over eye-glass	Available	Less Accurate
Calibration process	Lengthy	Short and simple
One eye usability	N/A	Available

the system is not distance flexible, as the user have to be in a fixed position to train the cursor pointer. Where in our system we used a cap with web cam, which gives the user the flexibility to move freely. The user do not need to be in a certain position to train or use the system, which is makes our system more user friendly. In terms of OS compatibility, we know that python is a platform independent language and it can be used in any of our operating systems. As we are using python, if python is installed in the machine this system will work on that. We have tested our system on windows, mac and ubuntu OS and it works completely fine but so far the latest version of gaze pointer is in windows variant only. Both the systems are screen size independent, which means whether it is a 1k or 2k display, the system will not be affected by the resolution.
Moreover, Gaze Pointer can work over eye glass, where our system is less accurate in this part. To improve this accuracy we need to use a better web cam, as the web cam we are using currently is not focused enough.The reason behind the limitation of our system is. the camera is low budget and it cannot overcome the glass reflections over eyes. So, the betterment of the camera with improved camera focus, our system will be able to overcome this limitation and provide a better result. Lastly, the calibration process is much simpler and short in our system. The user only need to spend few seconds to calibrate the system as he just need to determine the window of his iris by using the 4 image of calibration we mentioned in section 3.4.4.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The system mainly emphases on the development of human computer interaction (HCI). Trying to make a hands free system that controls cursor is not a recent invention as many researchers are working on this from decades. However, the latest progression and accessibility in technological sector helps us with an advantage to introduce the system as a cost effective system. This cursor control system will add a new dimension of accessibility for the disabled people. They do not have to stay in bed whole day without doing anything. So far, we have tried to provide them a solution that is able to cooperate with computer. Eye pointer evaluation is a process in computer navigation system where the navigation system of the computer becomes a real time process. The main intention and focus of the research is building a system which will be hands free and cost effective. However, we have faced many challenges in terms of designing the system and we have solved problems such as: improving the detection of eyes from region of interest (ROI) after framing, enhancing the accuracy rate of cursor movement as well as detecting the condition of both eyes as they are on or off. Moreover, when we were designing the system our goal was to prepare a system which is fit in every condition and also for every user. One of the important obstacles that have been removed which is, the system works perfectly in low light and in low intensity there is no problem in terms of the detection of iris area. The mouse pointer is operated along with the eyes. The most noteworthy thing of the system is user does not need to wear any kind of heavy wearable instruments. This feature has made the system more enjoyable and user friendly. One thing should be added which is mouse clicking events. Mouse clicking events has been implemented by eyes blink. In contrast with other available systems, the proposed system here can work perfectly without using the facial landmarks. When the light was not that much bright, the accuracy rate was fluctuating in terms of detecting the closed eyes. The challenge can be solved by using more advance camera where the focusing is more accurate. Finally, it can be stated that the system can improve human computer interaction in future by showing its accuracy in different environment with different ages of users.

5.2 Future Work

Our system is already able to serve all computer functionality. In future, we have a plan to use a better camera which will improve the system stability. Furthermore, with an integration of better camera this system can serve iris based security system too. Gesture based unlock system can be implemented using this system. There is also a plan to integrate machine learning along with this system which will give user suggestions about the randomly used applications. For example, if a user keeps checking mail in the morning, at the start up the system will give a small pop-up in the window so that with a blink he can open his required application. We believe that, this system will help people in the long run and serve their all desired purposes.

Bibliography

[1] S. Naveed, B. Sikander, and M. S. H. Khiyal, "Eye tracking system with blink detection", *Journal of Computing*, vol. 4, no. 3, pp. 51–60, 2012.

[2] C. D. R. Foundation, "Paralysis statistics.", *Stats about paralysis, Available Online at https://www.christopherreeve.org/living-with-paralysis/stats-about-paralysis*, 2019.

[3] A. Poole and L. J. Ball, "Eye tracking in hci and usability research", in *Encyclopedia of human computer interaction*, IGI Global, 2006, pp. 211–219.

[4] J. Wahlström, "Ergonomics, musculoskeletal disorders and computer work", *Occupational Medicine*, vol. 55, no. 3, pp. 168–176, 2005.

[5] A. Cooper and L. Straker, "Mouse versus keyboard use: A comparison of shoulder muscle load", *International Journal of Industrial Ergonomics*, vol. 22, no. 4-5, pp. 351–357, 1998.

[6] M. Kowalik, *How to build low cost eye tracking glasses for head mounted system*, 2010.

[7] Y.-S. Yeung, "Mouse cursor control with head and eye movements: A low-cost approach", *Master's Thesis, University of Applied Sciences Technikum Wien, Available Online at http://www. asterics. eu/fileadmin/user—upload/Thesis—Yat-sing% 20Yeung—final. pdf*, 2012.

[8] R. Ramesh and M. Rishikesh, "Eye ball movement to control computer screen", *Journal of Biosensors & Bioelectronics*, vol. 6, no. 3, p. 1, 2015.

[9] S. S. Wankhede, S. Chhabria, and R. Dharaskar, "Controlling mouse cursor using eye movement", *International Journal of Application or Innovation in Engineering & Management*, vol. 36, pp. 1–7, 2013.

[10] T. PAUL, U. A. Shammi, M. U. Ahmed, R. Rahman, S. Kobashi, and M. A. R. Ahad, "A study on face detection using viola-jones algorithm in various backgrounds, angles and distances", *International Journal of Biomedical Soft Computing and Human Sciences: the official journal of the Biomedical Fuzzy Systems Association*, vol. 23, no. 1, pp. 27–36, 2018.

[11] G. Papari and N. Petkov, "Edge and line oriented contour detection: State of the art", *Image and Vision Computing*, vol. 29, no. 2-3, pp. 79–103, 2011.

[12] S. Priyanka and N. Kumar, "Noise removal in remote sensing image using kalman filter algorithm", *Int. J. Adv. Res. Comput. Commun. Eng*, vol. 5, pp. 894–897, 2016.

[13] A. Saha, “Read, write and display a video using opencv (c++/ python)”, *Read, Write and Display a video using OpenCV (C++/ Python), Available Online at https://www.learnopencv.com/read-write-and-display-a-video-using-opencv-cpp-python*, 2019.

[14] CVisionDemy, “Extract roi from image with python and opencv”, *Extracting a ROI (Region of Interest) using OpenCV and Python, Available Online at https://cvisiondemy.com/extract-roi-from-image-with-python-and-opencv*,

[15] A. K. Alexander Mordvintsev, “Changing colorspaces”, *Opencv-python-tutroals.readthedocs.io, Available Online at https://opencv-python-tutroals.readthedocs.io/en/latest/pytutorials/pyimgproc/pycolorspaces/pycolorspaces.html*, 2013.

[16] M. S. Manjare and M. S. Chougule, “Skin detection for face recognition based on hsv color space”, *International Journal of Engineering Sciences & Research Technology*, vol. 2, no. 7, pp. 1883–1887, 2013.

[17] P. M. B. Ong and E. R. Punzalan, “Comparative analysis of rgb and hsv color models in extracting color features of green dye solutions”, in *DLSU Research Congress*, 2014, pp. 1500–20.

[18] I. S. P. James, “Face image retrieval with hsv color space using clustering techniques”, *The SIJ Transactions on Computer Science Engineering & its Applications (CSEA)*, vol. 1, no. 1, 2013.

[19] X. S. Ltd, “Image masking”, *Masking images, Available Online at http://www.xinapse.com/Manual/masking.html*, 2015.

[20] T. point, “Concpent of masking”, *Masking techniques, Available Online at http://www.xinapse.com/Manual/masking.html*, 2015.

[21] E. S. Gedraite and M. Hadad, “Investigation on the effect of a gaussian blur in image filtering and segmentation”, in *Proceedings ELMAR-2011*, IEEE, 2011, pp. 393–396.

[22] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, “Hypermedia image processing reference”, *England: John Wiley & Sons Ltd*, 1996.

[23] S. C. Foundation, “Blurring images”, *Blurring images, Available Online at https://mmeysenburg.github.io/image-processing/06-blurring/*, 2016-2018.

[24] A. K. Alexander Mordvintsev, “Smoothing images”, *OpenCV-Python Tutorials 1 documentation. (2019), Available Online at https://opencv-python-tutroals .readthedocs.io/en/latest/pytutorials/pyimgproc/pyfiltering/pyfilter ing.html*,

[25] P. S. Patil, “Iris recognition based on gaussian-hermite moments”, *International Journal on Computer Science and Engineering*, vol. 4, no. 11, p. 1794, 2012.

[26] O. D. Team, “Hough circle transform”, *Hough Circle Transform, Available Online at https://docs.opencv.org/2.4.13.7/doc/tutorials*, 2018.

[27] P. Joshi, “Window to viewport transformation.”, *Window to viewport transformation, Available Online at https://www.ques10.com/p/11199/define-window-and-viewport-derive-window-to-view-1*, 2016.

[28] S. Pradhan, “Mid-point line generation algorithm”, *Mid-Point line drawing algorithm, Available Online at https://www.geeksforgeeks.org/mid-point-line-generation-algorithm*, 2018.

[29] A. Rasmussen and D.-A. Jirenhed, “Learning and timing of voluntary blink responses match eyeblink conditioning”, *Scientific Reports*, vol. 7, no. 1, p. 3404, 2017, ISSN: 2045-2322. DOI: 10.1038/s41598-017-03343-2. [Online]. Available: https://doi.org/10.1038/s41598-017-03343-2.