“Eye Writer – An Interaction Device for Disabled Person”

Submitted By

Vishwa Gandhi (Enrollment No. 120110107019)

Jigar Mangukiya (Enrollment No. 120110107027)

Faculty Guided

Dr. Maulika S. Patel (PhD)

[Head - Computer Engineering Department]

A Project Report Submitted to Gujarat Technological University

In Partial Fulfillment of Requirements for the Degree of Bachelor of Engineering

In **Computer Engineering**

May 2016

Academic Year 2015 – 2016

Department of Computer Engineering

G. H. Patel College of Engineering & Technology

Bakrol Road, VallabhVidhyanagar-388120 (Gujarat)

Phone: (02692) 231651

CERTIFICATE

This is to certify that the project entitled "**Eye Writer – An Interaction device for disabled person**" has been carried out by **Vishwa Gandhi (120110107019) & Jigar Mangukiya (120110107027)** at **G H Patel College of Engineering & Technology** for partial fulfillment of B.E degree to be awarded by Gujarat Technological University. This project work has been carried out under my supervision and is to the satisfaction of department.

Date:
Place:

Project Guide	**Head of the Department**
Dr. Maulika S. Patel	**Dr. Maulika S. Patel**

Principal

Dr. Himanshu Soni

Seal of Institute

Undertaking of Originality of Work

We hereby certify that we are the sole author of this report and that neither any part of this work nor the whole of the work has been submitted for a degree to any other University or Institution.

We certify that, to the best of our knowledge, our work does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in our report, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that we have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Indian Copyright Act, we certify that we have obtained a written permission from the copyright owner to include such material in our work and have included copies of such copyright clearances to our appendix.

We declare that this is a true copy of our report, including any final revisions, as approved by our supervisor.

Date:
Place:

-Vishwa Gandhi(120110107019) & Jigar Mangukiya (120110107027)

Acknowledgement

We would like to express our gratitude to our project guide **Dr. Maulika S. Patel** for her supervision, encouragement, suggestions and guidance throughout the development of this project. For all the other faculties of department of Computer engineering for their constant support and motivation without which we would not have been able to complete our work.

Without the support of our colleagues this project wouldn't have been feasible for us. We would also like to thank our families for their love and morale support that made everything easier for us.

And in the end, we would like to thank the Almighty for making us worthy enough to accomplish our goal.

.

- **Vishwa Gandhi & Jigar Mangukiya**

Table of content

List of Figures

Page No.

List of Tables

Page No.

“Eye Writer – An Interaction device for disabled persons”

Submitted By

Vishwa Gandhi **(Enrollment No. 120110107019)**

Jigar Mangukiya **(Enrollment No. 120110107027)**

Supervised By

Dr. Maulika S. Patel (PhD)

[Head - Computer Engineering Department]

Abstract

“Eye Writer” is a wearable device, which helps a paralyzed or disabled person interact with other persons using their eye movements. A person having Neuromuscular paralysis cannot move his/her limbs and hence cannot talk or make gestures. Which makes a person with such symptoms, unable to express himself to doctors or family members.

Eye Writer tracks their eye movement from a camera mounted on a spectacle frame and by that it allows user to manipulate cursor on a digital screen. This way a person having neuromuscular paralysis can communicate to some extent.

Fully functional prototype is implemented using MATLAB. First attempt was made by using “.NET framework” along “OpenCV” library which failed to give a real-time feel to device.

GUI has all day-to-day and basic requirements that a user may need. This requirements are categorized and are divided into menus and sub-menus, which in turn makes Eye Writer more user friendly.

Eye Writer has a calibration mechanism which helps it to be used universally. Calibration is required because each human eye has different size and color. Certain surveys are conducted on different people to test the universality of the Eye Writer using manual calibration.

1. INTRODUCTION

1.1 Summary

There are millions of people worldwide who suffer from a health situation called 'Neuromuscular paralysis". This people cannot move their body parts because of muscular non-functionality, and as a result they are unable to talk with people around them. May they be doctors who are asking the person how is he feeling? or a family member, who is asking what he wants? The patient can hear and understand but cannot reply, and hence they feel severe frustration in some cases because of their inability to communicate.

Eye Writer is a device developed for helping people having such inabilities, communicate with the world around them, using only their eye movements. Fig. 1.1 shows the hardware assembly of the proposed solution to the problem. It has a camera mounted on a spectacle frame with artificial light source. This camera will capture the real time eye movements of the user

Fig 1.1: Hardware of assembly of Eye Writer
[www.instructibles.com/Eyewriter]

and this video data will be processed by a MATLAB software on a computer. It will manipulate the cursor on the screen and let the user select whatever he wants to convey. There are multiple pre-loaded routine words in the user friendly GUI of Eye Writer. User can navigate through the menus and sub-menus to select appropriate words which will be typed on the screen as well as spoken on the speaker.

First attempt to develop the software using .NET Framework and OpenCV library resulted in a performance failure, as the nonnative library was unable to give real time results. Which led to the code to be migrated to a more sophisticated computation – MATLAB. Because of its high performance computational and image processing capabilities, it is possible to fetch frames and process them from a live video, which was required to develop a functional prototype of Eye Writer.

1.2 Purpose

The major problem which the prototype is intended to solve is the problem a neuromuscularly disabled person faces when he wants to interact with exterior environment.
Neuromuscular disability syndrome is a medical situation when a person is unable to control total muscular movement of the body. Which means a person is not able to move any limb of his body except eyes. It is obvious that a person having this kind of disability will face a big problem when he wants to communicate.

Fig 1.2: Patient using device
[www.instructibles.com/Eyewriter]

Eye-Writer will help the person having this kind of disability to communicate with person around him by providing a system (combined hardware & software) to type on a digital screen i.e. computer monitor.

The other alternatives available in the market are pretty expensive as compared to this prototype. As they use a BCI-Brain Computer Interface technology, which is based on the Artificial Neural Network (ANN). Which is more accurate technique, but at the same time more expensive. Which in turn makes it less affordable to all the patients around the world.

Major advantages of the prototype are, It is highly cost effective as compared to other alternatives available in current market. Currently very few equipment's are commercially available in market. It has a greater moral value because, for a paralyzed person, being able to talk to someone is the most essential and basic requirement.

1.3 Scope

Primary objective to be served by this device is to help a paralyzed person communicate. Nevertheless some of the functionalities of the algorithm can be extended to support other system like home automation, Home automation, UGV or UAV Navigation, Gaze Tracker for automatic weaponry.

2. Requirements

2.1 Software Requirements

- Windows 7 or higher
- DirectX 9.1 or higher
- MATLAB R2013B or higher
- 2048 MB RAM or higher
- 1.8 GHz CPU or higher

2.2 Hardware Requirements

- External USB Micro cam
    - Resolution : 640*480 px
    - Camera Type : CMOS Sensor
    - Frame Rate : 30 fps
    - Continuous frame Streaming Capabilities
- Light Source (8 LEDs White)
- Micro Camera Mount
- USB port 2.0 or higher
- Integrated or external Speakers

3. Literature Review

Different methods used to detect pupil from live video feed results in different accuracy. Detecting circle is substantially harder as it can't rely on number of points in a contour. A Simple Circle Detection(SCD) algorithm does not work well with this project. It detects every possible circle in an input image. Hence, it gives undesirable outputs. We can't discriminate pupil from multiple circles. Accuracy of SDC is 55-65%.

Another method is Hough Circle Detection(HCD) algorithm. It seems complicated at first but result is better than SDC. As in HDC image taken from live video feed is converted into grayscale to remove noise from an image. For further better result grayscale image is converted into threshold image. How to do all this things with code is discussed later in this document. Main parameter of HDC is minDist that is minimum distance between centers of detected circles. It eliminates circles and generates final circle by choosing best possible circle. That results in accuracy of HCD is 70-75 %. So we have implemented HDC in our project Eye-Writer.

Tracker types:

Eye trackers measure rotations of the eye in one of several ways. Basically they fall into 3 categories:

Fig 3.1: Tracking an eye [2]

Assumption that it does not slip significantly as the eye rotates. To provide sensitive recordings of eye is has measurements with tight fitting contact lenses.

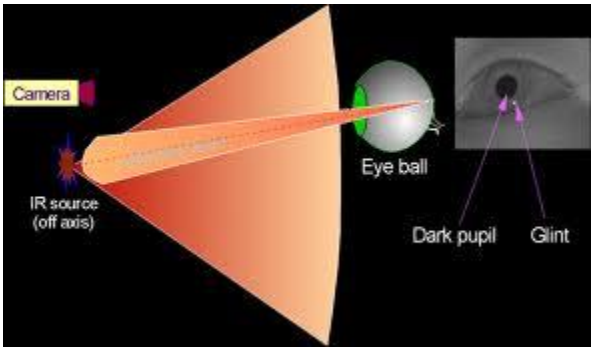


Fig 3.2: Eye tracking using magnetic field sensor[2]

Optical method uses some non-contact, optical method for measuring eye motion. Typically infrared light is reflected from the eye and sensed by a video camera or some other specially designed optical sensor. Then this information is analyzed to extract eye rotation from changes in reflections. Video based eye trackers typically use the corneal reflection and the center of the pupil as features to track over time. A more sensitive type of eye tracker, the dual-Purkinje eye tracker, uses reflections from the front of the cornea and the back of the lens as features to track. A more sensitive method of tracking is to image features from inside the eye, such as the retinal blood vessels, and follow these features as the eye rotates. Optical methods, particularly those based on video recording, are widely used for gaze tracking and are favored for being non-invasive and inexpensive.

Eye trackers measure the rotation of the eye with respect to the measuring system. System s are of two types: head mounted and table mounted. In case of head mounted system eye-in-head angles are measured. Whereas in case of table mounted system gaze angles are measured.

In head mounted system the head position is fixed using a forehead, so that eye position and gaze are the same and head position and direction are added to eye-in-head direction to determine gaze direction. In other cases of table mounted systems, the head is free to move and head movement is measured with systems like magnetic or video based head trackers and head direction is subtracted from gaze direction to determine eye-in-head position.

4. Hardware Assembly

Eye Writer has a pretty straightforward hardware to support the software system. The major hardware component is an Infrared camera mounted on the spectacle frame which will be connected to a computer running the backend software application.

Hardware assembly of the eye writer contents the following components.

1. USB micro camera with infrared censor
2. Infrared LEDs
3. USB connector cable
4. Spectacle frame
5. Lens mount
6. Alligator clips
7. Pack of wire-ties
8. Camera mount
9. Processing unit
10. Digital monitor

Here, the process of assembling the hardware is explained in detail.

1. Make the camera arm

To capture the video, the camera arm needs to hold the camera rigidly in front of one eye. For that we will use a camera arm. One of the best suitable material for that can be 9-guage aluminum wire. USB camera is mounted with support of it. Aluminum wire is coated with plastic in order to electrically isolate camera circuit board from camera arm.

Camera arm is attached with spectacle frame. It is connecting bridge between camera and spectacle frame.

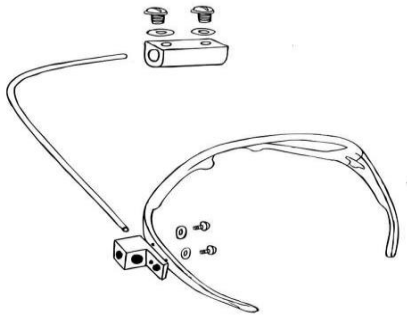


Fig 4.1: Camera arm
[www.instructibles.com/Eyewriter]

2. Mount the camera on the frame.

Put the rigid substrate in between the camera and the wire armature. The camera should be pointed toward the eye of the glasses. Use 4 wire ties to firmly attach all three pieces together.

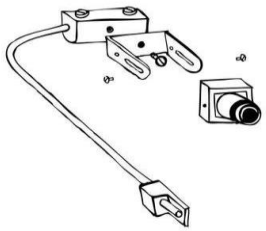


Fig 4.2: Camera mount
[www.instructibles.com/Eyewriter]

3. Fix the IR LEDs on the frame

To provide an artificial light source we will use 2 IR LEDs. We will put those LEDs on the frame using alligator clips and a battery pack. IR light source is used so that, user doesn't have continuous irritation because of normal LEDs used as camera flash.

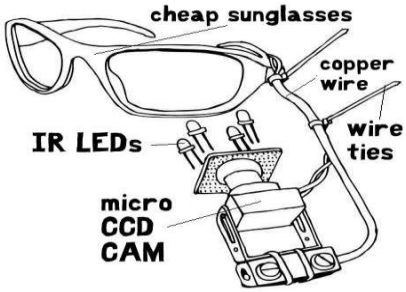


Fig 4.3: Spectacle frame
[www.instructibles.com/Eyewriter]

This is the basic hardware assembly required to make the system functional. Following are sample pictures that represents how the actual assembly will look like.

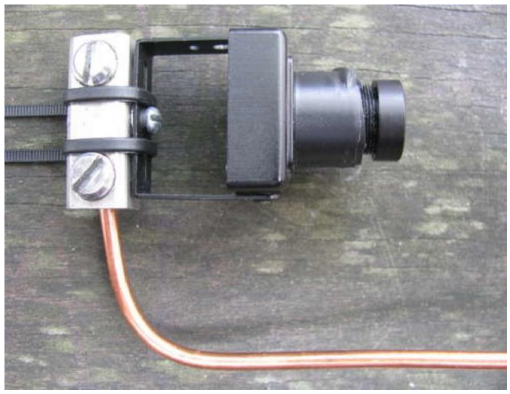


Fig 4.4: Camera mount
[www.diyworld.com/eyegazetracker/]

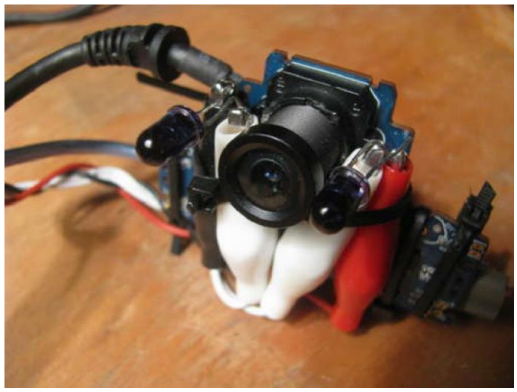


Fig 4.5: Camera with IR light source [2]
[www.diyworld.com/eyegazetracker/]

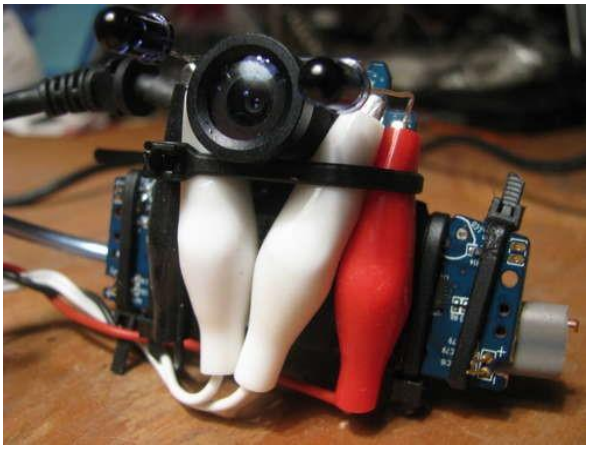


Fig 4.6: Camera with IR light source
[www.diyworld.com/eyegazetracker/]

Fig 4.7: Camera mounted on spectacle frame
[www.diyworld.com/eyegazetracker/]

5. Algorithmic Processing Steps

Software components of the Eye-Writer are the back end essentials are mainly based on .NET framework 4.5 and openCV (open source image processing library). There are three major software components each of which provides a specific functionality.

1. **Eye Tracking Component** – This is the core software component which directly works with hardware and does the tasks such as, tracking eye centroid from video image or live feed.

2. **Data Processing Component** – This component processes the raw data fetched by the former component to calculate the relative movement of the pupil and computes the coordinates to manipulate the cursor on screen

3. **On Screen Keyboard & Typing Tools** – This tools are used in typing mode while an on screen interactive keyboard will be used to allow user to type. Other supportive basic text editing tools allows the user to have a more sophisticated use of the device

4. **Sketching Canvas & Basic** – These are basic sketching tools that allows user to sketch on canvas. With some basic tools for drawing like, different types of brushes, color selection and shapes tools, this component provides a basic MS paint-like environment to user.

All of these components are further elaborated in this chapter

5.1 Eye Tracking Procedure

This component provide following functionalities.

- Fetch the video stream from the camera
- Extract images from the video stream
- Convert the raw RGB image into Greyscale image
- Convert the Greyscale image to Threshold image

5.1.1 Fetch the video stream from the camera

Here the argument to the CaptureFrom function "0" indicates the camera from which the data is to be acquired. 0 indicates data is to be acquired from the integrated webcam. If we want to retrieve the data from the external webcam then we have to replace the value with 1.

If due to critical failure, system is unable to initialize the camera the program exits with an error message.

5.1.2 Extract images from the video stream

Every video is made up of multiple images traversed at a constant frame rate. This images are called frames. Even if we want to get a real time processing of video we need to process one frame at a time.

So, for our device it is crucial to fetch frames from the video and then process each frame acquire the coordinates of the center of the iris, and then move on to the next frame. This way, we need to process each frame separately. For a video captured by camera with a resolution of 640*480 and 30 FPS frame rate, it will have 92,16,000 pixels to be processed each second with color complexity of 24 bit per pixel, it results in immense amount of data to processed which is beyond capabilities of normal computers. Hence we need to drop the frame rate to somewhere near 5-10 FPS to overcome performance limitation. We also need to reduce the color complexity which is explained in next section.

5.1.3 Convert the image to greyscale

While colored images enhances visibility and understandability, at the same time it increases the processing complexity. As color images uses a 24 bit color scheme, it becomes very complex to process the image because there are approximately 16.2 million colors possible at every pixel position on screen.

Processing becomes simpler if the image is converted into greyscale format. As general colored I/O devices use an 8 bit color format which reduces the possible value at each pixel to a mere 256 from 16.2 million.

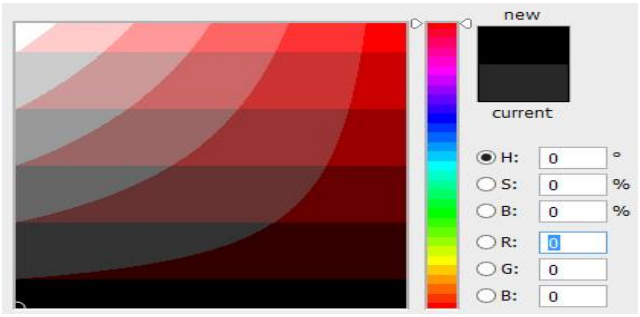


Fig 5.1: Color Spectrum

Further the greyscale image format is easy to process because the value of the color at each pixel is just a
shade of grey (#000 – black to #FFF – white, 256 distinct values). So greyscale image is actually an image from which the color entropies are removed and the raw intensity of light is shown. This way processing becomes efficient as well as simple.

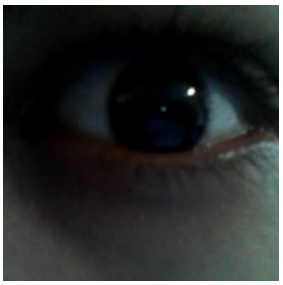


Fig 5.2: RGB Image

Fig 5.3: Grayscale Image

5.1.4 Convert the Grayscale image to Threshold image

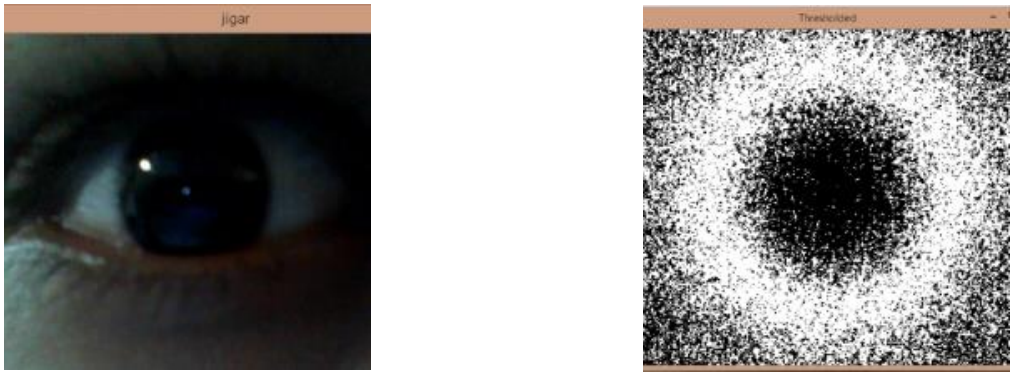
Thresholding is the simplest segmentation method. Application example for Thresholding Separate out parts of an image corresponding to objects which is to be analyzed. This separation is depended on the variation of intensity between the object pixels and the background pixels. We perform a comparison of each pixel intensity value with respect to a threshold to differentiate the pixels from the rest.

Thresholding is one of the simple method of image segmentation. It highlights the part of image on which we need to focus. Separate out regions of an image corresponding to objects which are to be analyze. This separation is based on the variation of intensity between the object pixels and the background pixels.

In our case we are converting the contours from the greyscale image to threshold. Contours represent the points of image or graph which are at the same altitude.

Fig 5.4: Threshold image [4]

Thresholding of image further reduces color complexity of the image to only 2 values from 256 possible values. Each point will now have a 1 bit for color black or white. From this we can easily find the point which forms most perfect circle using predefined algorithm. In our case we are using hough circle detection algorithm. Which detects most circle like figure from given threshold image.



(Real Image) (Threshold Image)

Fig 5.5: Real image and Threshold image of input

5.1.5 Detect the circle from the thresholded image

Successfully detecting a circle depends on multiple factors. Every person has different eye attributes. Certain attributes may affect the efficiency of the algorithm. These attributes are diameter of the pupil, color of iris, end to end displacement of the pupil etc.

Without using proper parameters to limit the effect of the these attributes, the algorithm may fail to detect the expected circle. So we need to feed approximate value of the parameter for it to work properly.

This parameters are discussed in later section of this document. With proper parameters it is fairly easy to find a circle from given thresholded image using MATLAB functions.

6. MATLAB Library

6.1 Introduction

The name MATLAB stands for MATrix LABoratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package) and EISPACK (Eigen system package) projects. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., C, FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

6.2 MATLAB Migration

Implementation of the tracking mechanism and image processing mechanism in MATLAB can reduce the processing time by cutting the frame rate down to half of the original, which results in Real time processing and reduction in lag.

Migrating the original code in the MATLAB environment requires some basic changes in the code and flow of the processing the data. Some of the reasons why MATLAB execution is faster than C# & .NET execution are as follows.

1. The most expensive operation (in terms of time) in the .NET version of the code was Thresholding the entire image frame which is replaced by two sequential cheaper operations – “color subtraction” and “binary conversion”.

   As we only need to work with the part of eye and not whole face or other color intensive images most of the pixel will have only white and black color. So we can eliminate pixels having black color from the greyscale image. And later we will convert the resulting image to binary image, so that we will get image with a white circular void. This whole process is implemented in the C# as a single expensive operation called Threshold.

2. The MATLAB uses matrices internally to process the images and the structure of the objects which handles the graphical data can be altered according to our need and can be mapped to objects of the other structure. This provides extended flexibility of using the data.

3. It is easy to cut the idle frames(frames which have no significant movement to track). So we can cut down frame rate without affecting the quality of the tracking.

6.3 MATLAB object

The object that handles the input video is returned by VideoRead function. The object has many attributes which are set implicitly when it acquires the video. We will see all the attributes in brief.

This are the structure that hold the input videos the attributes of this structures are listed in the figure.

Property	Value	Min	Max
Duration	37.4596	37.4596	37.4596
Name	'irisavi.avi'		
Path	'C:\Users\jigar\Docu...		
Tag	''		
Type	'VideoReader'		
UserData	[ ]		
BitsPerPixel	24	24	24
FrameRate	30	30	30
Height	240	240	240
NumberOfFrames	1122	1122	1122
VideoFormat	'RGB24'		
Width	426	426	426

Fig 6.1: VideoReader object's attributes

1. **Duration** – specifies the duration of the video in seconds.

2. **Name** – name of the file that holds the input video in .avi format.

3. **Path** – Path of the video file

4. **Tag** – Additional tags that are implicit with the file

5. **Type** – type of the parent Function which has returned the object attributes.

6. **BitsPerPixel** – Specifies the number of bits per pixel of image to store the color information of the image. 24 specifies that the image has 24 bit color format.

7. **FrameRate** – The actual frame rate of the input video.

8. **Height** – Specifies the height of the actual video in pixels

9. **Number of Frames** – specifies total number of frames that the object hold (Duration * Number of frames per second)

10. **VideoFormat** – specifies the color scheme and the BitsPerPixel information

11. **Width** – Specifies the width of the actual video in pixels

7. Frame acquisition, processing and iris tracking

The major computational task is performed in following steps, which represent actual flow from data acquisition to result generation. Each of these steps is represented in detail further in this chapter.

- Reading the video
- Frames acquisition from the video
- Convert the image to RGB to Gray
- Subtract the black color from the image
- Convert the grayscale image to binary image
- Perform morphing & find the coordinates of the centroid

7.1 processing loop

The continuous streaming requires real time processing of the data and simultaneous modification of on screen entities. This requires minimum overhead of processing and frame trimming as it is not possible to process all the frames at 60 fps. So these are the necessary and sufficient functions which can efficiently do required processing on video to convert it to the desired format.

7.1.1 Reading the Video

It is necessary to read the video stream either from a camera or from a video file. The data being read must be stored in some object. Following functions are used in this project to work with the video stream.

- **VideoReader()**: to create a multimedia reader object.

Definition of VideoReader(): `OBJ = VIDEOREADER(FILENAME);`

This functions constructs a multimedia reader object that can read in data from a multimedia file.

7.1.2 Frames acquisition from the video

After reading the video we need to acquire frames from that video stream. Every video is made up of sequence of images which are called frames and each video has an implicit attribute called FPS, which specifies the number of the frames (or images) to show from the sequence of the frames per second. FPS can range from 15 to 60 in normal videos. If the FPS is to low then it will degrade the quality of the video and at the same time if it is too high it will generate lag because of performance bottleneck. Here in this project we will be using FPS from 20 to 30 which will have reasonably good performance without degrading the quality of the output video.
To acquire frames from the video we will use following function.

- **Read()**: to fetch the frames form the video (for graphic processing terminology).

Definition of read(): `Img_frm = read(VideoReader obj,Int Index);`

This function reads takes as input the object returned by the VideoReader function And index number of the frame to be read. The frame is stored in the variable mentioned on the left side of the equation and can be used as input to other image procession operations.

These are some of the acquired frames from the video.

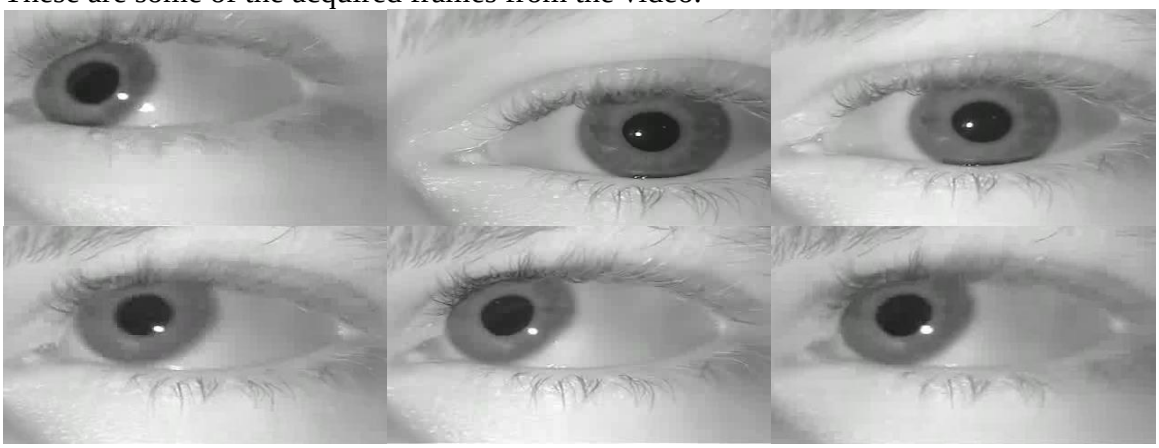


Fig 7.1: Frames acquired from the video [7]

7.1.3 Convert the image from RGB to Gray

The regular colored images that we see every day on a digital screen of a computer follows RGB color scheme with 24 bits per pixel color information (8 bits for Red, 8 bits for Blue, 8 bits for Green – 256 shades of each). Which results in total of more than 16.2 million shades of different colors forming the complete visible light spectrum with depth of 256 * 256 on either axis.

To process each pixel with all of its 24 bits of color information will generate huge amount of processing overheads. We need to somehow reduce the processing overhead if we want to enhance the real time applicability of the prototype. One of the different available methods is, Grayscale conversion.

We can convert a 24 bit RGB image to equivalent 8 bit Greyscale image. Which has reduced color complexity of 256 possible color shades. This gives a large discount on the processing to be done.

To convert the image into grayscale we use the following function.

- **Rgb2gray()**: to create a multimedia reader object.

  Definition of rgb2gray(): `Img_frm = rgb2gray(Img_frm obj);`

The function takes as input a frame from the video stream (an image in general) and converts its rgb pixels to grayscale by mapping each 8 bits of r g & b to 2 bits of grayscale with remaining 2 bit having the value of hue.

7.1.4 Subtract the absolute black color from the image

The eye has pupil of different shades of colors ranging from black to gray but the iris has the absolute black color in an human eye no matter what color the pupil has. So after constructing an image made sole shades of gray, we will have an image as shown in figure. As you can see the iris has absolute black color. So we can simply remove all the parts of image which has black color. This way we will get an image which has a void in place of iris.

Following function is used to subtract the color from the image.

- **imsubtract()**: to subtract a specific color form an entire image

Definition of imsubtract (): `Img_frm = imsubtract(X,Y);`

This function Subtracts image y or constant y from the image x. returns the difference to ima_frm. This is a powerful function if used properly. It can be used to subtract a specific color from an image or to subtract an entire image form another image. The subtraction happens according to the pixel position. A pixel of y is subtracted from the corresponding pixel of x. in case when second parameter is a constant, same value is subtracted from all the pixels of x. In this project we use this function to subtract black iris from the entire image so we can detect the remaining image to track exact location of the center of the pupil.

7.1.5 Convert Grayscale image to binary image

As we saw in section 7.2.3,we reduce the color complexity and hence the processing overhead by converting RGB image to grayscale. That conversion brought down the possible colors to 256 from 16 million. We use yet another conversion to do the further reduction in complexity.

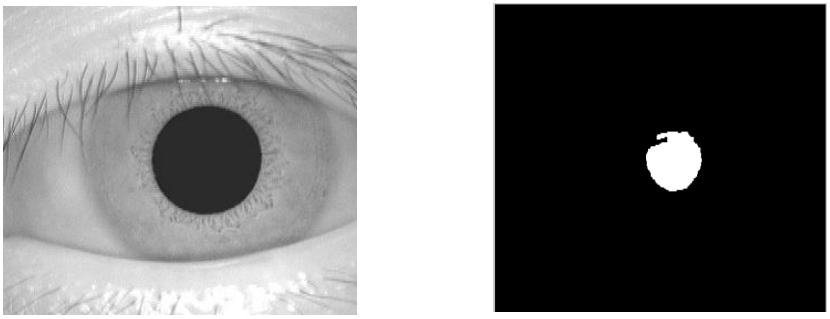


Fig 7.2: Binary conversion [5]

We convert grayscale image from which the black color has been subtracted to a binary image which has only two colors. White and black. Where all the other colors represent value 1 and absolute black color has the value 0.
The function we use to do the conversion from an grayscale or rgb to binary is as follows.

- **im2bw()**: to subtract a specific color form an entire image

Definition of im2bw (): `Img_frm = im2bw(img frm, Float Level);`
This final and ultimate reduction in color complexity results in the image which takes almost no time to process as compared to processing the other images.

The image in the first argument is converted into a binary image with level of conversion specified in the second argument. The represent necessary tolerance of the absolute black.

7.1.6 Perform morphing & find the coordinates of the centroid

Even after doing multiple operation to enhance the quality of the tracking mechanism, Certain morphological operations are still required for proper execution of the algorithm. Morphological operations or morphing ops are used to provide a finishing element to the image.

It is quite possible that lighting conditions, camera resolution and shape of eye may prevent the acquisition of complete pupil. So we perform certain morphing ops to prevent afore mentioned factors from affecting the tracking mechanism.
Following are the morphological operations used in this project

```
Piel = bwmorph (piel, 'close');
Piel = bwmorph (piel, 'open');
Piel = imfill  (piel, 'holes');
```

This operation serves the following purposes.
Bwmorph – close: Performs morphological closing (dilation followed by erosion).
Bwmorph – open: Performs morphological opening (erosion followed by dilation).
Imfill – holes: Fills the holes (empty close areas formed in the image) with adjacent color to complete the image.

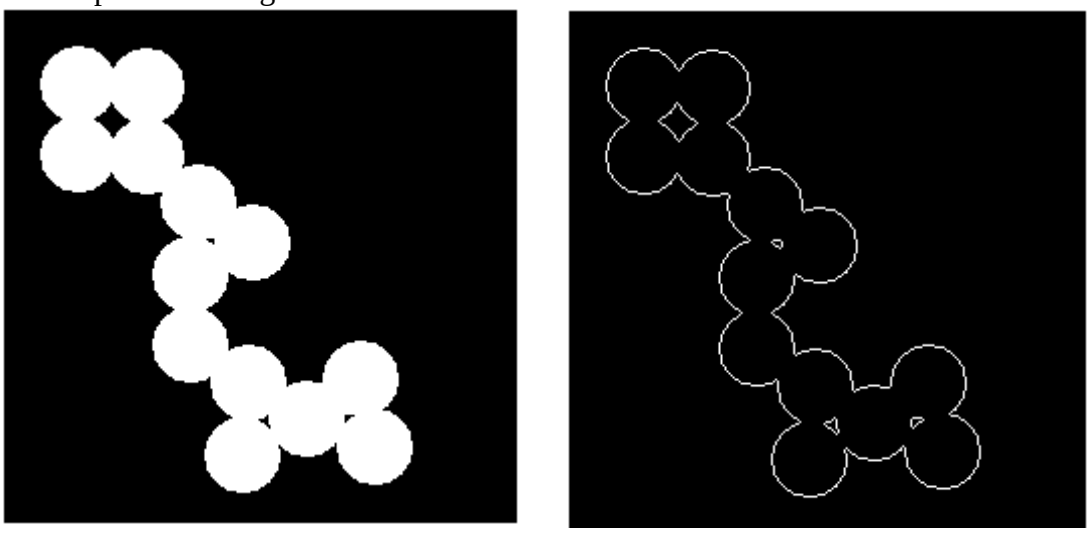


Fig 7.3: Morphing and filling[www.mathwork.com]

After the image has gone through all the above mentioned operations, the coordinates of centroid of iris can fetched from it. Acquiring the coordinates of iris is done in a sequence of operation.

First we acquire all the areas that are plotted in the final binary image. And store those area information in a variable. After that we find the largest area to locate the emphasis center. This will help ignoring all the areas generated because of noise pixels as the largest area would be of the iris only and noise pixels covers comparatively smaller areas. After that we label the

iris' area and acquire its center coordinates which will ultimately be the coordinates of the centroid of the pupil.

The following operations show the above mentioned procedure in terms of the implementation.

1.
```
out_a=regionprops(L);
```

- This will give measuring properties of all the connected components of a binary image.

2.
```
N=size(out_a,1);
    if N < 1 || isempty(out_a)
        solo_cara=[ ];
        continue
    end
```

- Here we count the number of connected components of the binary image which will give us the number of areas to be evaluated.

3.
```
areas=[out_a.Area];
    [area_max pam]=max(areas);
    subplot(211)
```

- This piece of code will evaluate all the available areas from the binary image. It will find out the largest area formed after morphological operation. It is obvious that the largest area would be of iris and all the other smaller areas will be of noise pixels. Subplot function is used to find the center of the largest area. The basic task of the subplot function is to divide the area into m*n grid and draw grid lines at position specified in the arguments.

  This way we can track the center of the iris without actually using any complex and processor intensive algorithm. Which enhances the real time feel of the prototype.

8. Graphical User Interface

8.1 Elemental design of GUI

Eye-writer has an easy to use GUI for novice users. Though it is not directly commercially applicable because of precision hardware would've raised the base cost associated with the prototype, it can still be useful to form a firm basis for a more sophisticated device to be developed.
So here, in this chapter basic GUI is described in detail along with the initial results of the algorithm.

GUI is basically built by the GUIDE (Graphical User Interface Design Environment) and Simulink environment provided in MATLAB.

The GUI has five major components in a 1366*768 pixel resolution which is supposed to be supported by most common purpose computers. Components are,

1. Raw video preview window.
2. Processed video window.
3. On screen keyboard.
4. String output box.

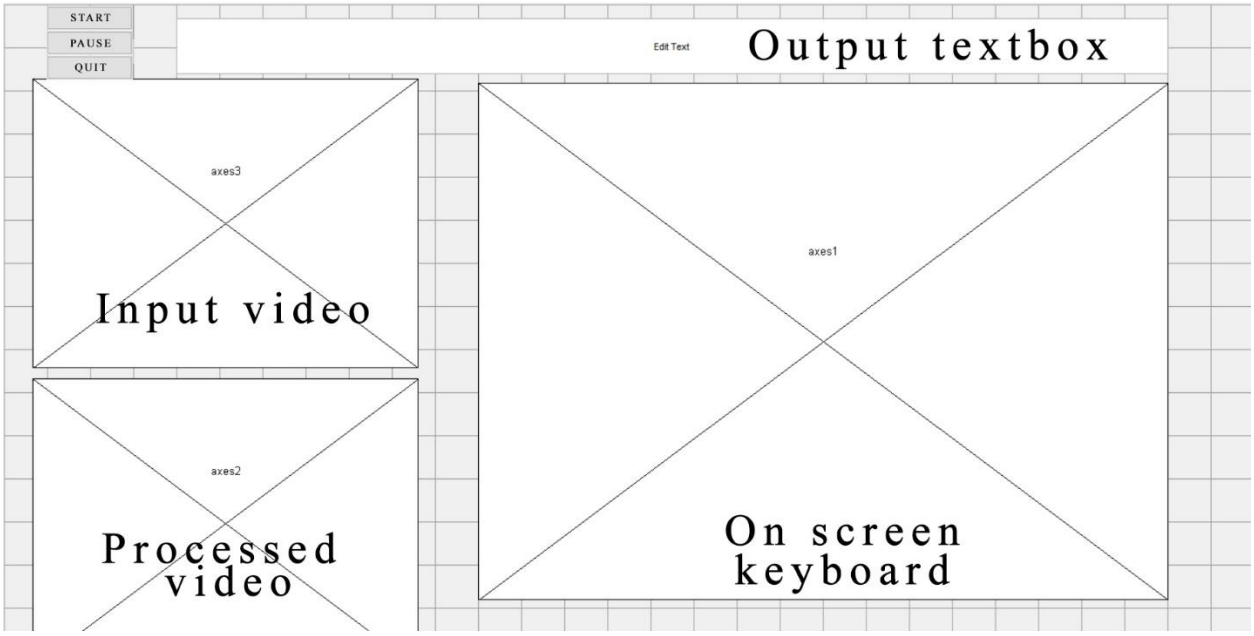


Fig 8.1: GUI plan for Eye-Writer

From the above figure one can understand the simplicity of the proposed GUI. It provides means for both Diagnosis as well as practical use of the device and software. Further each of the components are explained in detail.

Input video preview – Uses the handler mentioned in the function CaputrDevice of Winvideo video handling libraries set. Simply displays the RGB input video stream captured from the USB micro cam attached to spectacle frame.
The window pane in GUI is scaled down to 0.75 of the original resolution(640*480). This is because raw input has little significance for other than initial manual calibration of the mounted camera.

Processed video preview – This window displays the processed feed of the input video. However it is not the binary image which is shown in this window because it is quite unintelligible to us directly. So we rather decided to show a greyscale image so that a person can understand how and where the circles are being detected. This video labels the detected circles with red border.

On screen keyboard – This keyboard will show letters along with some cursor manipulating controls to type the letters. This window pane is scaled to 1.25 of the original resolution so that user can easily look what he/she is typing. The letter which is being typed at the moment are highlighted by a colored rectangle border so that user can see the selection and move the eyes appropriately. It has 6*6 grid which holds the letters and other buttons.

String output textbox – This box holds the letters that are typed. It uses simple string concatenating operations to add or remove letters to the string.

Sample GUI screenshot

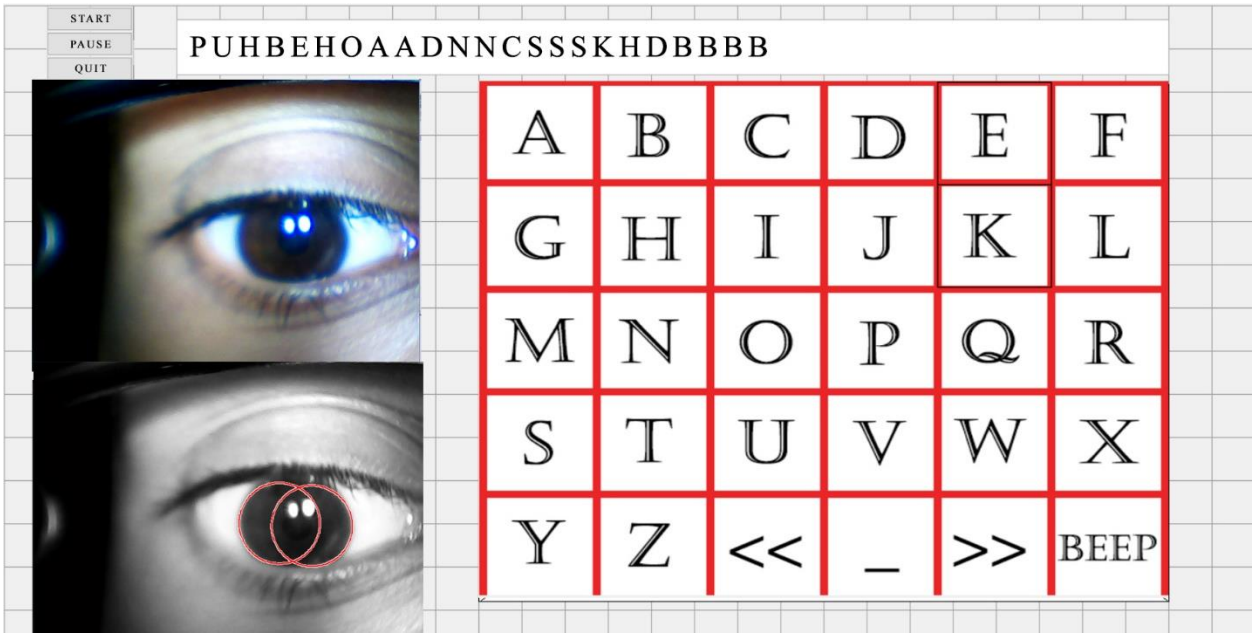


Fig 8.2: Functional GUI screenshot - 1

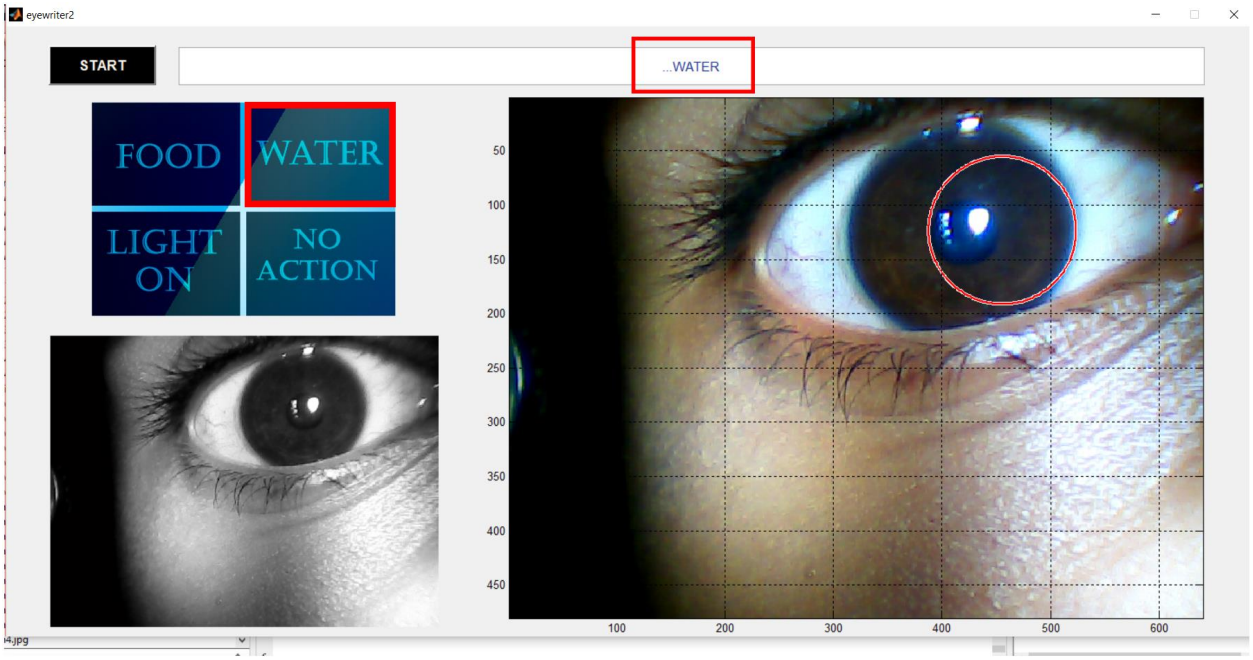


Fig 8.3: Functional GUI screenshot - 2

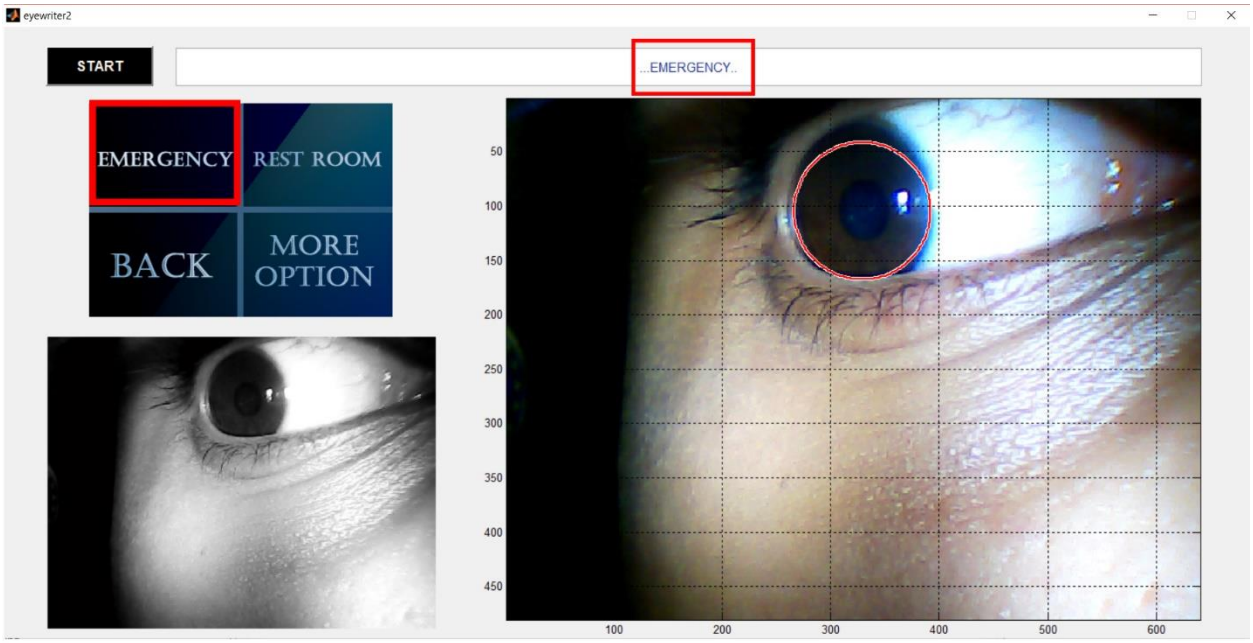


Fig 8.4: Functional GUI screenshot - 3

Fig 8.5: Spectacle frame with camera mount

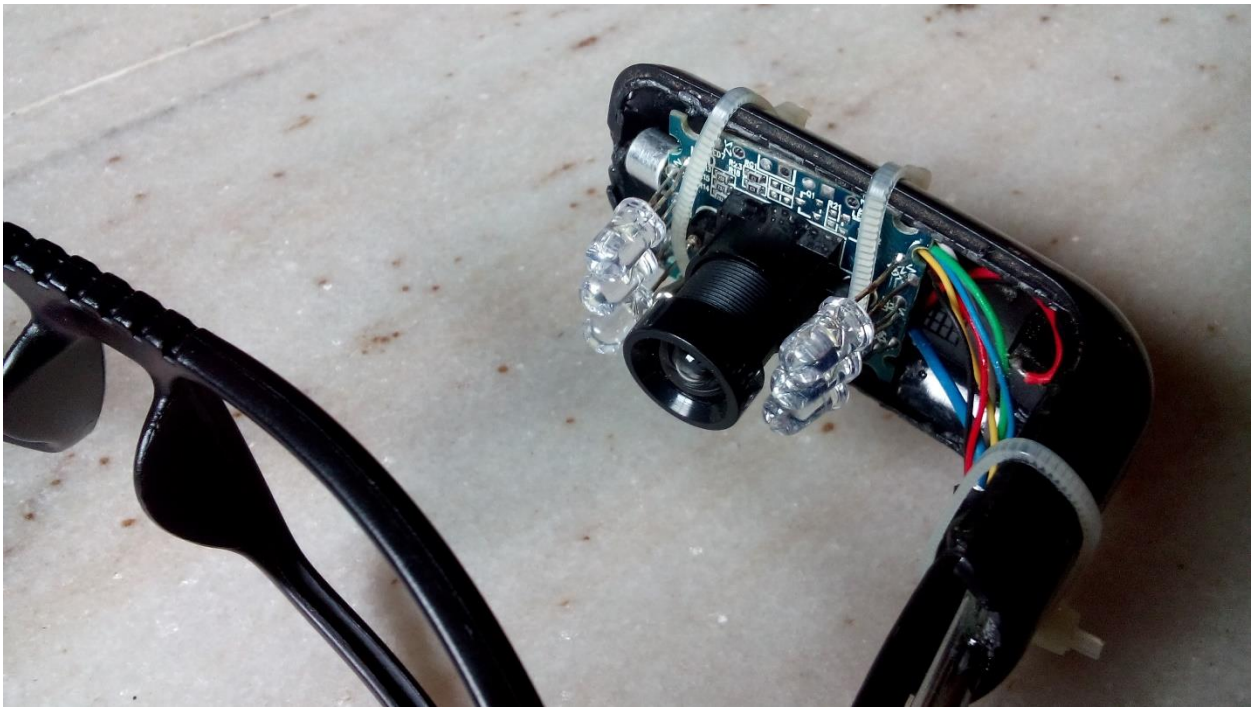


Fig 8.6: Mounted Micro CAM & LEDs

9. Testing & Usability Survey

To test the universal usability and efficiency we conducted a survey on 10 different people. This people were selected at random and had different eye attributes. Which are mentioned in the table 1.1. This attributes were measured by the calibration tool.

USER	PUPIL COLOR	PUPIL DIAMETER (PIXELS)	END TO END DISPLACEMENT (PIXELS)
1	BLACK	202	410
2	BLACK	184	365
3	BLACK	201	401
4	LIGHT BROWN	202	407
5	BLACK	198	389
6	GREY	181	388
7	BLUE	183	397
8	BLACK	203	400
9	BLACK	178	401
10	LIGHT BROWN	190	387

Table 9.1 Users' pupil attributes

This is a sample image of the calibration tool, which was used to measure attributes of the users eye.

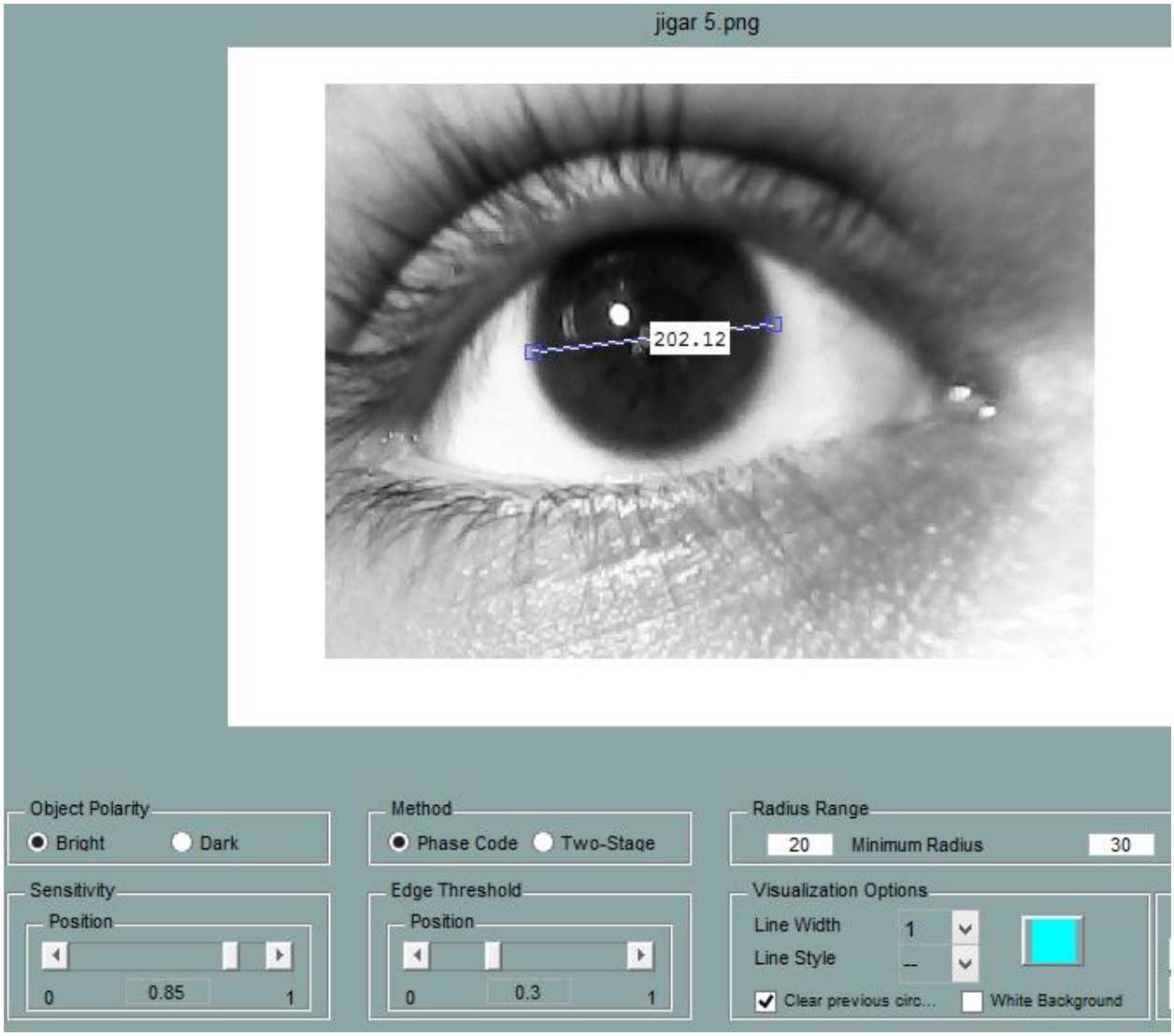


Fig. 9.1 Calibration tool

Series of operations which user had to perform are as follows

Task – 1 Ask for water
Task – 2 Turn the light on
Task – 3 Open emergency menu
Task – 4 Change channel on TV
Task – 5 Go back to main menu

A sample screen shot of step one of the testing process is shown in fig A-2. Same way all the users have tried to perform all the 5 composite tasks. Their success to perform the task is marked by right mark and failure is left blank in the table 1.2.

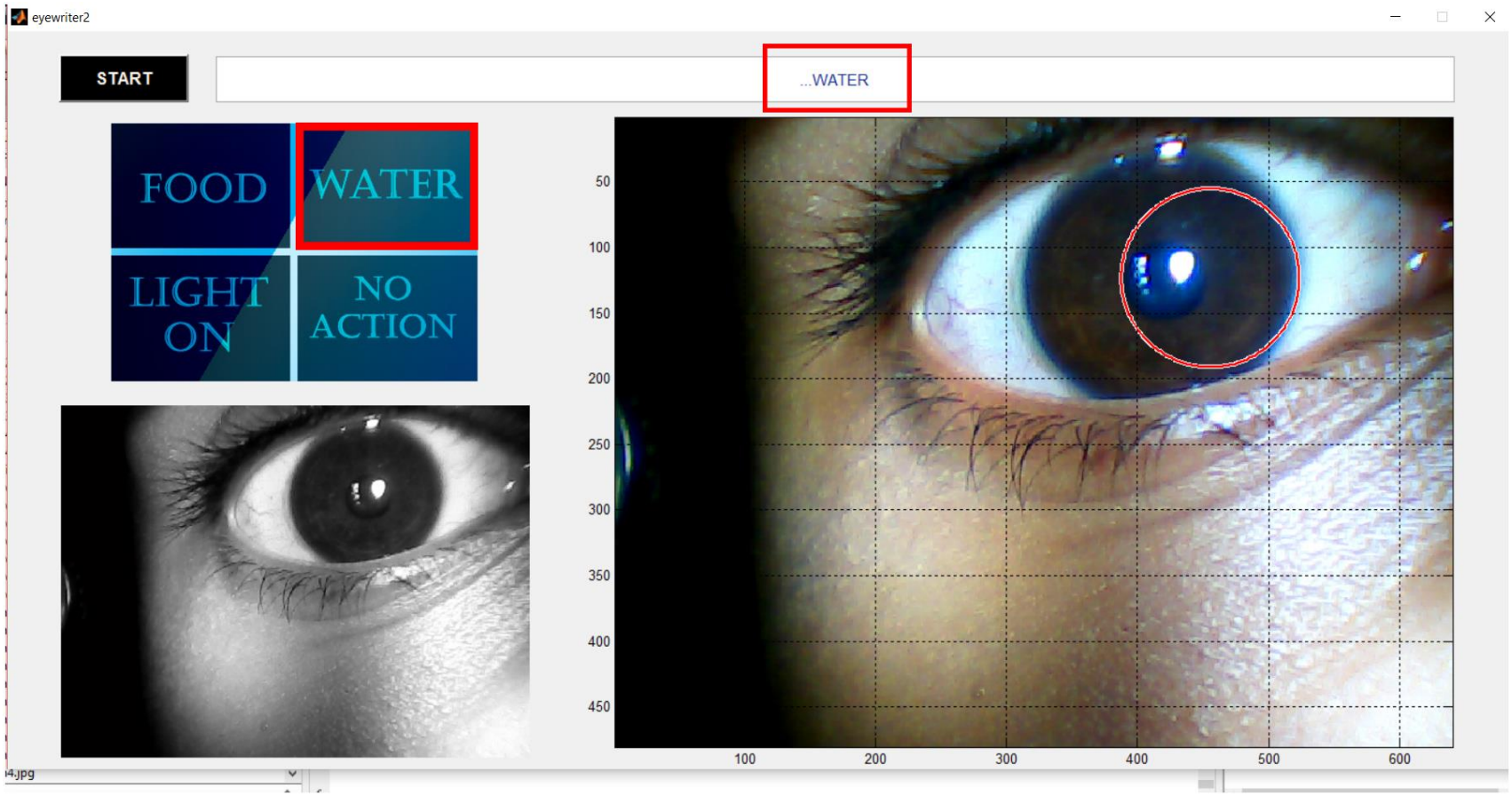


Fig. 9.2 GUI Task – 1 – Request for water

USER	Task – 1	Task – 2	Task – 3	Task – 4	Task – 5
1	✓	✓	✓		✓
2		✓	✓	✓	✓
3	✓	✓	✓	✓	
4	✓	✓	✓		✓
5		✓	✓	✓	✓
6	✓	✓		✓	✓
7	✓	✓	✓	✓	✓
8	✓	✓		✓	
9	✓	✓	✓	✓	✓
10	✓		✓	✓	✓

Table 9.2 Task Completion

Results from this test are quite satisfying as most of the users were able to complete at least 4 out of the 5 given tasks. With an average of 4.1 out of 5.0, we have around 82% successful hit ratio. Which is pretty good considering the fact that all the users were using the device for the first time. And it can be surely said that the hit ration can be increased with more routine usage of the device

By this statistics we can say that the Eye Writer can be a reliable and useful device for persons having neuromuscular paralysis, and it can bring a change in their life by making them capable of communicating with people around them.

10.Conclusion & Future Work

10.1 Conclusion

Eye-Writer is aimed to help the differently enabled persons around the world in their daily life by allowing them to communicate with world around them. This prototype, with its limited functionalities, is the first step towards the process of creating a sophisticated device which will bring a change to the life of its users.

Initial implementation in .net framework was rendered unsuccessful due to the performance lag. However MATLAB implementation has a notable increase in the frame rate that can be achieved, it still needed some more custom and minor modification in algorithm to get a full frame rate required to make the device give a feel of real time interactive device.

A simple and easy to use GUI allows even novice users to use the device without much hassle. While this version of Eye-Writer is good enough to be used in a noncommercial manner, provided with enough time and more suitable precise hardware, this prototype can be made better in several aspect.

10.2 Future Work & Scopes of improvement

There are certain aspects which can be improvised to provide better level of sophistication to the user. For commercial utilization of the Eye-Writer may require the following changes to the current prototype.

- **Use of high precision camera**
  The camera mounted in the current prototype is a general purpose micro USB camera which has infrared filter in its lens as the normal use of a web camera does not require infrared spectrum imprinting.

  But as the performance of Eye-Writer is greatly influenced by alterations in lighting condition, it will be better to use a night-vision or infrared sensitive CMOS sensor in place of a regular RGB webcam.

  It'll be easy to change the current camera because of the modular design of Eye-Writer hardware. So if a Precise Infrared camera is available, it'll be better to use it.

- **Adjustable camera mounts**
  The mechanism which is used in the current prototype to the extent that we can adjust the distance of camera from eye as well as the angle of the camera. But if a more sophisticated version is to be developed, it would be wise to make the mount with greater flexibility in terms of horizontal and vertical placement of camera along with angular movement.

If the position of the light source can adjusted or if the light source can be added or removed as per the requirement of artificial lighting, it'll be better as we can remove extra weight from the frame for a long term use.

- **Responsive voice command**
  A speech processing engine can be used to make device useful to a person with physical paralysis. If a person can talk but cannot move his limbs, it'll make much more sense if the Eye-Writer can be operated using voice. Though the typing mode will be rendered useless, but drawing mode can still be used to let the user have some way of expressing the creativity by making sketches without moving limbs.

- **Using AI Engine and artificial neural networks**
  Even after customizing the detection and tracking algorithm to reduce the overhead, it is still lagging when there are multiple circles appearing on screen.

  An artificial neural network powered by an AI processing engine can utilize parallel relaxation technique to find the pupil pretty quickly as compared to our approach.

  Though the implementation of such network will be hard for the programmer, it'll be exponentially better performance-wise.

REFERENCES:

PAPERS

[1] Free Art and Technology (FAT), OpenFrameworks and the Graffiti Resarch Lab: Tempt1, Evan Roth, Chris Sugrue, Zach Lieberman,Theo Watson and James Powderly (2003), Eye Writer.

[2] "Paralysis Facts & Figures - Spinal Cord Injury - Paralysis Research Center". Christopherreeve.org. Retrieved 2013-02-19.

[3] Bojko, A. & Stephenson, A. (2005). Topic: How eye tracking can help answer usability questions. User Experience, Vol. 4, No. 1.

[4] Pieters.R & Wedel.M (2007). Topic: Visual Attention to Advertising: The Yarbus Implication, Journal of Consumer Research, 2007.

[5] J.Daugman, "High confidence visual recognition of persons by a test of statistical independence", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, pp. 1148-1161, 1993.

[6] Ho, C., and Chen, L., "A fast ellipse/circle detector using geometric symmetry", Pattern Recognition, vol.28, no.1, pp: 117-1995

[7] Tsuji, S., and Matsumoto, F., "Detection of ellipses by a modified Hough transformation", IEEE Transactions on Computers, vol.C-27, no.8, pp: 777-781, 1978.

BOOKS

[1] Maria Petrou, Costas Petrou, "Image Processing: The Fundamentals" Vol. 12

[2] Rafael C. Gonzales & Richard E. Woods, "Digital Image Processing", single reference for matlab image processing. [Third Edition].

WEBSITES

- MATLAB Official - "MathWorks - Support - MathWorks India" http://in.mathworks.com/support/search_results.html?suggestion=true&q=winvideo+asset_type%3A%22Documentation%22&q1=winvideo /..

- "Paralysis - Wikipedia, the free encyclopedia" - 26 May 2016 https://en.wikipedia.org/wiki/Paralysis

- Development Platform

MATLAB - Version – R2013B
Licensed to G. H. Patel College of Engineering & Technology
License Number – 909887

APPENDIX – A

List of Abbreviations

- CV : Computer Vision
- BCI : Brain Computer Interface
- OSK : On Screen Keyboard
- USB : Universal Serial Bus
- UAV : Unmanned Aerial Vehicle
- UGV : Unmanned Ground Vehicle
- ANN : Artificial Neural Network
- LED : Light Emitting Diode
- SCD : Simple Circle Detection
- HCD : Hough Circle Detection
- EOG : Electrooculogram
- FPS : Frames Per Second
- GUI : Graphical User Interface
- GUIDE : Graphical User Interface Development Environment