Iris Communicator

This thesis was submitted in partial fulfillment of the requirement for the Degree of
Bachelor of Computer Science and Engineering

Under the Supervision of

Dr. Md. Ashraful Alam

By

Shahriar Ali (14101242)
&
Nafis Hasrat Arnob (14101202)

School of Engineering and Computer Science
BRAC University, Dhaka, Bangladesh

Declaration

We hereby assure that all the work done for this thesis project is compiled with the results obtained from our work. All the algorithms, tools and materials that helped us to develop this project have been properly acknowledged and referenced. This thesis either in whole or in part, has not been previously submitted for any degree anywhere.

Signature of Supervisor

Dr. Md. Ashraful Alam

Signature of the Authors

Shahriar Ali (14101242)

Nafis Hasrat Arnob (14101202)

Acknowledgement

We would like to show our immense and sincerest of gratitude to our respected Supervisor Dr. Md. Ashraful Alam for his support and guidance in this project and the writing of this paper. His constant vigilance on ensuring we always made steady progress and care to always point us in the right path ensured the completion of this Thesis. We are absolutely humbled to have had him as our Supervisor.

We'd also like to thank Almighty for being fortunate enough to remain on track of progress for an entire year both physically and financially without having to face any serious illness or emergencies.

We want to thank our Parents for their constant support as well as our friends who helped in aiding with data collection which was crucial for our research. It wouldn't have been possible to maintain a steady workflow without their help.

Finally, this goes without saying, we want to say a special thank you to BRAC University for enabling us to conduct this research and providing us with the right support and tools to d

Table of Contents

Table of Figures

List of Tables

List of Abbreviations

ALS	Amyotrophic Lateral Sclerosis
Pyttsx	Python text to speech
gTTS	Google text to speech
VR	Virtual Reality
AR	Augmented Reality
UI	User Interface
GUI	Graphical User Interface
UX	User Experience
BGR	Blue – Green – Red
RGB	Red – Blue – Green
OpenCv	Open source computer Vision
EoG	Electro-oculogram

Abstract

This research purposes an eye pupil tracking experimental software to enable communication abilities for individuals with severe motor impairment. Since the ability of moving eye pupil is often preserved, even in severe conditions such as locked-in syndrome, Iris Communicator allows text entry with eye movement as the only input modality. To begin with, a camera was used to get video input from where we got BGR (Blue-Green-Red) images. Afterwards, we converted those BGR images into grayscale images. Then, we used Adaptive Threshold method for smoothing the images. The images were finally ready for eye pupil detection. Here, we used Hough Circle Transform method for detecting eye pupil as a circular object. Then, we programmed an algorithm to plot and keep track of the eye pupil movement and we designed a new keyboard layout to synchronize. In addition, we programmed a calibration method to extend the gaze area of eye pupil. Finally, we included text to speech feature by using pyTTSx3 and created a combination chart for each English letter. In future, we will be working on creating a user interface using Tkinter. Also, we will add more features like taking blink input and adding a cursor mode.

Chapter 01

Introduction

1.1 Introduction

In most cases, the sense is sight is the main source of data for knowing the surrounding environment. Therefore, it is natural to assume that information about where a gaze is focused can be helpful in finding out how our brain make communications with the environment. In the area of Human-Computer Interaction (HCI) that knowledge is critical for creating an intuitive and ergonomic user interface. However, the fundamental step in implementing such an interface is the exact location of user eye-pupil.

The aim of this paper is to describe selected algorithms for eye pupil detection, compare their effectiveness using static digital images, and implement in an application for eye-controlled computer operation. The effectiveness assessment was based on the collection of facial images with the actual pupil's locations attached. The implementation was performed using an ordinary webcam, with a standard resolution of 640 x 480 or even 320x240 pixels.

In the next section, the general algorithm for finding eyes on a digital image is presented. The following part concentrates on three commonly used methods for determining eye pupil position. It also describes all the important implementation details, e.g. threshold values providing the best possible results. Section 4 provides the comparison of the algorithms performed on static images as well as on images acquired in a real-time mode. Short summary presents the conclusions.

1.2 Motivation

The knowledge we gathered from our university is the tool that we tried to use for the welfare of our society. There are extremely talented people who suffer from paralysis. Iris communicator People who suffer from Lou Gehrig's Disease. Moreover, Technology has played the most vital role in advancement of world health and medical science. As a developing country, Bangladesh shows a promise in this arena. We tried to implement the feature of image processing for the advancement of medical science. We want to contribute to the scientific development not only our country, but also the whole world.

1.3 Problem Statement

It is very effortful to get only the eye pupil tracking as a circular object Hough circle method. For example, we used a regular camera sensor which had problems as the blurry images could not give correct result. In addition, the eye pupil detection was impossible in low light situation. Moreover, when it detected a circle sometimes the method detected something other than eye pupil as a circular object. In the meantime, we had to modify methods and we used different libraries. Also, we faced problem merging python 2.7 codes in python 3.7. Again, the co-ordinates value varies from person to person.

1.4 Solution

Firstly, we bought a Logitech webcam C310 to get clear images. Secondly, we used extra lights to solve the low light situation. Thirdly, we took an average value of the radius and limit its value while programming. So that, the program only detects objects same size as an eye pupil. Fourthly, we selected python's OpenCV library and python 3.7 (latest) to optimize our program. Finally, we created a calibration method to take new values for x axis and y axis which eliminated variable co-ordinates value problem.

1.5 Contribution Summary

The researched Iris Communicator software can be used in laptop with built in webcam. The software is unique in every way, from its value to its layout. It can contribute in many ways, such as when someone is badly injured and can only move his/her eye pupil, when someone's body is burnt to extreme level and of course someone suffering from ALS, he or she can convey important message. Now, we know that there are other technologies available for special persons like Steven Hawking. However, those technologies are very expensive and modified generally for a single person. For this reason, we have come up with something that may contribute to the hospital and patients for free of cost.

1.6 Methodology

In the literature review, previous works regarding eye tracking are mentioned. We are using a very different approach to the conventional eye tracking and designed a completely different yet easy to understand keyboard layout. The whole program is complete new and unique. It has never been visualized or done in such a way we have done Also, we have created just a software which will work with any laptop webcam to make its production cost zero. We can recommend certain types of cameras which will produce clear images which can be used with desktop computers for those patients in wheel chair. Also, our calibration system will give the user the scope of changing the position of the camera for his/her convenience. Also, we have added a text-to-speech functionality so that the user may hear what he/she has written. Our program ran without any error and delivered the word 'Hello' as well as reading it out loud at the very end. So, we are building a User Interface using python Tkinter and we will give the scope of editing delay time for choosing a word in future. Our software is an experimental software with work in progress to make it an end user software.

1.7 Thesis Outline

- Chapter 2 elaborates the background study behind the motivation of this project
- Chapter 3 covers methodology
- Chapter 4 presents the experimental run and the comparison
- Chapter 5 concludes mentioning the promising future of this research

Chapter 02

Literature Review and Related Research

2.1 Literature Review

The history of eye tracking reaches back to the late 19th century. At the beginning, mechanical devices were used to detect light reflected by a plate implanted directly into the cornea. Development of photography and video recording allowed for much more reliable and less invasive methods of eye movement observation over long periods of time. Such studies became more popular especially in psychology and medical research as well as in diagnostics. However, it has been only recently that the computing power become high enough to allow for the development of a computer interface based on a real time eye tracking analysis.

The history of eye tracking reaches back to the late 19th Century. At the beginning, mechanical devices were used to detect light reflected by a plate implanted directly into the cornea. Development of photography and video recording allowed for much more reliable and less invasive methods of eye movement observation over long periods of time. Such studies became more recognized especially in psychology and medical research as well as in diagnostics. However, it has been only recently that the computing power become high enough to allow for the development of a computer interface based on a real-time eye tracking analysis.

Currently, the eye tracking techniques develop in two directions, electrooculography (EOG) and digital image analysis. The last one, which is the research area of this work, uses cameras operating in the visible light spectrum and software analyzing digital images. The increase in computing power also gave way to the number of techniques carrying out such analysis. The advantage of methods using visible light is their versatility. They are independent of such individual characteristics of an eye such as current flow in the cornea.

Unfortunately, the commercially available applications require specialized equipment (e.g. sensitive low noise video camera allowing fast transfer of high resolution frames), which makes them quite expensive. There is also alternative approach using open--- source software based on eye pupil reflection in infrared light, but the hardware needed limits its features. The only cost-effective solution using visible light is the Eye Track; however, it does not allow for the precise comparison of different algorithms.

2.2 Related Works

Tobii Eye Tracker

Tobii eye tracker is quintessential to eye tracking technology. This software and hardware combination made by Tobii company is industry leading. Also, they have partnered with giants like Microsoft where Microsoft recommends using Tobii eye tracking devices for disable people in their official website.

QVirtboard

Qt Virtual Keyboard is a simple on-screen keyboard application, which extends its functionality in order to allow disabled people to control computer environment. This program can be used as a standard point-and-click virtual keyboard, as well as a more complex tool which can cooperate with web cameras or more sophisticated devices.

EoG based assistive communication

An electro-oculogram based assistive communication system with improved speed and accuracy using multi-directional eye movements. In this system, horizontal and vertical electro-oculogram were measured using two electrodes attached on top and on side of the eye and referring to an earlobe electrode and amplified with AC-coupling in order to reduce the unnecessary movement. [27]

BlinkWrite 2

BlinkWrite2 was developed as a utility for scanning specific keyboard. It got the past features of BlinkWrite such as duration-based blink selection, error correction and auditory feedback. This software enables disabled people to communicate using blink as the only input system. [28]

Chapter 03

Methodology

3.1 Procedure

We started off using OpenCv by taking inputs from webcam [2]. We got video input in BGR. We converted it into grayscale. We used adaptiveThreshold method and ADAPTIVE_THRESH_GAUSSIAN method for making the image smoother. After that, we used Hough transform's Circle method get circular object from a limited number frames of a video. As we set the camera in front of the eye at a certain range in which we get the eye pupil as only circular object (theoretically) we get the X-axis, Y-axis and radius (eye pupil) value from it.

The initial theoretical plan was about to be implemented, the first few steps were expectedly successful. Here, Haar Cascade did an excellent job recognizing face and eyes. Also, the Hough Circle Transform technique was stable enough to detect circle shaped eye pupils. Unfortunately, two various errors were found while using Hough circle transform technique. Firstly, Python's PyGaze library was selected for detecting the gaze. Here the problem is the library is made for Python 2.7 which is going to be outdated on 2020. As the Iris Communicator project focuses on future and long-lasting implementation with an open option for improvement with modification, the PyGaze library cannot be used. Secondly, taking the eye pupil movement input and plotting it into the graph showed error as the values received from ordinary webcam showed dynamic result. This dynamic result occurred because of dynamic light ambiance, lack of fixed auto focus option and the low resolution of the webcam. The research was done on multiple webcams all of which showed the same error.

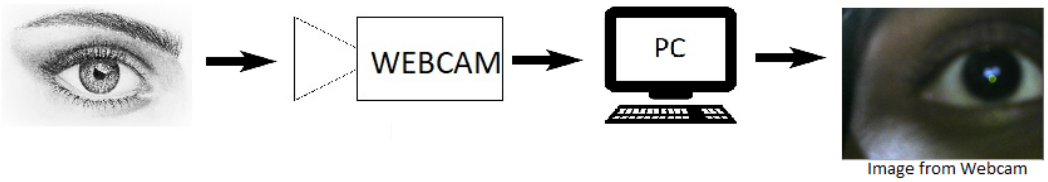

Figure 1 Getting Image via Webcam

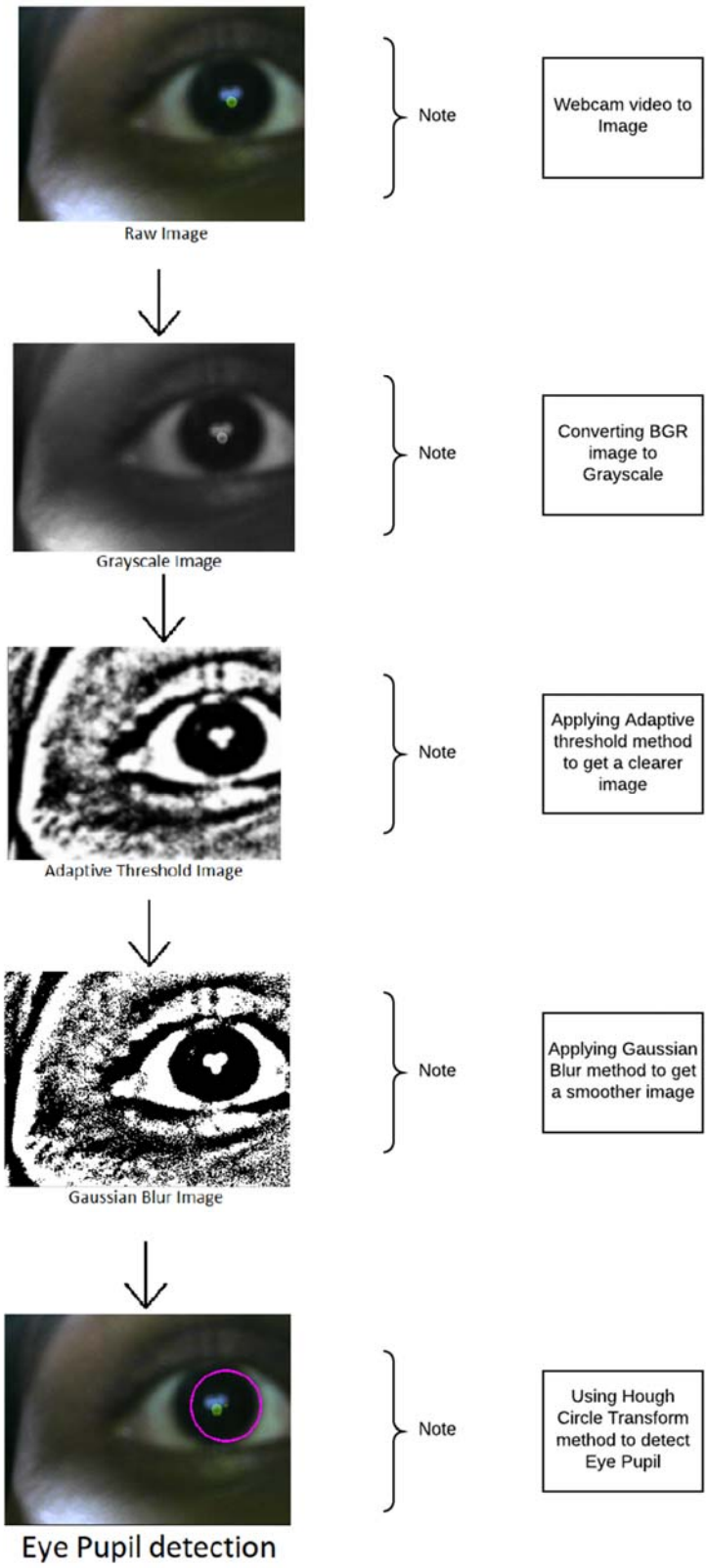

Figure 2 Image Processing

3.2 OpenCV

We used OpenCV library as it is free to use for academic purposes and commercial purposes. Also, it supports python 3.7 and an optimized library. Moreover, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform [2].

3.3 BGR to Grayscale conversion

There are a lot of benefits for using OpenCV library [13]. The equation converting BGR to grayscale:

RGB[A] to Gray: Y←0.299·R + 0.587·G + 0.114·B

Also, the following lines converts a BGR image to Grayscale:

```
“import cv2
image = cv2.imread('ironman.png')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imwrite('gray_image.png',gray_image)
cv2.imshow('color_image',image)
cv2.imshow('gray_image',gray_image)”
```

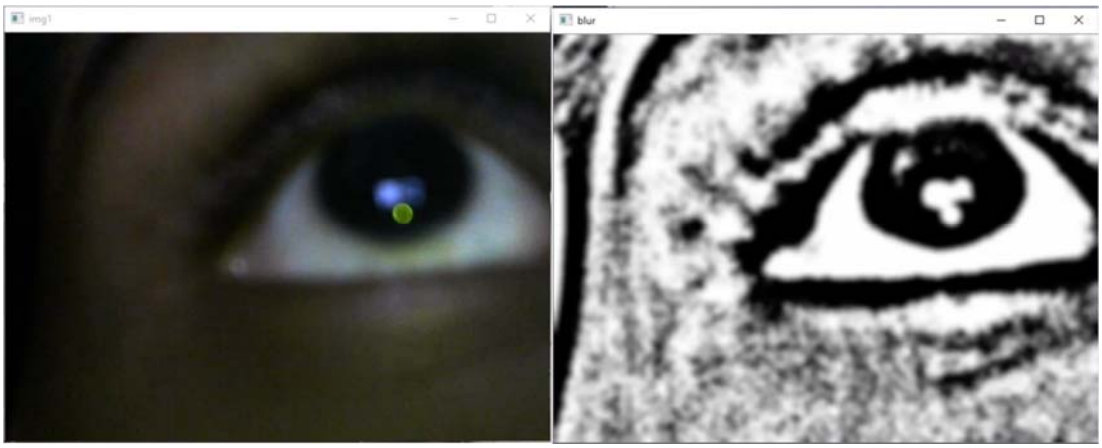

Figure 3 BGR (left) to Grayscale (right) conversion

3.3 Adaptive Threshold

Adaptive thresholding is the method where the threshold value is calculated for smaller regions and therefore, there will be different threshold values for different regions [15]. Also as we are using OpenCV library, we can perform Adaptive threshold operation on an image using the method adaptiveThreshold() of the built in Imgproc class. There are two methods for

adaptive threshold. We used ADAPTIVE_THRESH_GAUSSIAN method where threshold value is the weighted sum of neighborhood values where weights are a Gaussian window [16].

3.4 Hough Transform

Hough Transform was initially developed for detecting lines from images. However, Now, it has been updated to detect circular objects [3].

Parameterization

A circle can be fully defined with three parameters: the center coordinates (a, b) and the radius (R):

$$x = a + R\cos\theta \quad (1)$$

$$y = b + R\sin\theta \quad (2)$$

As the θ varies from 0to 360, a complete circle of radius R is created.

So with the Circle Hough Transform, we expect to find triplets of (x,y,R) from the image. In other words, our purpose is to find those three parameters [9].

3.5 Gaze Detection

The complex process of the whole eye pupil detection to conversion into a language starts with a very basic step. From, the triplet value we got from Hough Transform, we took 40 average values for error correction. We plotted the values in as co-ordinates in graph.

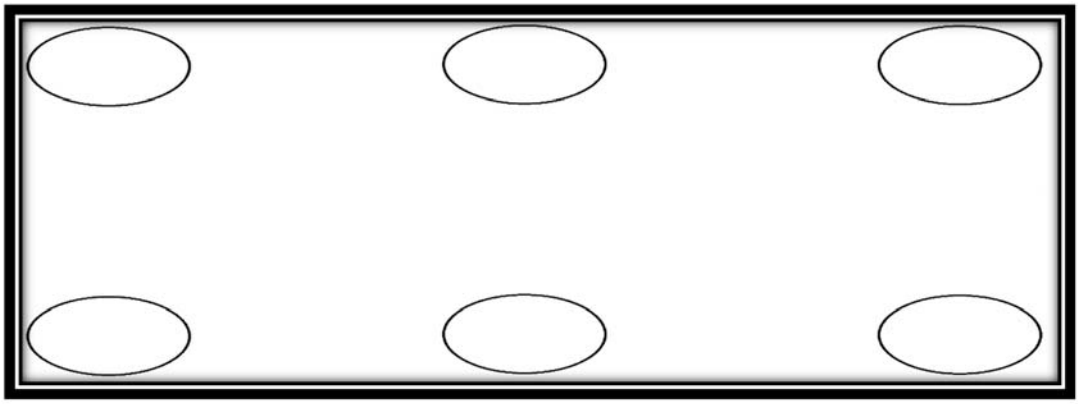

Figure 4 Initial Selected Gaze Areas

We named the areas as Top Left, Top Middle, Top Right, Bottom Left, Bottom Middle and Bottom Right.

3.6 Calibration

The most vital part of our whole research was creating the calibration part. The calibration part was done by plotting and finding revised mid-point. Firstly, we selected six areas of the screen. Figure 4 shows our initial selection.

However, we faced difficulties looking at a particular area. So, we tried plotting it in graph. Secondly, we got a range of values for x-axis, y-axis and radius value of circular object.

Figure 5 shows the average values we got generally looking at six pre-defined and selected areas.

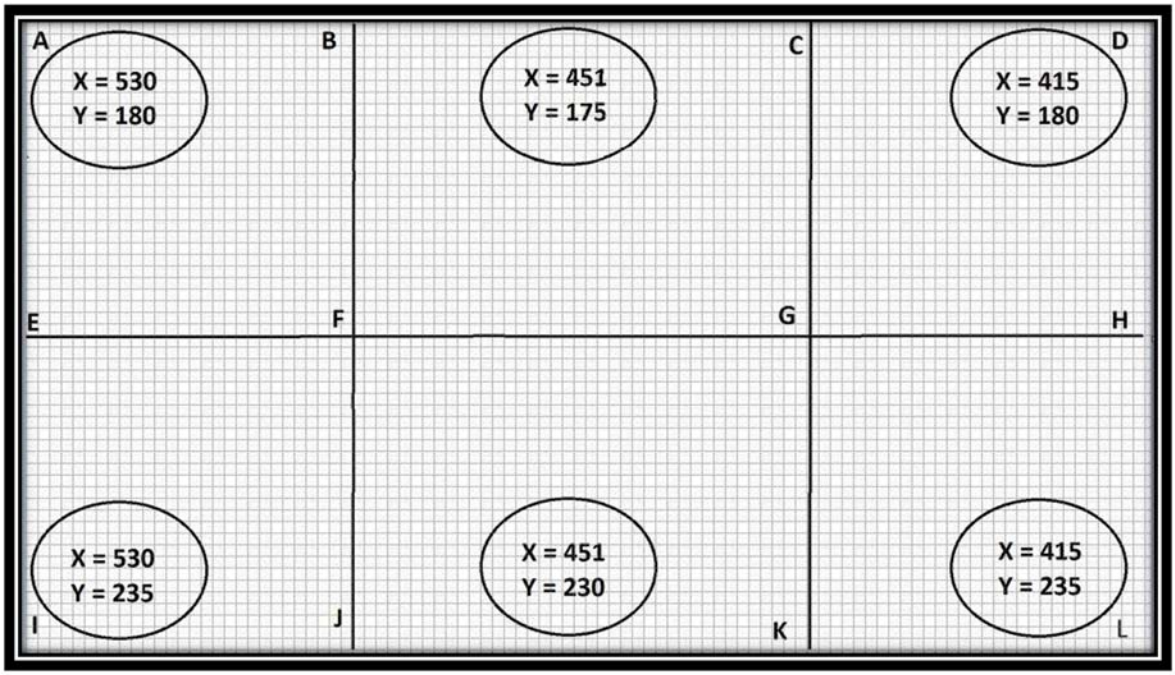

Figure 5 Extension of input area

Thirdly, we extended the circular region to a rectangular region. Now the following table shows the redefined area:

Table 1: Selected areas for extension

Area	Rectangle
ABEF	Top Left
BFCG	Top Middle
CDGH	Top Right
EFIJ	Bottom Left
FGJK	Bottom Middle
GHKL	Bottom Right

To calibrate and for making it more user convenient we took 40 values per frame of the video.

3.7 Eye movement tracking

We are taking 40 values of the triplets that we got from Hough transform and we are taking the frame which appeared most number of times. Thus, we got the co-ordinate of eye pupil. Then we plot the co-ordinate in to the 6 rectangles we found while calibration.

3.8 Keyboard Layout

After tracking eye movement, we generated letters based on the combination of two different co-ordinates as depicted in table 3. For this, we designed a keyboard layout. It is mentionable that our designed keyboard layout is really unique and never been done this way previously.

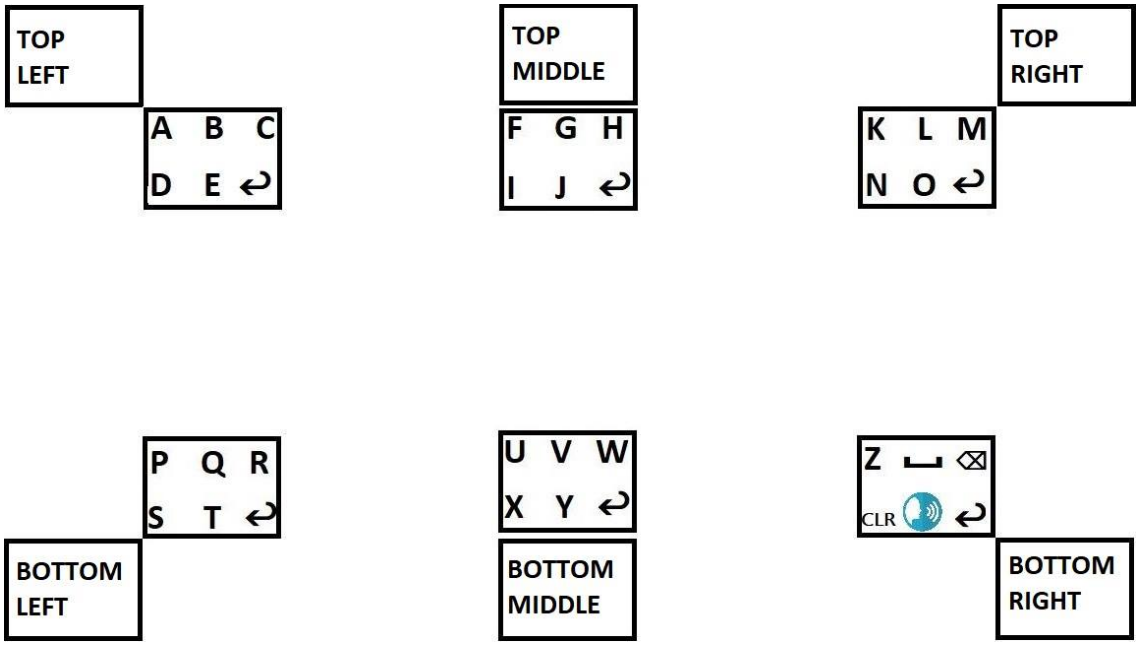

Figure 6 Keyboard Layout

We selected 6 areas of the screen and divided letters into 6 groups. The combination chart for each word is given below:

Table 2: Combination Chart

1st Combination	2nd Combination	Output
Top Left	Top Left	A
	Top Middle	B
	Top Right	C
	Bottom Left	D
	Bottom Middle	E
	Bottom Right	Back
Top Middle	Top Left	F
	Top Middle	G
	Top Right	H
	Bottom Left	I
	Bottom Middle	J
	Bottom Right	Back
Top Right	Top Left	K
	Top Middle	L
	Top Right	M
	Bottom Left	N
	Bottom Middle	O
	Bottom Right	Back
Bottom Left	Top Left	P
	Top Middle	Q
	Top Right	R
	Bottom Left	S
	Bottom Middle	T
	Bottom Right	Back
Bottom Middle	Top Left	U
	Top Middle	V
	Top Right	W
	Bottom Left	X
	Bottom Middle	Y
	Bottom Right	Back
Bottom Right	Top Left	Z
	Top Middle	Space
	Top Right	Delete
	Bottom Left	Clear All
	Bottom Middle	Text-To-Speech
	Bottom Right	Back

3.9 Text-To-Speech

There are two effective open source methods for Python's text-to-speech operation. One is gTTS and the other is PyTTSx3 [17]. We chose PyTTSx3 as it can run in cross platform – Python 2.7 and Python 3.x. Also, it can be run offline whereas gTTS downloads the audio data from internet and requires internet connection.

3.10 Drawback

The main problem with our keyboard layout is figuring out the co-ordinates which intersects in multiple regions.

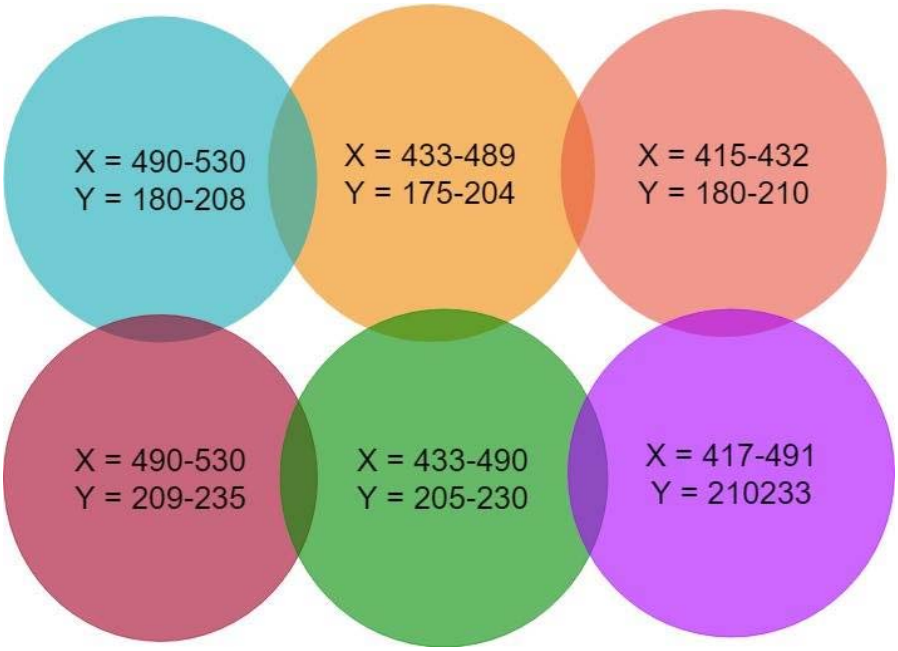

Figure 7 Multiple Region Intersection Problem

Here in figure 7, we can see that some areas are included in two regions. As a result, sometimes it is difficult to get the correct result.

Chapter 04

Experiment & Comparison

4.1 Hardware Setup

Earlier we used an old webcam which gave us blurry images. Facing this issue, we had to buy new webcam. We researched and bought Logitech C310 for this research.

Figure 8 Logitech C310 (Left) and Dissembled Part (Right)

We faced issue with C310's holder as it is really heavy. It was supposed to be on a holder attached to an eye glass. Therefore, we found it unusable with its holder attached. Later on, we decided to give it a usable shape. Firstly, we dissembled it and took out the main motherboard with all the components attached. Secondly, we bought an eye glass with a window large enough not to make any distraction in the way of the camera sensor. Thirdly, we soldered it with a steel like material attached to it. Also, we kept a certain distance from the camera so that it can get the proper image. Finally, we selected an angle in which eye pupil is easily detected and made the holder slightly movable. Also, the camera does not block one eye entirely and the user has the independence of moving his/her eyes elsewhere and see beyond the webcam. The glass setup with webcam

Figure 9 Camera module with holder

PC Configuration

Table 3 Demo system configuration

Item	Specification
Processor	Intel core i5 6500-CPU @ 3.2 GHz
Graphics Card	AMD Radeon rx570 4GB memory
RAM	8GB ddr4

4.2 Experiment

Before stating the demo result we would like to inform about the mirror effect of a camera. Due to mirror flipping effect, the mirror image we see is an exact inverted image in x-axis.

Here, we successfully generated the word ‘HELLO” and also read it out loud using a similar combination. The following figures contain captured images while generating ‘Hello’:

Generating ‘H’

Figure 10 Top Middle (left) + Top Right (Right)

Generating ‘E’

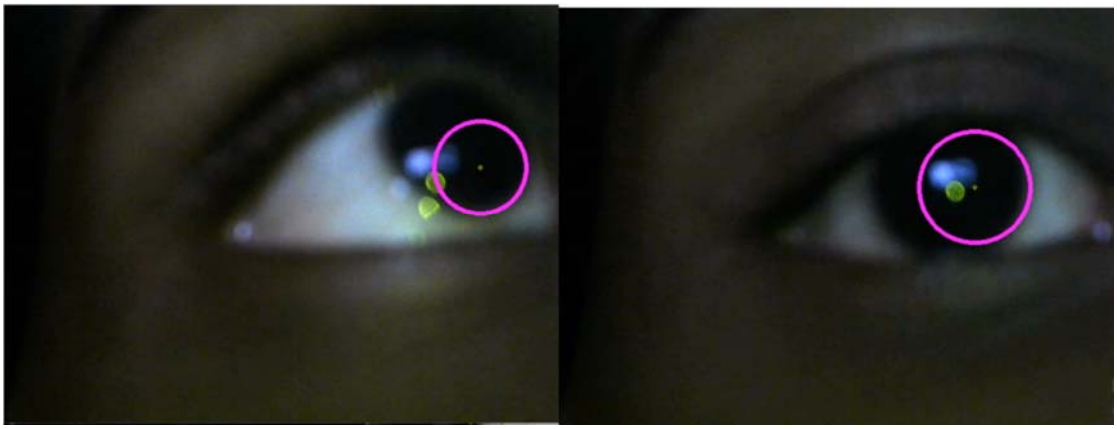

Figure 11 Top left (left) + Bottom Middle (right)

Generating ‘L’

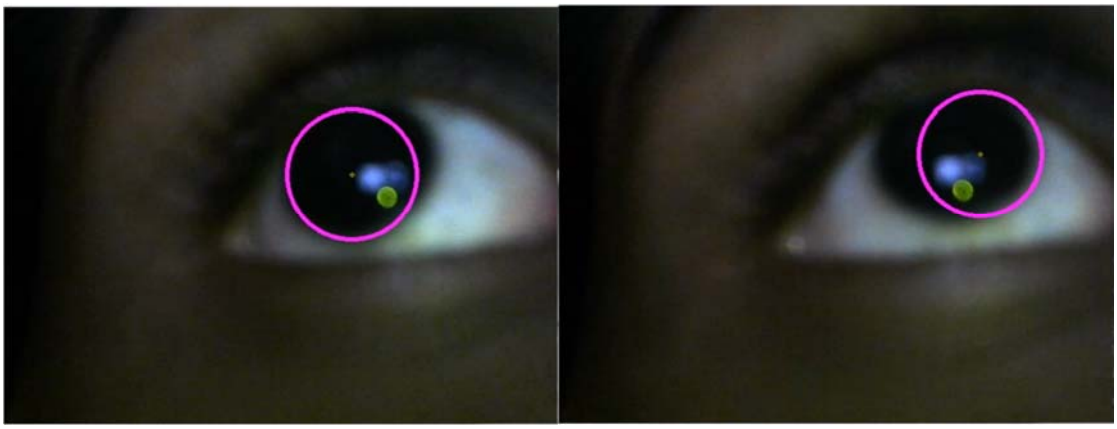

Figure 12 Top right (left) + Top middle (right)

Generating ‘L’

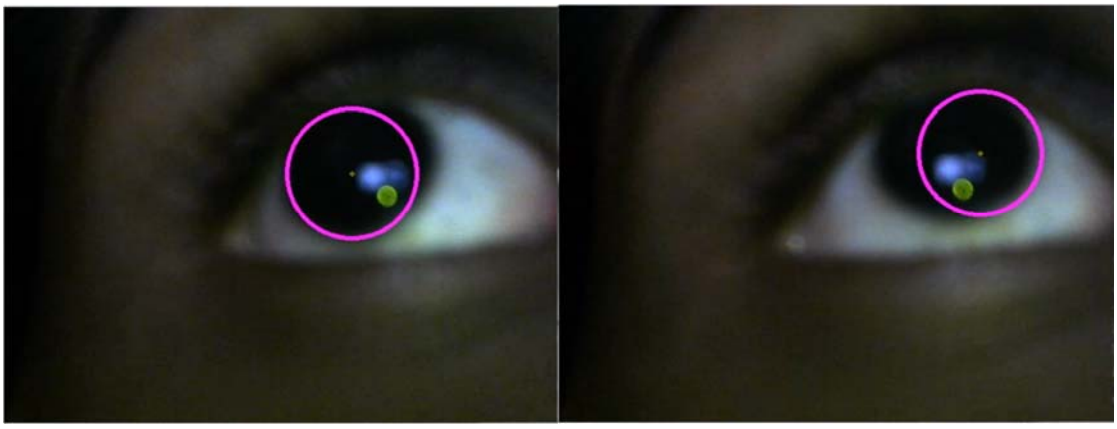

Figure 13 Top right (left) + Top middle (right)

Generating ‘O’

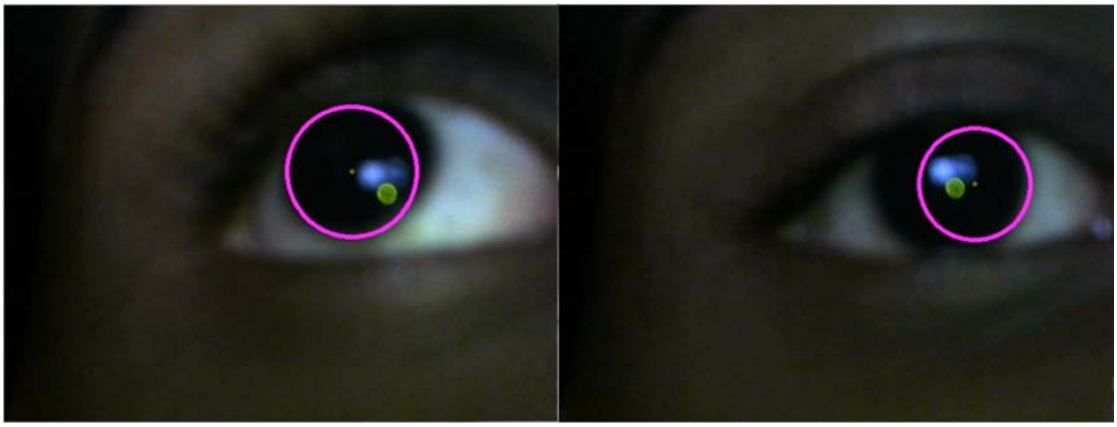

Figure 14 Top Right (left) + Bottom Middle (Right)

Triggering Text-to-Speech function

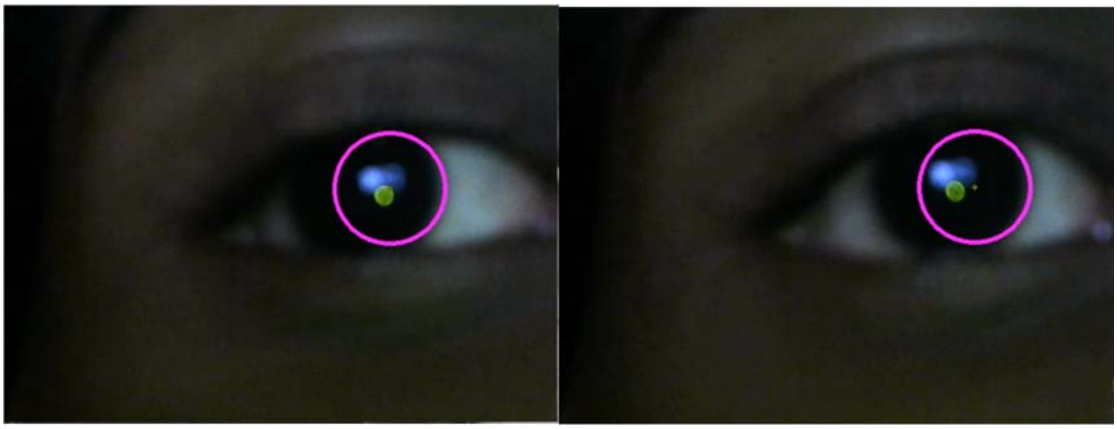

Figure 15 Bottom Right (left) + Bottom Middle (Right)

4.3 Result

Table 4 Experiment result

Session	X*	Y*	Radius*	Selected Area	Figure Number	Selected
1	470	160	60	Top Middle	10	H
2	410	170	62	Top Right		
3	500	168	63	Top Left	11	E
4	405	201	61	Bottom Middle		
5	470	162	60	Top Middle	12	L
6	410	170	65	Top Right		
7	470	163	60	Top Middle	13	L
8	405	168	64	Top Right		
9	415	171	63	Top Right	14	O
10	465	202	62	Bottom Middle		
11	410	210	63	Bottom Right	15	Text to Speech
12	460	201	65	Bottom Middle		

* Here the values are average of 40 frame values

The program was run directly from command prompt and the following is a screenshot of the prompt:

Figure 16 Command Prompt screenshot of the Demo

In short, we successfully generated ‘Hello’ in text form and also made the program read ‘hello’ out loud. However, we would add that our program does not always get the right result because of mixed co-ordinate issue described in Chapter 3 section 3.10.

4.4 Comparison

We have selected BlinkWrite2 as a competitor to Iris Communicator. BlinkWrite 2 is defined in Related Research section 2.2. We have selected to parameter for comparison. The first one is speed of typing. Figure 17 shows the speed comparison.

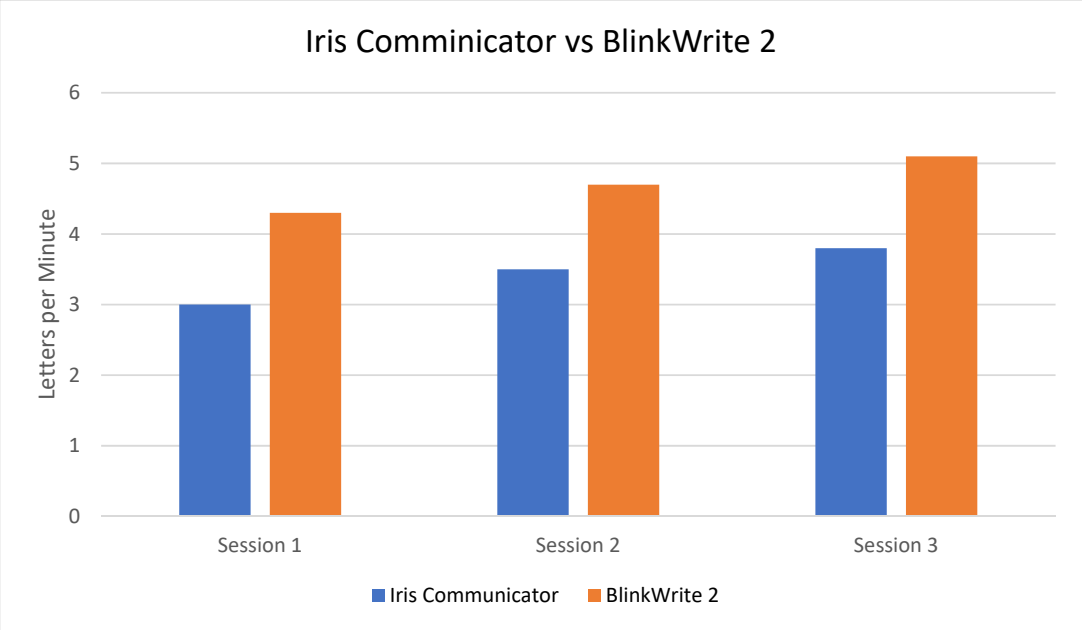

Figure 17 LPM Comparison with BlinkWrite 2 (Higher is Better)

Here, in speed, BlinkWrite is better in all 3 experiments.

The second parameter is accuracy. Figure 18 shows the letter deletion per minute comparison.

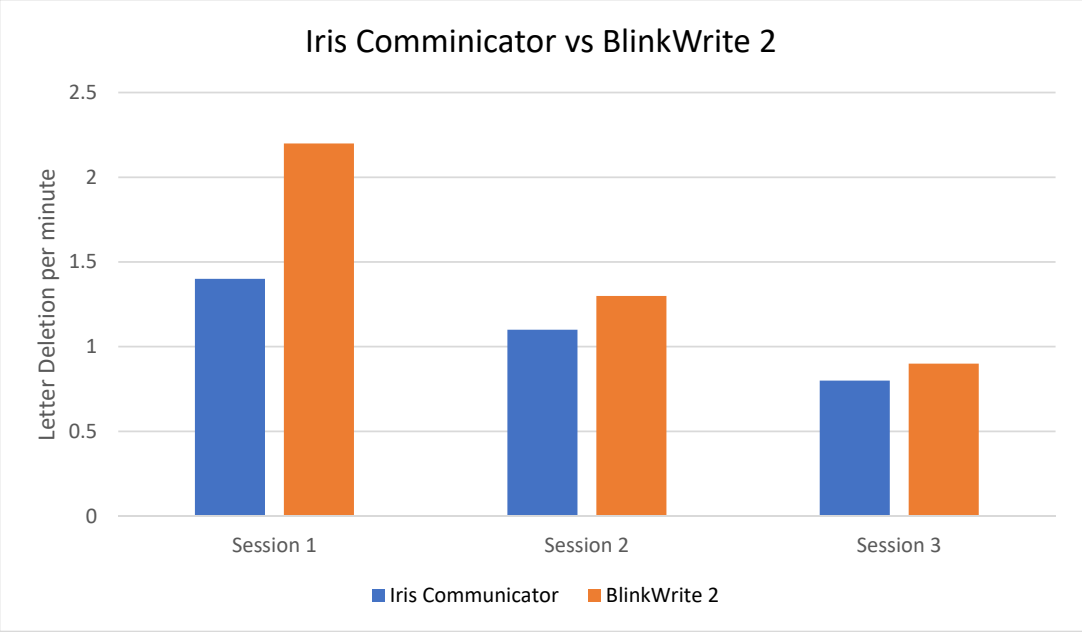

Figure 18 Accuracy comparison with BlinkWrite2 (Lower is Better)

Here, Iris Communicator is better in all 3 experiments in accuracy. We valued accuracy more that speed and we got the expected result. It is notable that our software is on experimental stage and BlinkWrite is on the revision 2. Therefore, with future work Iris Communicator will only get better.

The experiment was captured in a video and the video link is given below:
Video Link (Google Drive)

Chapter 05

Conclusion & Future Work

5.1 Conclusion

Application of eye tracking in human–computer interaction remains a very promising approach as its technological and market barriers are finally being reduced. As a direct control medium, the eye movements are obtained and used in real time as an input to the user–computer dialogue. The eye movements might be the sole input, typically for disabled users or hands-busy applications, or might be used as one of several inputs, combining with mouse, keyboard, sensors, or other devices. It is mentionable that while researching for Iris communicator we found out most research in eye tracking are made recently. Here, we tried popular methods like Haar Cascade and PyGaze but we failed. Also, we thought about buying an eye tracking device. However, our main motto was to build a free to use software. For hardware any working computer with webcam can handle Iris Communicator. The whole development is based on four steps on image processing and setting a letter for a co-ordinate. We are glad to make a new keyboard layout which also intrigued a newly formed combination chart. Also, we are satisfied with our experiment result as it delivered the expected output. The software is built targeting paralyzed people. People who are suffering from ALS will be benefited using it. Moreover, Iris communicator can be connected with controller and can be modified for people who can move their hand or finger. The possibilities of using eye tracking are endless. For example, a eye tracking can be used for security pattern using eye movement. However, there are limitation of such eye tracking prototype. Till now Samsung semiconductors have built an eye tracking system which will detect if the eyes are closed or not. Eventually, their attempt did not improve over the time due to the lack of further steps for improvement. This effort from Samsung opens Iris Communicators future for mobile devices. From the perspective of mainstream, eye-movement research, human–computer interaction, together with related work in the broader field of communications and media research, appears as a new and very promising area of applied work. Both basic and applied work can profit from integration within a unified field of eye movement research. We expect to add something important to eye tracking research and we are eager to discover more in this exciting new sector of research and modern technology.

5.2 Future work

User Interface

We are currently working on python Tkinter. We will make a user-friendly UI which will have interface for both patient and assistant. Also, we will be improving UX with Tkinter GUI. It will also feature a customizable delay time.

Cursor

The possibilities of this project are endless. During our work process we found out that the same eye pupil detection can be used to take an input like mouse. The process is very basic yet complex in real life because of maintaining a real time detection of eye pupil to reach up to the user expected mouse level accuracy. However, the uses of this process are specific and prospective. For example, when lying and keeping the laptop on the lap while watching a video the user may use Iris Communicator to change media control.

Blink Input

The eye tracking process includes the blinks of a person as well as the number of blinks. In addition, the duration of a blink is also included. For example, one blink for less than a second is a ‘Space’ and one long blink is for stopping the application from tracking the eyes so that the user may take rest for a while.

Improved Peripherals

An improved camera with IR sensor will definitely improve the detection. Also, integration with Tobii eye tracker can really upgrade eye detection and typing speed. Moreover, a better wearable design will make Iris Communicator more convenient.

References

[1] Hammoud, R. (2008). Passive Eye Monitoring. Berlin, Heidelberg: Springer-Verlag.

[2] Horsley, M. (2016). Current Trends in Eye Tracking Research. [S.l.]: Springer International Pu.

[3] Illingworth, J. and Kittler, J. (1987). The Adaptive Hough Transform. IEEETransactions on Pattern Analysis and Machine Intelligence, PAMI-9(5), pp.690-698.

[4] Nakano, Y., Conati, C. and Bader, T. (2013). Eye gaze in intelligent user interfaces. London: Springer.

[5] Bogotobogo.com. (2018). OpenCV 3 Hough transform: Circle - 2018. [online] Available at: https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Hough%20Circle_Transform.php

[6] Bojko, A. (2013). Eye tracking the user experience. Brooklyn, New York: Rosenfeld Media.

[7] Docs.opencv.org. (2018). OpenCV: Hough Circle Transform. [online] Available at: https://docs.opencv.org/3.1.0/da/d53/tutorial_py_houghcircles.html

[8] Duchowski, A. (2017). Eye Tracking Methodology. Cham: Springer.

[9] Numpy.org. (2018). NumPy ó NumPy. [online] Available at: http://www.numpy.org/

[10] Opencv.org. (2018). OpenCV library. [online] Available at: https://opencv.org/

[11] Stephen Hawking. (2018). The Computer. [online] Available at: http://www.hawking.org.uk/the-computer.html

[12] Dalmaijer, E., MathÙt, S. and Van der Stigchel, S. (2013). PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. Behavior Research Methods, 46(4), pp.913-921.

[13] Docs.opencv.org. (2018). OpenCV: Color conversions. [online] Available at: https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html

[14] Docs.opencv.org. (2018). OpenCV: Image Thresholding. [online] Available at: https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html

[15] Docs.opencv.org. (2018). OpenCV: Smoothing Images. [online] Available at: https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html

[16] Docs.python.org. (2018). 3.7.1 Documentation. [online] Available at: https://docs.python.org/3/

[17] En.wikipedia.org. (2018). Eye tracking. [online] Available at: https://en.wikipedia.org/wiki/Eye_tracking

[18] En.wikipedia.org. (2018). Hough transform. [online] Available at: https://en.wikipedia.org/wiki/Hough_transform

[19] iMotions. (2018). What is Eye Tracking and How Does It Work? - iMotions. [online] Available at: https://imotions.com/blog/eye-tracking-work/

[20] Lin, X. and Otobe, K. (2001). Hough transform algorithm for real-time pattern recognition using an artificial retina camera. Optics Express, 8(9), p.503.

[21] Lin, Y., Lin, R., Lin, Y. and Lee, G. (2012). Real-time eye-gaze estimation using a low-resolution webcam. Multimedia Tools and Applications, 65(3), pp.543-568.

[22] Opencv-python-tutroals.readthedocs.io. (2018). OpenCV-Python Tutorials ó OpenCV-Python Tutorials 1 documentation. [online] Available at: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

[23] Tobii.com. (2018). Tobii Tech - What is eye tracking? [online] Available at: https://www.tobii.com/tech/technology/what-is-eye-tracking/

[24] Wang, B. and Zhang, L. (2014). Fast and Effective Algorithm of Iris Localization Based on Hough Transform. Applied Mechanics and Materials, 519-520, pp.786-791.

[25] YouTube. (2018). Intro and loading Images - OpenCV with Python for Image and Video Analysis 1. [online] Available at: https://www.youtube.com/watch?v=Z78zbnLlPUA&list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq

[26] Yuen, H., Princen, J., Illingworth, J. and Kittler, J. (1990). Comparative study of Hough Transform methods for circle finding. Image and Vision Computing, 8(1), pp.71-77.

[27] Divya, S., Vinod A.P., and Kavitha P. Thomas, (2012), Published in the Proceedings of 35th International Conference on Telecommunications and Signal Processing (TSP), Prague, Czech Republic, July 2012.

[28] Behrooz, A. and Scott M. (2010). BlickWrite2: An Improved Text Entry Method using Eye Blinks.