

---

**ELECTRONIC, ELECTRICAL & COMPUTER ENGINEERING**

**UNIVERSITY OF  
BIRMINGHAM**

B.Eng Final Year Project (EE3P)

Ying Deng

1205690

A portable gaze monitoring system using Raspberry Pi

Supervisor: Dr. Neil Cooke

---

# Abstract

This project report describes the design, development, implementation, improvement and testing of a portable gaze monitoring system. This system is based on the Linux Raspberry Pi micro computer and specially focused on the solutions for the portable design. Enhancements for a more accurate and robust reaction are made for a better performance. The completed system can detect the center position of the pupil and the radius of it. Returned data and recorded video will be displayed as well as stored in local files. Evaluations and testing procedures are applied to the system for its performance on accuracy, speed and hardware parameters such as power consumption, weight and size.

# Acknowledgement

I would like to thank Dr. Neil Cooke for his patient help and professional advices.  
I would like to thank the pioneers and great researchers in relevant areas that generously shared their experience and knowledge that inspired me in this project.  
I would express my gratitude to the participants that kindly volunteered in the testing procedures of this project

---

# Table of Contents

<b>1. Introduction .....</b>	<b>7</b>
<b>1.1 Background .....</b>	<b>7</b>
<b>1.2 Aims and objectives.....</b>	<b>7</b>
<b>1.3 Project timeline.....</b>	<b>8</b>
<b>2. Literature survey .....</b>	<b>8</b>
<b>2.1 Basic information of eye movement and gaze tracking.....</b>	<b>8</b>
<b>2.2 Existing gaze tracking methods and example projects .....</b>	<b>9</b>
<b>2.3 Evaluation methods for gaze tracking.....</b>	<b>10</b>
<b>3. Available options and their comparison.....</b>	<b>11</b>
<b>3.1 Tracking method.....</b>	<b>11</b>
<b>3.2 Source image.....</b>	<b>11</b>
<b>3.3 Camera .....</b>	<b>12</b>
<b>3.4 Processor .....</b>	<b>13</b>
<b>3.5 Software.....</b>	<b>13</b>
<b>3.6 Algorithm .....</b>	<b>14</b>
<b>3.7 Portable design of batteries and weight.....</b>	<b>15</b>
<b>4. The Solutions.....</b>	<b>16</b>
<b>4.1 Hardware choices .....</b>	<b>16</b>
<b>4.2 Software choices.....</b>	<b>18</b>
<b>4.3 Algorithm development .....</b>	<b>19</b>
4.3.1 Stage1-use of existing algorithm.....	19
4.3.2 Stage2-decreasing noise .....	19
4.3.3 Stage3-increasing accuracy (system learning from previous input).....	22
4.3.4 Stage4-from pupil tracking to gaze tracking .....	24
4.3.4 Stage4-increasing speed .....	25
<b>4.4 Hardware overview (diagram) .....</b>	<b>26</b>
<b>4.5 Software overview (diagram) .....</b>	<b>26</b>
<b>5. The Implementation Process/Results .....</b>	<b>29</b>
<b>5.1 Hardware .....</b>	<b>29</b>
5.1.1 Webcam (consideration of light level adjustment, position adjustment) .....	29
5.1.2 Communication channel between processor, display and controlled device .....	30
5.1.3 Portable battery solution.....	31
<b>5.2 Software development on PC .....</b>	<b>31</b>

5.2.1 Installation of software and compiling tools .....	31
5.2.2 Image processing before applying algorithm .....	33
5.2.3 Testing accuracy with pictures and videos .....	33
5.2.4 Testing with real time eye movements .....	34
<b>5.2 Software development on R-Pi: Installation of software and compiling tools.....</b>	<b>34</b>
<b>6. The evaluation of the system's performance.....</b>	<b>35</b>
6.1 Power consumption .....	35
6.2 Weight, cost and size of the whole system .....	35
6.3 Outlier removal before data handling--Error rate of obvious detection mistakes .....	36
<b>6.4 Influence from external environment—natural light, bright light, dim and dark.....</b>	<b>36</b>
6.5 Changes brought by improving the algorithm .....	38
6.5.1 Accuracy tested by 20 preset images of the eye .....	38
6.5.2 Speed test (r-pi's specific algorithm speed, changing image size's speed) .....	40
<b>6.6 Difference between PC and R-Pi.....</b>	<b>40</b>
6.6.1 Detecting accuracy for eye closure.....	40
6.6.2 Detecting speed .....	41
<b>6.7 Cognitive workload testing for the system .....</b>	<b>41</b>
<b>7. Conclusion.....</b>	<b>42</b>
<b>8. Future work .....</b>	<b>43</b>
 <b>Appendix A: Test data.....</b>	 <b>43</b>
<b>Appendix B: Statements.....</b>	<b>55</b>

## References and Bibliography

## Ethics form

## Table of Figures

Figure 1 Dark and Bright pupil images .....	11
Figure 2 Retro-reflectivity of the eye. ....	12
Figure 3 The Hough circle algorithm's theory demonstration.....	14
Figure 4 Project hardware components .....	16
Figure 5 Overall detection process for pupil .....	19
Figure 6 Noise introduced by eyelashes .....	20
Figure 7 Noise introduced by corneal reflection.....	20
Figure 8 Demonstration of pixel counting section.....	21
Figure 9 Demonstration of Double Hough algorithm.....	22
Figure 10 Demonstration of eye zone segmentation.....	24
Figure 11 System hardware overview.....	26
Figure 12: System software overview as state flow diagram .....	27
Figure 13: System software overview as flow chart.....	28
Figure 14: Camera module for the system.....	29
Figure 15: Circuit adjustment for the system.....	29
Figure 16: The Putty server on PC.....	30
Figure 17 Hardware structure of the battery module .....	31
Figure 18: Sample image for accuracy testing.....	38
Figure 19: Detected useful data percentage with different input .....	39
Figure 20: X-Y position recording of reading text .....	41
Figure 21: X-Y position recording of looking at a website .....	41

## Table of Tables

Table 1: Comparison between cameras.....	13
Table 2: Comparison of processors.....	13
Table 3: comparison of portable batteries' regulators.....	16
Table 4: Detailed information of adopted camera.....	17
Table 5: Detailed information about adopted processor .....	18
Table 6: Record of system working time .....	35
Table 7: record of system weight and size evaluation .....	35
Table 8: Statistics of performance and outlier percentage .....	36
Table 9: Statistics of system performance and variation of data.....	37
Table 10: Accuracy testing results .....	38
Table 11: System speed with different output resolutions .....	40

# 1. Introduction

This part introduces the general background of pupil tracking technology and the specifications and a overall timeline for this project.

## 1.1 Background

Pupil tracking relates to accurately locating and tracking of the pupil and its parameters like, shape, color, size and distance between two pupils. Pupil tracking has wide range of applications in medical, biometrics, behavioral research, and so on.

The potential for systems controlled by gaze is attracting a growing degree of interest from the academic and industrial direction. The recent solution<sup>1</sup> from the gaze tracking focused company Tobii shows the trend of gaze controlling going from research use into individual use. The software created by the Eye Tribe<sup>2</sup> allows users to interact with mobile devices by looking at it. These developments in the gaze tracking area have indicated the trend of simpler, more convenient and portable devices for gaze tracking.

The availability of small portable Linux computer promises a new era in portable gaze tracking solutions – for example walking or driving. This project attempts to build a gaze tracking system using the Raspberry PI. Specific focus will be given to the performance and specification requirements for implementing for 700MHz ARM-11 processor and 512MB memory and the use of formal software engineering techniques.

## 1.2 Aims and objectives

The aims and objectives of this project are:

- Build an infra-red monocular gaze tracker based on USB Webcam.
- Design a portable power solution for the Raspberry PI.
- Design a software architecture based around Linux to realize the following functions using standard software engineering techniques:
  - USB image capture into buffer
  - Processing and Parsing of image to detect pupil, occlusion and diameter using Open CV libraries
  - Detection of eye closure and recording to time-stamped buffer / file for storage.
- Performance evaluation of device
- Conduct cognitive workload experiment.
- Performance improvement.

Also, two bench inspections taking place at the end of each term are scheduled for the project, and the

final project report should be submitted before 15<sup>th</sup> of April, and a project poster submitted before 18<sup>th</sup> of April. At the first week of the summer term a poster display will be held.

### 1.3 Project timeline

The timeline of the project are as follow:

- **Autumn term:**

Research for basic background knowledge on gaze tracking; Decide a specific direction for the project based on the general instructions; Work in group to design a solution for capturing the pupil information into system; Build the capturing solution individually; Research for suitable hardware and software solutions for the system

- **Autumn term outcomes:**

Decision of building portable gaze tracking system; Decision of using Raspberry Pi and OpenCV library for the system; Knowledge of gaze tracking technologies; Camera module for the system built

- **Christmas vocation**

Knowledge learning of OpenCV library

- **Spring term:**

Knowledge learning of OpenCV library; Knowledge learning of using the OpenCV library on PC and on Raspberry Pi, under the system of Windows and Linux respectively; Development tools installation and adjustment for PC and Raspberry Pi; Software designing and programming for the system; System implementation and testing; System improvement; Researching for portable battery solution; Portable battery building

- **Spring term outcomes:**

Portable battery solution for the system; Communication within the system and system with PC  
Overall system for pupil detection; Testing results of the system and improved system

## 2 Literature survey

This part provides the reader basic information of the eye movement and gaze tracking, and the history and previous researches and accomplishments are also included. The rest of the literature materials can be found in section 3: Available options and their comparison.

### 2.1 Basic information of eye movement and gaze tracking

The eye is referred to as the “world’s worst camera” (Andrew Duchowski, 2007), for the distortions caused by the curvature of the eye. The information retrieving of the eye is complex in consideration of the types of movement, and the speed of movements.



The movement of the eye can be sorted into the general types of: Fixation: eyes being relatively still; Saccade: rapid movement between fixations; Smooth pursuit: tracking a moving target; Nystagmus: involuntary smooth pursuit or saccade; Vergence: eye moving inwards to fixate on close objects; Vestibular: rotation to compensate for head movement; Drift: eyeball wandering due to imperfect motor control; Microsaccade: short rapid movement. These raw movements also combine into complex movement types such as long glances or concurrences. Movements that acquire visual information are fixation, smooth pursuit and vergence. The eye being never still adds difficulty for gaze tracking and minimizing errors.

During reading or surfing the internet, pupil movements convey useful message about how people processes the received visual information. Considering this, pupil tracking can be used for psychological studies, human and machine interface, driver fatigue detection and so on. The advantage of gaze tracking as an input is its similarity to the instruction system and the operational methodology of a computer mouse. It allow users to handle the computer more naturally without having a big problem. In addition, it provides an opportunity for disabled people to operate computers and other devices, which generally use keyboard, mouse, and remote controllers that require hand movements.

There are different types of pupil tracking algorithms that have varying accuracies and applications. But they can be categorized in two main groups: intrusive and non-intrusive. Intrusive methods, also known as electro-oculography (EOG) contact sensors or electrodes to the eye or skin around eye for tracking. Though these method is more accurate in tracking, they are generally not user friendly and hard to setup. Non-intrusive methods, known as video-oculography (VOG), advance in aspects of being user friendly, low cost, flexible and easily performable.

Most of the implemented approaches to detect the eye position and the pupils in video image rely on the notion of image gradient (Sirohey et al., 2002; Peng et al., 2005; Ji & Bebis, 1999, Wang & Sung, 2001; Halir & Flusser, 1998; Smith et al., 2000; Li et al., 2004). The pupil borders have an obvious contrast that enables the detection of these borders using methods based on the image gradient.

## **2.2 Existing gaze tracking methods and example projects**

Numerous researches have been conducted in the field of eye tracking and yield to different methods and applications that will be mentioned below.

In 2011 Liaghatdar et al.<sup>3</sup> built an eye tracking system for investigating person behavioral changes in video. Pupil is detected and segmented by Otsu method and genetic algorithm respectively. They used Circular Hough transform to modeling pupil and kalman filter for tracking and compensating errors.

In 2010 Asadifard andShabanzadeh<sup>4</sup> created a changeable eye detection system by capture of the entire face and analysis of relevant distribution function. Face location is approximated by viola-jones face de-

tector and the eye is located by structural information.

In 2008 Panev et al.<sup>5</sup> tracked eye by matching pupil with an ellipse and approximate detected pupil position by GLD and used a particle filter with limited particles to track pupil.

In 2007 Komogortsev et al.<sup>6</sup> designed an eye controlled computer interface with kalman filter. They focused on kalman filter (AFKF) that changes the eye movement to real-time perception and modeling of human visual system.

In 2011 Jian-nan et al.<sup>7</sup> used a fixed camera with IR illumination to create bright pupil. A morphology operator is applied for accurate contour of pupil and they modeled the pupil by an ellipse.

In 2002 Zhu and Ji.<sup>8</sup> designed a real time eye tracking that combines Kalman filter and mean shift under IR illumination. The bright pupil is created by illuminating pupil by IR across camera optical axis and pupil is seen as a bright spot in the image.

In 2002 Ji et al.<sup>9</sup> tracked gaze and face to monitor driver's vigilance. NIR4 illumination was used to creating bright pupil. By image subtraction, pupil position is approximated. Thresholding by Kullback information distance contributed to accurate detection of pupil.

### **2.3 Evaluation methods for gaze tracking**

The evaluation of an eye tracking system is technically difficult because of the eye's characteristics of being spherical and chromatic, and also because of its dynamic and small scale movements (the drift and microsaccade).

One useful evaluation technique is to record from the two eyes simultaneously and compare the vertical rotation records. The two eyes of a normal subject are very tightly coordinated and vertical gaze directions typically agree to within  $\pm 2$  minutes of arc (RMS of vertical position difference) during steady fixation. A properly functioning and sensitive eye tracking system will show this level of agreement between the two eyes, and any differences much larger than this can usually be attributed to measurement error.<sup>10</sup>

Another way of evaluating the eye tracking system is using the Fitts' Law (or ISO 9241-9). Xuan Zhang et al.<sup>11</sup> used this standard for their pupil tracking system and compared the differences of the Long Technique, Short technique, Spacebar technique and the Mouse. The evaluation involves both the accuracy and the comfort level of the system.

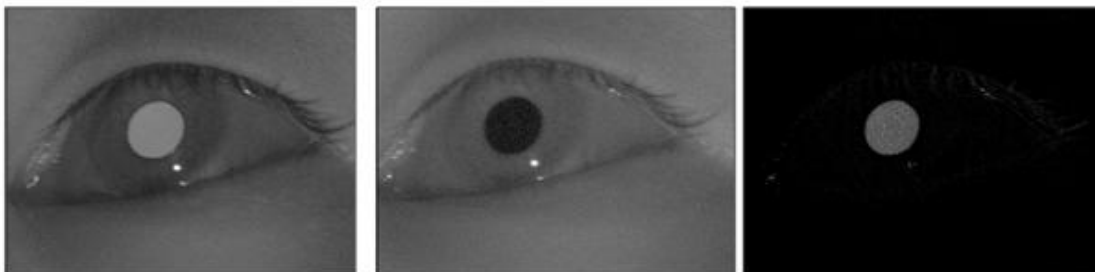
### 3 Available options and their comparison

#### 3.1 Tracking method

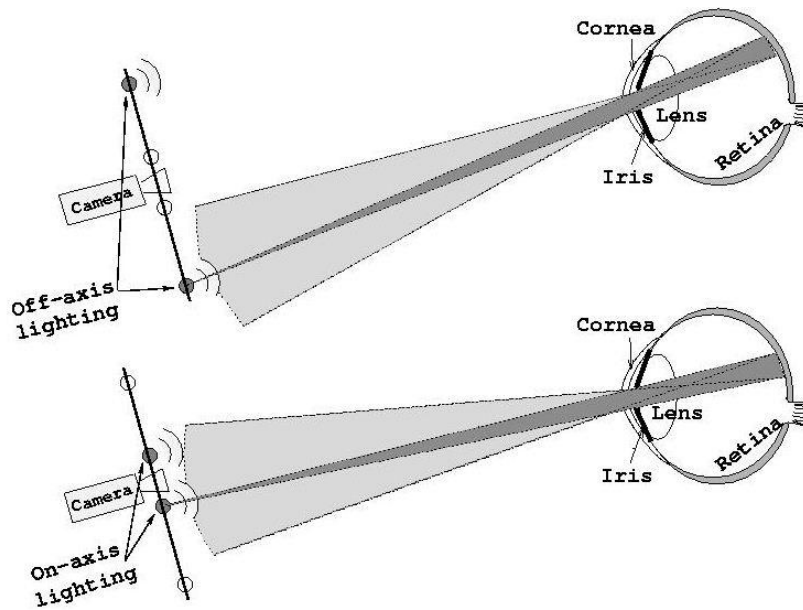
The existing gaze tracking methods related to the input data type that the system gets, can be divided into three main methods: skin electrodes, contact lens, and head mounted devices. The skin electrodes method (Electrooculogram, EOG) measures the electrical difference of the eye's retina and cornea, by attaching the electrodes around the eye, hence estimating the rotation of the eyeball, deciding the gaze position. The main constraints are the limited facial movements, repulsion that users can feel, and the low detection accuracy. The contact lens method tracks the pupil position with high accuracy and comparatively simple algorithm by placing contact lens into user's eye and mirroring the movements by a magnetic coil or similar device. It cannot detect blinking; users feel discomfort of the contact lens; it can only measure eye movements within restricted areas. Furthermore, the device itself is costly and cannot apply to vast user range. The head mount device (HMD) method adopts cameras or lights on a headwear to acquire eye images. The gaze position is detected from the relative relationship between the obtained images from camera and those from the user's eyes<sup>12</sup>. Though the disadvantages such as lower accuracy and the blocking the eye's sight, the method is popular because of its low cost and easy implementation for multiple users. Among the three methods, this relates closely with image processing, and requires more complex algorithms or even calibration to identify the movement. The HMD method is preferred for the project because of its feasibility compared to contact lens, and higher accuracy compared to EOG method.

#### 3.2 Source image

The most commonly seen methods to segment the pupils from the eye zone are by using the dark/bright pupil effect. Many researches adopt one or both of these effects for pupil detection<sup>13</sup>. The effect is obtained by illuminating the eye zone near-infrared light emitting diodes (IR-LED) from different positions related to the camera optical axis. Due to the retro-reflectivity of the eye, the bright pupil effect is achieved when a light source is placed very close to its optical axis (Figure 1 and 2).



**Figure 1 Dark and Bright pupil images**



**Figure 2 Retro-reflectivity of the eye.**

When the light source is not on the camera's optical axis, a dark pupil is seen<sup>14</sup>. Both effects can segment the pupil area, though neither of them can eliminate the impact of the eyelashes or the cornea, nor can they eliminate the first Purkinje image, known as glint or corneal reflection. The glint is useful in the estimation of the eye-gaze direction<sup>15</sup>; however, it will introduce potential error in pupil detection if a pixel counting algorithm is adopted.

### 3.3 Camera

Based on the previous decision of using HMD method for pupil tracking, the hardware components are hence required to have capabilities for image processing, either digital or analogue. The color of the images does not have influences on processing, but only the brightness has influences, both for dark and bright pupil effects.

There are plenty of choices of cameras available, for example, analog and digital cameras, or using another sorting method, charge-coupled device (CCD) cameras and complementary metal-oxide-semiconductor (CMOS) optical camera. The first two types of camera are sorted by the camera's sensor type, and the output signal type can be changed by its data converting module. The Web cameras are usually camera with CMOS sensors and data converting modules that transmits digital serial USB standard image signals. Most processors cannot read directly from CCD or CMOS cameras unless a first in, first out (FIFO) buffer is connected between them. In consideration of resolution, all of them can satisfy the need of pupil tracking, but the most suitable size of resolution based on the computing ability of the processor lies on the CMOS and USB Web cameras. As a comparison, the typical types of these cameras are listed below in Table 1 with their specifications.

Specifications	Type	OV9650 (CMOS)	VC112 (CCD)	Kinovo B3 (Webcam)
Extra supporting components		FIFO buffer	FIFO buffer	-
Resolution		Up to 1280*1024	752 *552	640*480
Price		Medium	High	Low (12.99 GBP)
Frame rate		Up to 30 fps	Up to 30 fps	Up to 30 fps

Table 1: Comparison between cameras

### 3.4 Processor

With a close relation to the previous consideration of the camera module, the system, as an image processing system, requires basic critics of being data-intensive, and compute-intensive. It will need to be capable of handling large amount of data in real time, requiring efficient data transfer and high computing power. The available choices include digital signal processor (DSP) for imaging applications (TMS320C6416, TMS320DM642), system on chip (SoC) development board for imaging applications (S3C2410<sup>16</sup>, S3C2440), and single-board, micro-scale computer (Raspberry Pi).

TMS320C64x is an advanced, very long instruction word (VLIW) processor, suitable for imaging applications with its high computing power and large on-chip memory. It also provides enhanced direct memory access (EDMA) and cache to efficiently transfer data to/from off-chip memory. The TMS320CM642 DSP uses the c6416 as its core processor.

Specification type	TMS320DM642	S3C2410/2440	Raspberry Pi (BCM2935)
Processor type	DSP	ARM processing core	Micro computer
Frequency	720MHz	200M/266MHz	700Mhz
External dependencies	Video-standard converting module chips	Video-standard converting module chips	None
Suitable camera type	Analog	Digital/analog	digital
Size and Weight	Large (in DSP)	Medium (in DSP)	Small
Price	High	Medium	Low

Table 2: Comparison of processors

### 3.5 Software

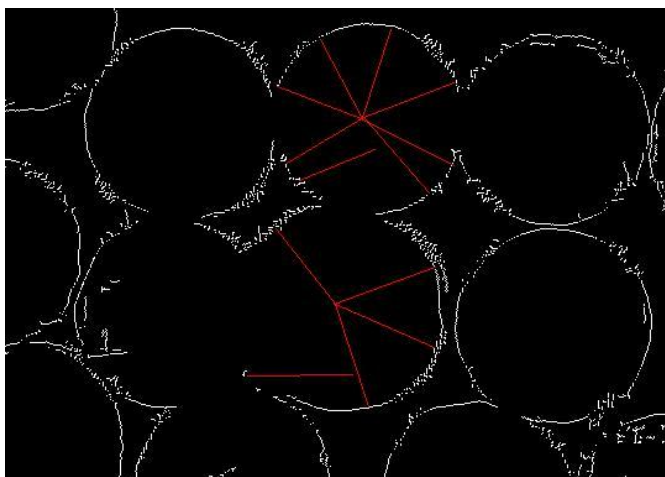
The system's chosen processor being Raspberry Pi, there is large range of available software platforms for the system. The Raspberry Pi supports multiple programming types and multiple developing platforms. From the degree of existing development level, the available software can be divided into these following types: already fully and specifically developed systems that have limited functions, such as the ITU gaze tracker; library of functions with dependencies fully developed by installation supporting various image processing aims, such as OpenCV; Programming language and its developing platform that has the highest degree of freedom of functions, but without any established dependencies for image processing, in this case, C/C++/Perl language.

Considering the ease of developing the system, especially establishing the image retrieving and pro-

cessing function, the method of developing it without any existing libraries or applications would be of high difficulty, low stability and also time-costly. On the other hand, the already well-developed applications such as the ITU gaze tracker and the Starburst<sup>17</sup> lack flexibility on altering the algorithm for performance enhancements, as well as the problem of not being supported by the selected hardware platform--Raspberry Pi. At this stage, the ideal software platform for the system will be OpenCV, a library of programming functions specially designed for computer vision and image processing. OpenCV library supports multiple programming languages, such as C/C++, Java, Matlab, and it can be installed in both Windows and Linux systems, which can be an advantage for future designing processes because it will be more convenient and fast to complete and improve the algorithm on PC and then transfer to the Raspberry Pi's Linux system. OpenCV requires a local compiler for its applications. The Raspberry Pi's Linux-based system provides a pre-set compiler called GNU Compiler Collection (GCC) for C language. On the PC Windows platform, the Visual Studio package can act as the compiler for the program.

### 3.6 Algorithm

Various previous research projects in gaze tracking have provided great experience and samples of algorithm for this project. Pupil tracking algorithms can be divided mainly as two types: feature based and model based. Feature-based approaches detect and localize image features related to the position of the eye; model-based approaches do not explicitly detect features but rather find the best fitting model that is consistent with the image. The Starburst algorithm (Li et al, 2005) is a both feature-based and model-based approach that searches for the pupil's contour and match those points with an ellipse<sup>18</sup>. The ITU gaze tracking algorithm (San Agustin et al), however, uses a different algorithm that finds the pupil region and returns the centre of mass as the pupil's centre<sup>19</sup>. The most commonly used algorithms also include the pixel counting method, circular edge detection, contours detection, Harr features, etc. One of the most fast and simple algorithm suitable for pupil detection is the feature-based circular contour detecting algorithm from the OpenCV library—the Hough Circle algorithm. This algorithm finds the edges of an image by calculating the gradient of pixel values, and then uses the Hough transform to detect potential circles with a voting system. This is a diagram explaining how the algorithm works.



**Figure 3 The Hough circle algorithm's theory demonstration**

The following is a generalized pseudo code for Hough Circle algorithm:

1. *Train the shape by building the R-table<sup>20</sup>:*  
     *For all points on the boundary*  
     *Compute orientation  $\Omega$  (the gradient  $\pm 90^\circ$ )*  
     *Add an  $(r, \beta)$  entry into the R-table at a location indexed by  $\Omega$*
2. *Quantize the Hough transform space: identify maximum and minimum values of  $x_{ref}$  and  $y_{ref}$ ,  $\Phi$ , and identify the total number of these values.*
3. *Generate an accumulator array  $A(x_{ref}, y_{ref}, \Phi)$ , and set all values to 0*
4. *For all edge points  $(x_i, y_i)$  in the image*  
     *Do*  
     *Compute the orientation  $\Omega$  (the gradient  $\pm 90^\circ$ )*  
     *Compute possible reference points  $(x_{ref}, y_{ref})$*   
     *For each table entry, indexed by  $\Omega$*   
     *For each possible shape of orientation*  
     *Compute  $x_{ref} = x_i + r \cos(\beta + \Phi)$*   
      *$y_{ref} = y_i + r \sin(\beta + \Phi)$*   
     *Increment  $A(x_{ref}, y_{ref})$*
5. *For all cells in the accumulator array*  
     *Do*  
     *Search for maximum values of  $(x_{ref}, y_{ref}, \Phi)$*   
     *Return the position and the orientation of the shape in the image*

The algorithm detects the circles with user-set features of radius range, amount of circles, and distance between circles. It chooses the largest votes for the required amount of circles. The accuracy and speed depend on the preset parameters. If a learning module can be added to the system later, it is possible that the system can adjust the parameters by itself and hence largely increase the accuracy and working speed according to different input condition (relatively immune to the environment's influence).

### 3.7 Portable design of batteries and weight

The Raspberry Pi model B uses 5V power supply via micro-USB or GPIO header, and 700mA of maximum current. In order to achieve a portable design of the system, it is necessary for the Raspberry Pi to have a portable power supply, ideally using rechargeable batteries. The power supply will also need to provide power for the web camera module, so the current might be larger than 700mA, presumably 1000mA as maximum. An ordinary AA battery provides 1.5V in voltage and around 1000mAh in power. In order to provide a stable voltage supply to the system and enough large current to drive it, it is essential that some kind of regulator and stabilizer is added between the power and the computer.

There are several available choices for the regulator and stabilizer, namely the linear regulator, the RC-model ultimate battery eliminator circuit (UBEC), DC-DC converter, integrated battery box. Table 3 is a comparison between the choices:

Choice	Advantage	Disadvantage
Linear regulator	Lowest cos. Easy to implement.	Inefficient Need of a large regulator and large heat sink. Heavy and large compared to DC-DC converters.
UBEC	Low cost (GBP 3). More efficient than a linear regulator	Inefficient-- when the battery's voltage drop beneath 5V, the remaining power will be wasted. Heavy and large compared to DC-DC converters.
DC-DC converter	Low cost (GBP 3). More efficient Easy to implement. Minimum weight.	Soldering needed
Integrated battery box	No soldering needed. Stable and sturdy	Expensive (GBP 10 - 16) Heavy and large compared to DC-DC converters. Unable to change battery with low cost if damaged.

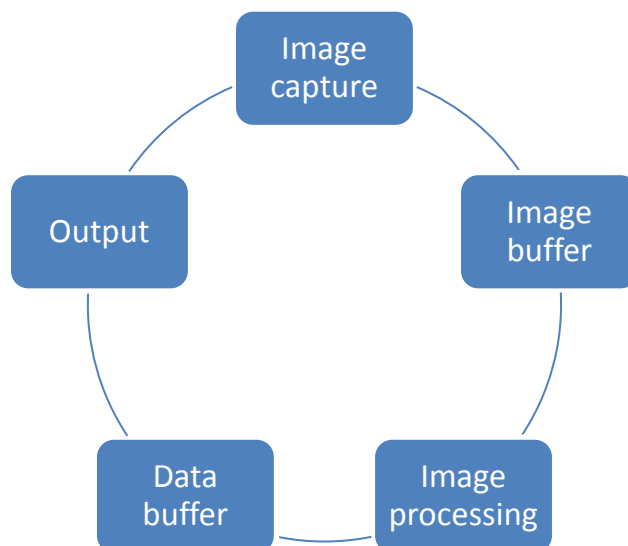
**Table 3: comparison of portable batteries' regulators**

Based on the above analysis of pros and cons, the best choice for the system appears to be the DC-DC converter. My choice of DC-DC converter is a "Solar Boost DC-DC 3V-5V Adjustable Power Supply Module" with average efficiency up to 94%, that can provide 1000mAh output current, and adjustable output voltage controlled by a rheostat.

## 4 The Solutions

### 4.1 Hardware choices

The hardware components chosen for this project are as follow, shown by Figure 4.



**Figure 4 Project hardware components**

The component for image capturing is the web camera.



Model	<b>Kinobo B3</b>
Price	11.99 sterling pounds
Material	Metal
Lead	1.3m, plastic
Accessory	1.4V LED*6, Light control knob
Focus adjustment	hand-adjusted
Maximum Resolution	6 mega pixels
Maximum Frame-per-Second	30
Used Resolution	320*240 pixels
Used Frame-per-Second	10

**Table 4: Detailed information of adopted camera**

This model of web camera is chosen for this project for several reasons. Firstly, its frame rate and resolution can satisfy the project requirement; secondly, the hand-adjusted method of focus helps to produce clearer captured images, which has a clearer edge of shapes, and this can contribute to the accuracy of the used algorithm; thirdly, as a portable system, the sturdiness of the system should be considered, and the metal shell of this web camera as well as the structure of the lens—it is protected by outer shell and does not require removing any shell to convert it into a infrared camera, increase the suitability of it to this project.

However, this model of web camera has its weak points. One of the most significant weaknesses is that the metal shell adds great weight to a camera bearer. This weakness has lead to the further consideration of mounting structure of the camera. Another significant weakness is that this model of camera requires hand adjustment to change focus. As the position of the camera may not stay the same when mounting the device on people's head, it is possible that the system will need hand adjustment to find a suitable focus every time before use. This implies that the system's operational complexity will be increased.

The Raspberry-Pi can satisfy the other four functions: buffering the image and the data, processing the image, and act as a controlled device by the system, as a main reason of selecting this integrated micro computer for the project.

<b>Price:</b>	<b>GBP35</b>
<b>SoC:</b>	Broadcom BCM2835 (CPU, GPU, DSP, SDRAM, and single USB port)
<b>CPU:</b>	700 MHz ARM1176JZF-S core (ARM11 family), Broadcom VideoCore IV
<b>GPU:</b>	OpenGL ES 2.0 (24 GFLOPS), MPEG-2 and VC-1 (with license), 1080p30 h.264/MPEG-4 AVC high-profile decoder and encoder
<b>Memory (SDRAM):</b>	512 MB (shared with GPU)
<b>USB 2.0 ports:</b>	2 (via the built in integrated 3-port USB hub)
<b>operating systems:</b>	Composite RCA (PAL and NTSC), HDMI (rev 1.3 & 1.4), raw LCD Panels via DSI, 14 HDMI resolutions from 640×350 to 1920×1200 plus various PAL and NTSC standards.
<b>Audio outputs:</b>	3.5 mm jack, HDMI, and, as of revision 2 boards, I <sup>2</sup> S audio (also potentially for audio input)
<b>Onboard storage:</b>	SD / MMC / SDIO card slot (3.3V card power support only)
<b>Onboard network:</b>	10/100 Ethernet (8P8C) USB adapter on the third port of the USB hub
<b>Low-level periph-</b>	8 × GPIO, UART, I <sup>2</sup> C bus, SPI bus with two chip selects, I <sup>2</sup> S audio +3.3 V, +5 V, ground

<b>erals:</b>	
<b>Power ratings:</b>	700 mA (3.5 W)
<b>Power source:</b>	5 volt via MicroUSB or GPIO header
<b>Size:</b>	85.60 mm × 53.98 mm
<b>Weight:</b>	45 g (1.6 oz)
<b>Operating systems:</b>	Debian GNU/Linux, Raspbian OS, Fedora, Arch Linux ARM, RISC OS, FreeBSD, Plan 9

**Table 5: Detailed information about adopted processor**

The Raspberry Pi micro computer model B is chosen for the project as a result of trade off between cost and functionality.

The benefits of using this computer as the core processor are:

- No consideration of signal preprocessing (USB to USB instead of USB to serial/parallel binary)
- No consideration of external buffer and storage for images
- Comparatively low price, small size and light weight
- Comparatively high processing speed
- Convenient communication between input, output and the processor itself (USB interface)

There are inadequacies for this processor too, listed below:

- Limited signal transmitting speed (restricted by the USB standard)
- Unnecessary I/O ports and module taking up extra space and weight
- Inability to display unless externally connected
- Inability to achieve wireless communication unless extra module added

In the designing process the limitations from the processor will be minimized.

## 4.2 Software choices

The software choices of the designing process are as follow.

- For program development and enhancement  
Windows PC + Microsoft Visual Studio 10 express + OpenCV library
- For testing and evaluation  
Raspberry Pi micro computer + OpenCV library +GCC compiler

The core algorithm chosen for the system is the cvHoughCircle. It reads a grey scale image and returns the center positions expressed in  $(x_i, y_i)$  of pixels and the radius expressed in pixels. The returned data is stored in the form of  $(x_i, y_i, r_i)$  in a OpenCV owned sequence<sup>21</sup>, each circle's data in the form of an array as one element in the whole sequence. The method to read from the sequence of returned data is to use the cvGetSeqElem(\*Seq, i) function.

The main enhancement in the algorithm will focus on analyzing the pixel values of a detected circle, and using double confirmation for the circle.

### 4.3 Algorithm development

#### 4.3.1 Stage1-use of existing algorithm

The cvHoughCircle algorithm is chosen to be the core of the system. The original function of the algorithm is to find the required number of circles from an image.

**Void cvHoughCircles (CvArr \*image, void \*circle\_storage, int method, double dp, double min\_dist, double param 1 = (100,0), double param 2 = (100,0), int min\_radius = 0, int max\_radius = 0)** <sup>22</sup>

After detection of the circles it is necessary to store them and show the results on screen for visual demonstration. **float \* pointer\_name=(float \*) cvGetSeqElem(image\_name,** the number of elements in the sequence, starting from 0); The output to screen is achieved by **cvNamedWindow ("window\_name", CV\_WINDOW\_AUTOSIZE);** and **cvShowImage ("window\_name", image\_name).**

The overall process of the detection is:

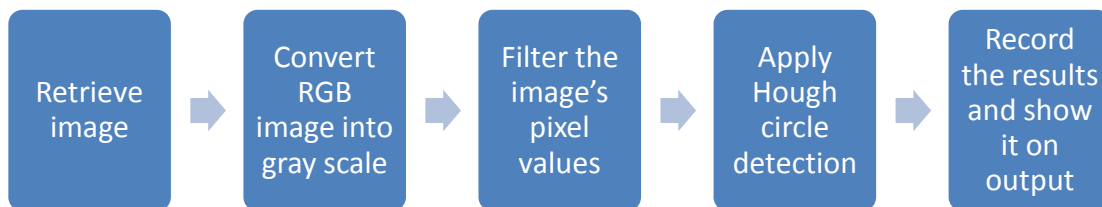
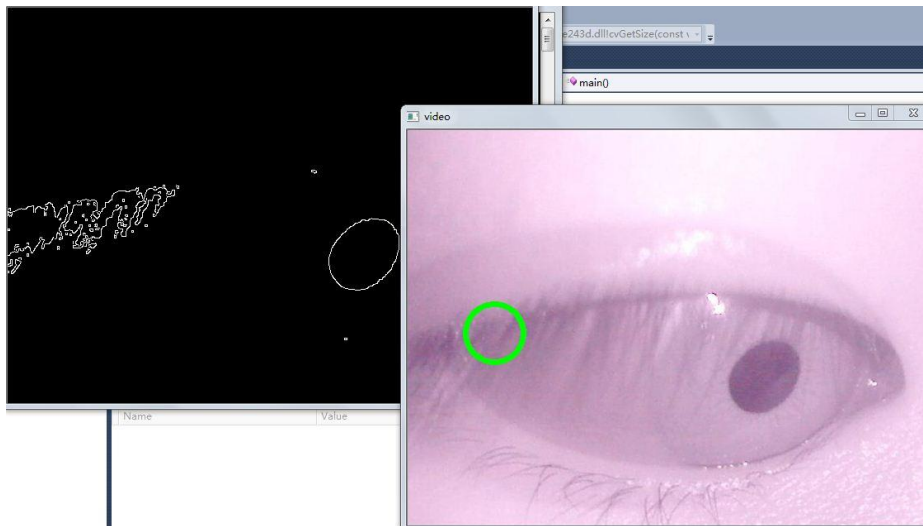


Figure 5: Overall detection process for pupil

#### 4.3.2 Stage2-decreasing noise

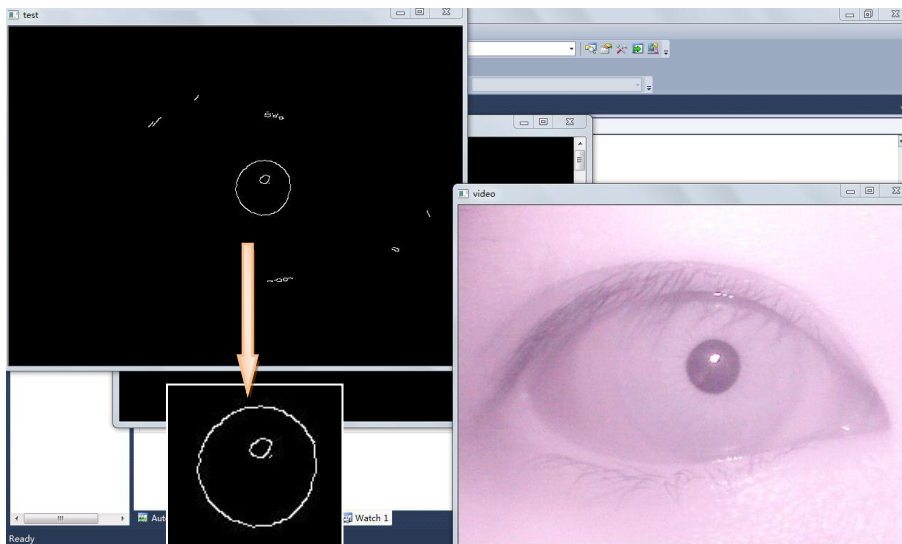
The main noise introduced into the system is from the existence of eyelashes and corneal reflections.

The noise from the eyelashes appears to be a dark area with no regular shape, but mainly appears to be intricate in pixel values, as shown in the image below. The left image is the result of a Canny edge detection. The fact of the eyelashes being intricate will result in large number of detected edge points and based on the algorithm's theory of using the edge points in the voting system, this large number of edges may result in the detected circle of the pupil having a less vote than a wrongly detected circle in the eyelashes area.



**Figure 6 Noise introduced by eyelashes**

The noise from the corneal reflection is in the form of a bright spot in the image. It is also called as “the first Purkinje image”. In this system, as it is not used for the position tracking because of the fact that pupil diameter monitoring and eye gesture recognition will only need a relative displacement<sup>23</sup> between different frames of images, instead of absolute displacement. In other words, this reflection will not contribute to the pupil tracking process. In the pupil detection process, it is treated as a noise because it may cover a portion of the pupil area, and hence causing false detection of edges, inputting noise into the algorithm. This will consequently lower the vote of the correct circle, or create a higher-vote circle than the correct one. The following image shows the influence of the corneal reflection to the edge detection.



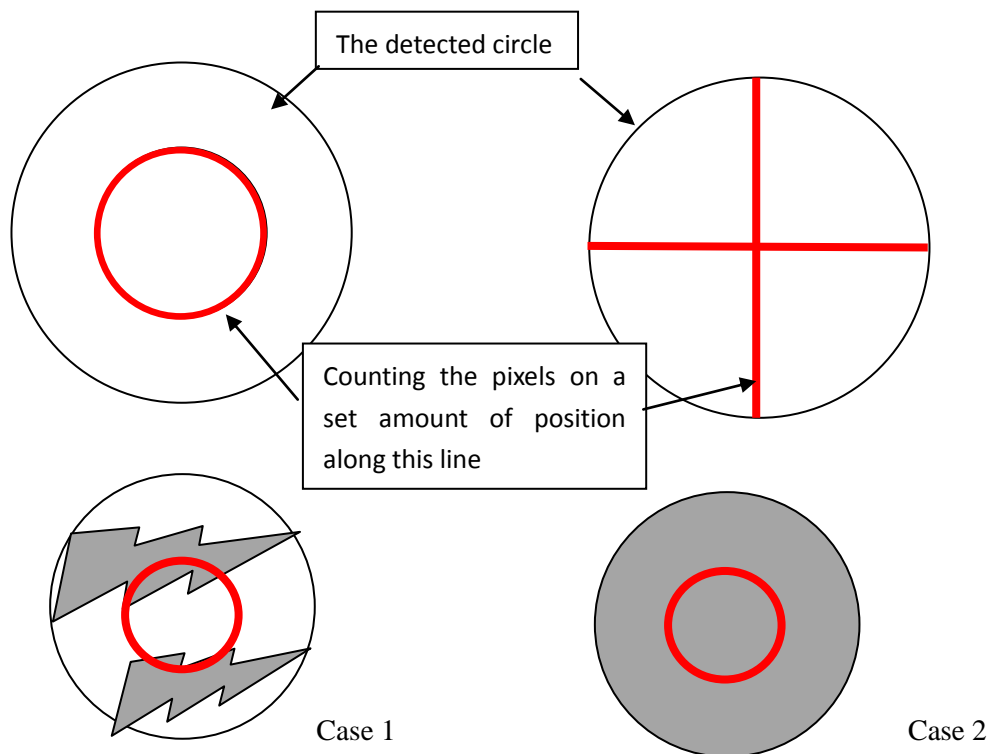
**Figure 7 Noise introduced by corneal reflection**

The Hough circle detection algorithm is in the package of OpenCV, and hence difficult to alter the algorithm’s code and change its functions. This being true, it is possible to let the algorithm to be a generator of possible results and afterwards filter out the most accurate one.

In order to reduce the noise levels introduced by these two sources, the idea of pixel counting is combined

to the algorithm. The idea is based on the fact that the pupil area in a dark pupil image is nearly all black—over 99% of the pixels in the area have values less than 50 (only untrue if the corneal reflection is inside the area). The eyelashes, though having more edges, has actually less pixels in black, the percentage varies with different person, but it will hardly exceed 70 percent of black pixels percentages. Based on this fact, it is possible to count the pixels inside a detected circle and build another voting system beside the edge based one in the Hough algorithm. By setting a loose parameter of the algorithm to return more than one circle, and computing the average pixel values of the circles, it is possible to ensure only the pupil's corresponding circle being recognized. This voting system will be based on the gray scale image, so that the amount of edges in the image will not impact this system.

There are two most simple and possible ways to achieve this pixel counting function, the first is counting a circle within the detected circle, and the second is counting along the diameters in a set axis. As the detected circles will vary in diameters, the total value of counted pixels inside a circle should be divided by a variable corresponding to the circle's diameter. The following graph shows the idea of this function.



**Figure 8 Demonstration of pixel counting section**

Example: Case one being rejected because of detected dark percentage is less than threshold; case two being accepted; Threshold being 50%.

This voting system is easy to construct and effectively reduces the error rate from 40% to 5% from the testing result of a 20 frames of random eye movement images. The most obvious improvement is in the condition of the eye looking downwards.

The pixel counting voting function can effectively eliminate any wrong detections caused by a corneal reflection, but in the case the eyelashes are dense and only have small portions of white pixels, the func-

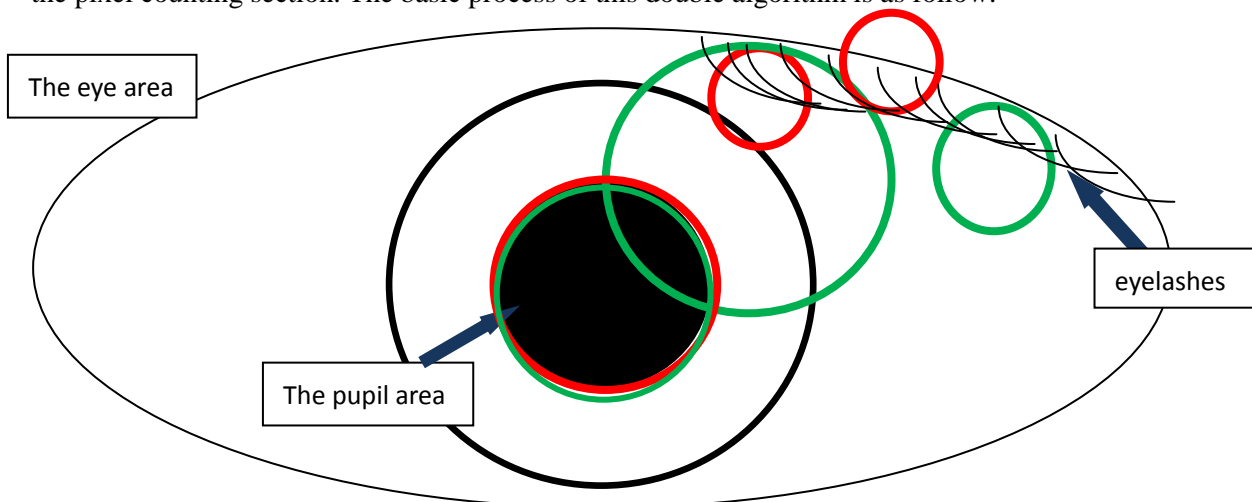
tion will be less effective for filtering out the best result. There is one difference between the eyelashes and the pupil is the average value of pixels. The eyelashes area can be featured by applying a binary conversion to the gray scale image, largely reducing the area of eyelashes. This can be achieved by using the function of cvThreshold.

**cvThreshold (const CvArr\* src, CvArr\* dst, double threshold, double maxValue, int threshold Type )<sup>24</sup>**

#### 4.3.3 Stage3-increasing accuracy (system learning from previous input)

The main target of this part of enhancement is to further reduce the error caused by the eyelashes and reflections, and also reduce the level of influence of blurred images caused by focusing the camera at a wrong distance. In this part the enhancement will be made from two aspects. One of the aspects is to improve the quality of image pre-processing, i.e. before the image data is sent into the algorithm. The other is to develop a time relevance of the frames hence lower the weight of a wrong circle in a specific frame between two correctly detected frames.

The Hough circle detection algorithm can be adjusted by changing the parameters dp, param1, and param2. The enhancement of the performance can be achieved by applying the algorithm twice with different parameters, with different images. The first time applies it with a binary image input and loose parameter, acts as the candidate generator. The second time applies it with a gray scale image input and strict parameter, acts as the filtering input. All the candidates will be matched to the filter input to see if they are detection of a circle or a false detection from an arc in the noise areas. Because the second Hough circle detection is based on the gray scale image, the detected edges of the noise will not be the same as the first while the pupil will remain. This will help to match the two set of detected circles, and by setting appropriate thresholds, this enhanced function can largely reduce the number of candidate circles sent to the pixel counting section. The basic process of this double algorithm is as follow:



**Figure 9 Demonstration of double Hough algorithm**

Green circles: detected circles from the first Hough algorithm (Cir 1, 2, 3 from left to right respectively)

Red Circles: detected circles from the second Hough algorithm (Cir 4, 5, 6 from left to right respectively)

The double Hough algorithm will accept the Cir 1 as a pupil candidate and process it to the next stage to the pixel

counting, and reject Cir 2 and 3. This is because the reference circle, namely the Cir 4, 5, 6 is matched to each of the circle candidates of Cir 1, 2, 3, and only Cir can satisfy the threshold criterions of the radius' difference and the circle center's difference.

The Pseudo Code for this algorithm will be:

*Begin*

*Process the source image for a gray scale image (img1)*

*Process the source image for a binary image (img2)*

*Apply Hough circles algorithm with parameter set 1 to img1, get a sequence of candidate circles (seq1[])*

*Apply Hough circles algorithm with parameter set 2 to img2, get a sequence of candidate circles (seq2[])*

*FOR each circle in seq1[]*

*FOR each circle in seq2[]*

*IF*

*The two circle's radius difference is less than threshold\_radius*

*AND The two circles' difference of the position in the X axis is less than threshold\_Xposition*

*AND The two circles' difference of the position in the Y axis is less than threshold\_Yposition*

*THEN*

*Accept the circle from seq1[] by storing the code of it into an array a[]*

*END IF*

*END For*

*END for*

*Proceed a[] and seq1[] to the next stage.*

*END*

By referencing to neighboring results, the future data processing can be easier than without this process. For example, 3 groups of data in a sequence being (100, 100), (210, 300), (120, 100) will not be a great sequence for an eye gesture, and a great possibility of a wrong detection in time 2. By applying this time-relevance enhancement, the groups of data will be stored as (100, 100), (110, 100), (120, 100), which can be an effective portion of a gesture "pupil moving left".

As the system work with a continuous data input from the web camera, the relation of a frame and its neighboring frames is that the previous position and the future position of the pupil may be near to the current one. One of the characteristics of a false detection caused by eyelashes is that it will usually be far away from the pupil, in the case of looking aside and looking up.

There is a possibility of using the previous detected data to be a reference of the current detection. If the detected pupil's distance of the neighboring frames exceed a certain level, then the current detection will be suspended and only if the next frame detects a circle that is closer to the current detection, will the current detection be considered as a saccade, else it will be identified as a fault detection caused by the eyelashes, and be replaced by an average value of the previous and future detection, to be stored for analyzing and sequence matching.

The drawback of this enhancement of accuracy is that it will need the next moment's data to distinguish

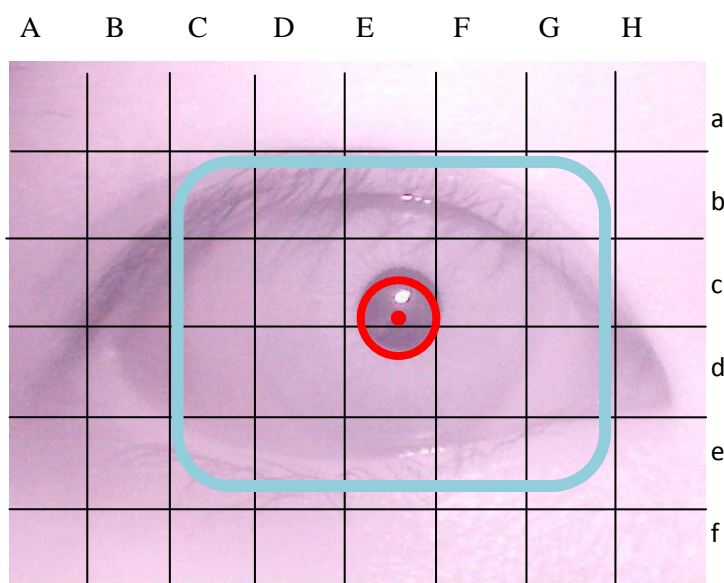
the pupil's saccade with a false detection. If the system only considers the previous data, the large position movements starting from a saccade or the eye closure will need two cycles to be accepted. Either using the previous information only or using both the previous and next data will result in a delay in the system. This means that the data available for analyzing or controlling will be the previous data instead of the current one. This will slow down the system by one cycle.

Based on the theory of the system, as the input is comparatively stable between two frames, the sudden wrong detection is unlikely to happen. (The most possible wrong detection comes from the particular position of the gaze that causes the eyelashes or reflections become obvious.) According to the 100 frames of image used for testing, there is little appearance of the targeted error of this enhancement. As a result of the analysis, this enhancement will not be made to the system.

#### 4.3.4 Stage4-from pupil tracking to gaze tracking

The system can not only monitor the pupil's diameter change and direction change, but also be advanced to a control of any other systems using the eye gestures, for example, controlling a computer, or controlling a robot. This is demonstrated on the bench inspection by controlling an window from opening and closing using the closure and fixation of the eye.

By developing a sequence matching library for the system, it can recognize more gestures and hence achieve more complex controlling orders. The resolution of the image is currently  $320 \times 240$  pixels, and it is reasonable to separate the image into  $8 \times 6 = 48$  sub-areas and construct a sequence library for these libraries.

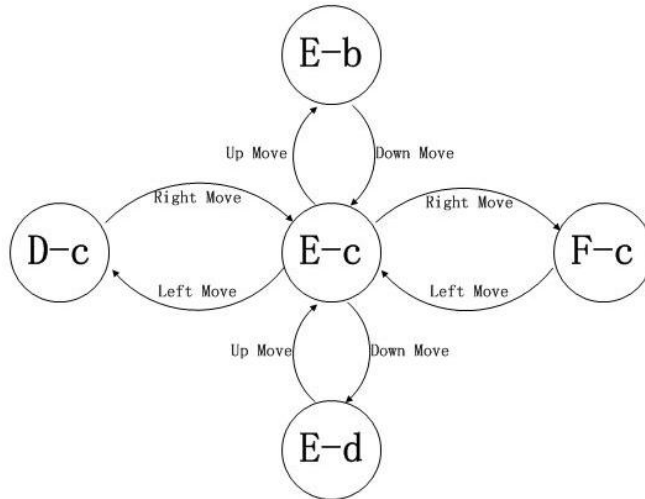


**Figure 10 Demonstration of eye zone segmentation**

The above image grid is a demonstration of how the area is separated. The demonstration's pupil position is located in the area of E-c (indicated by the red circle, and the center is also specified), and the possible position for the next frame will be the neighboring areas between C-b and G-e (indicated by the blue



square). The reason of considering both the D\F and the C\G position is because at certain times the movement of the eye is fast and the camera working in 10 frame per second, which is 100ms per frame, may not catch the middle state. By allocating an action code for every possible movement and building a finite state system, it is possible to use the results of this pupil tracking system to output eye gesture instructions to other systems.



**Figure 11 Example state diagram of area E-c. Diagonal transition not shown on diagram**

The above state diagram is just a demonstration of part of the possible movements of E-c. Those moving from left-down to right-up is not shown in this diagram, but they should be taken into consideration during the actual development of this gesture recognition system.

The gesture will be unsafe if only defined by two states and one transition (e.g. A-a to A-b), and will be too restrictive if too many states and transitions is required (e.g. A-a to A-b to A-c to A-d). Also, deliberately moving the eye is harder than moving hands or legs, so a complex gesture will actually increase the accuracy but lower the feasibility of a system. It is important to find a suitable combination of gestures for a system that is both easy to achieve and hard to be confused by any natural eye movements.

#### 4.3.4 Stage4-increasing speed

The Raspberry Pi has a processor of 700 MHz CPU, which is slower than most of the PCs. The PC used for the system development has a 2.27 GHz CPU, being 3 times faster than the Raspberry Pi. The above fact has caused the system to be much slower when running on the Raspberry Pi.

The bottle neck of the speed issue is outputting the image to screen, as a way of observing the performance of the system. If the displaying section is deleted, the computer can work and output at a maximum of 30 frames per second, and the Raspberry Pi can work and output at 10 frames per second; if the displaying section is added, the computer can work and output at a maximum of 10 frames per second the Raspberry Pi can only work and output and display at about 1 frame per second, which is unacceptable for a pupil diameter monitoring system or an eye gesture controlling system. The above data is based on

outputting 320\*240 resolutions to the screen on both processors. With a resolution of 640\*480, which is the maximum resolution for the camera, the computer can work at about 3 frames per second and the Raspberry Pi cannot correctly detect or display the images. All of the data mentioned is introduced in following sections. In order to observe the performance of the system on Raspberry Pi, the resolution for displaying should be reduced to 160\*120. With this resolution, the system can work at approximately 10 frames per second, satisfying the demands for a pupil diameter monitoring system. If the observation of performance is not required, the 10fps requirement can be easily achieved.

#### 4.4 Hardware overview (diagram)

As a conclusion from the available options and comparison, the overall hardware structure will be: Kinobo B3 web camera and a hat as the image capture module, Raspberry Pi as the core processor, the DC-DC converter with 4 AA batteries as the power supply, and the PC with Ethernet router as an external adjusting input and displaying. The hardware structure can be shown as the following diagram.

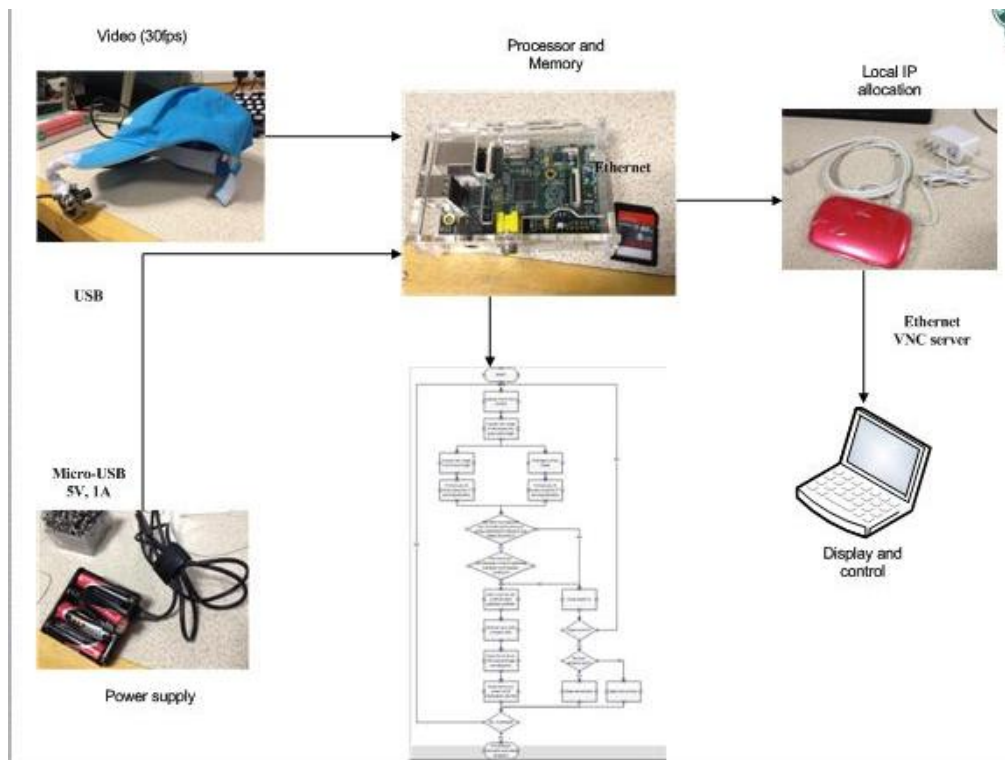


Figure 8: System hardware overview

#### 4.5 Software overview (diagram)

The software for the system will be demonstrated in the form of a state-flow diagram and a flow chart.

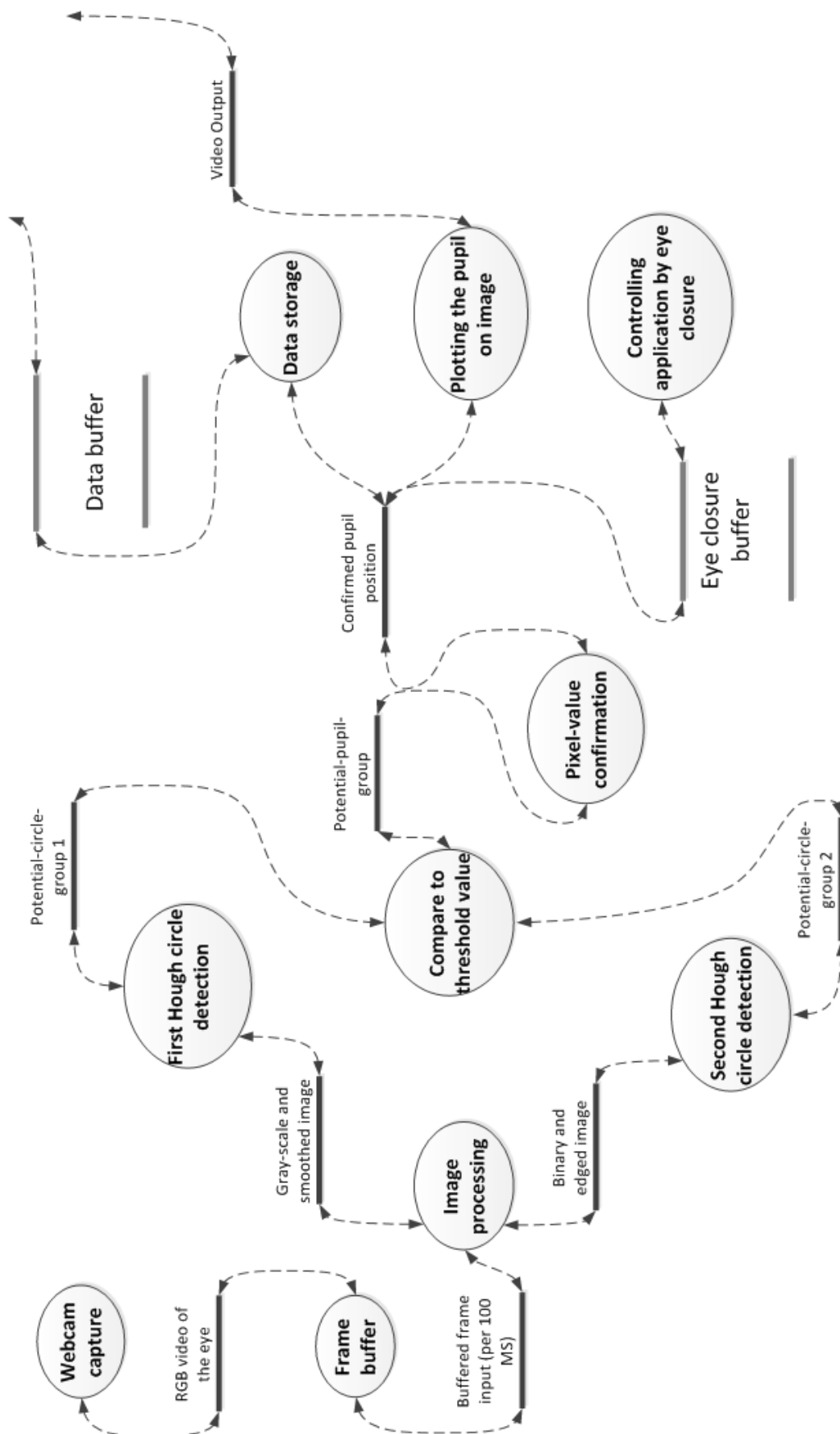


Figure 9: System software overview as state flow diagram

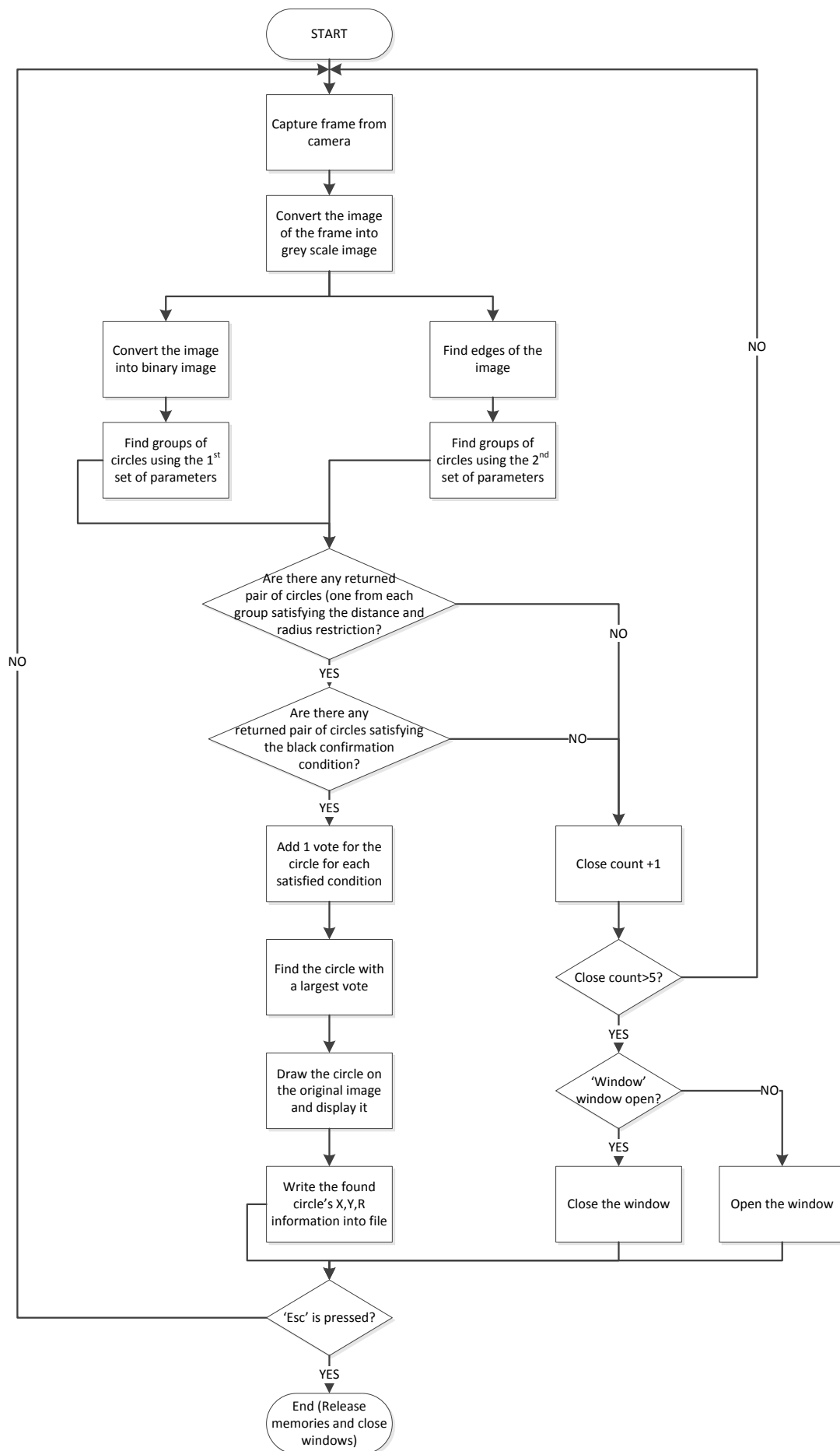


Figure 10: System software overview as flow chart

## 5. The Implementation Process/Results

### 5.1 Hardware

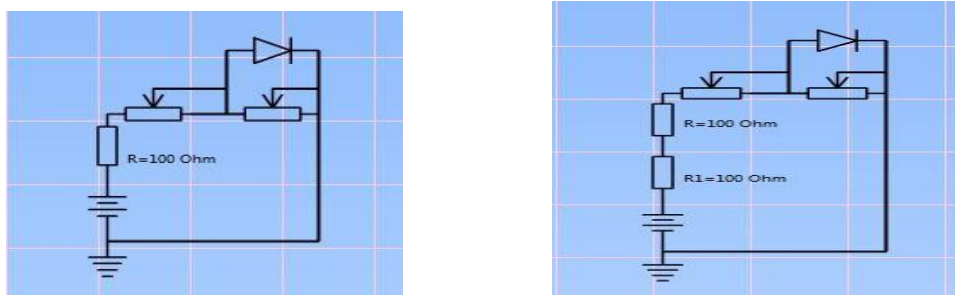
#### 5.1.1 Webcam (consideration of light level adjustment, position adjustment)

The system requires a camera that inputs natural-light-filtered images that only show the infrared light level. This is because the algorithm is based on the dark pupil image and it can be most easily obtained by infrared light. The available camera works in the visible spectrum. In order to modify the camera into an infrared one, the infrared light filter should be replaced by the visible light filter. The available visible light filter is an exposed negative film. Figure 14(a) is an image of the lens and a proper size negative film.



**Figure 11: Camera module for the system (a) Lens and negative film (b) Infrared LED and additional resistor (c) additional black flannel**

It is also necessary to change the normal LED in the camera into infrared LED to provide an infrared light source. The infrared LED works under a different voltage with the normal LEDs, namely 1.7V and 3.5V. The supply voltage for the camera is 5V from the USB lead, and the LED circuit of the camera can change the voltage across the LED from 0V to over 3.5V, and after 3.5V the voltage across the LED will drop back to 0V because it changes from the ‘off’ state to ‘on’ state. In order to replace the LEDs, it is necessary to add a division resistor in serial with the infrared LEDs. The circuit diagram of this modification is shown in Figure 15(a) the circuit for normal LEDs, which is already on the PCB of the web camera) and 15(b) (the modified circuit for infrared LEDs, with an additional division resistor in serial). The modified structure is shown in Figure 14(b). It is clearly seen that the height of the LEDs is changed because of the added resistor.



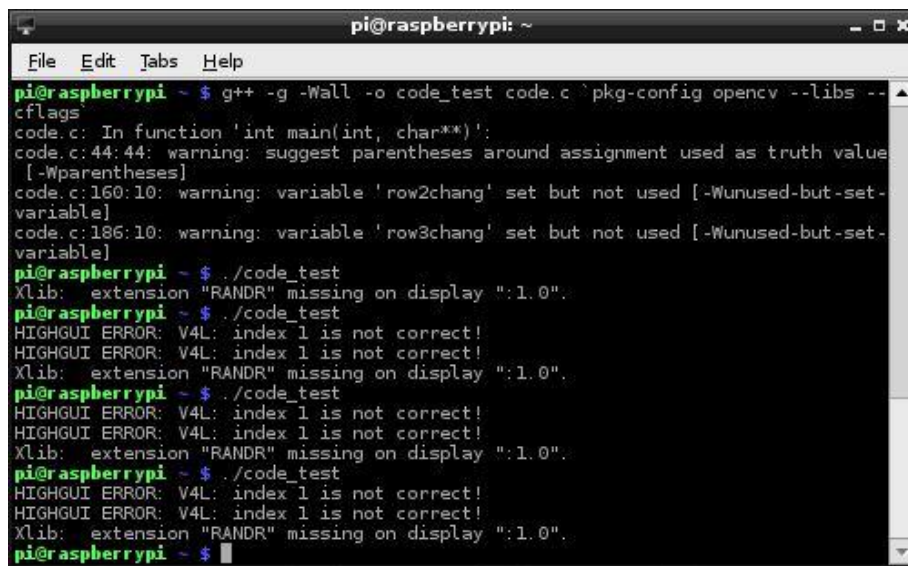
**Figure 12: Circuit adjustment for the system (a) original circuit of normal LEDs (b) adjusted circuit of infrared LEDs**

Because of the changed height of the LEDs, the separate lens and the sensors on the PCB board now have an additional gap between them. This gap should be covered by light-absorbing materials because if it is uncovered, the infrared light will cast on the sensor, disabling the sensor to receive the light coming from the lens. A piece of black, thick flannel is used to cover the gap between the lens and the sensors, as shown in Figure 14(c).

### 5.1.2 Communication channel between processor, display and controlled device

The camera's communication with the processor is through USB without alternatives, but the processor and the display and the controlled device can have more than one way of communication. The Raspberry Pi has ports for HDMI, Ethernet, USB, GPIO (general purpose input/output) communication. For the purpose of reducing cost, the Ethernet communication with a PC is selected for project development. The benefit of using the Ethernet is that the display can be removed once the system is complete and ready to work without supervision. The control is achieved either within the Raspberry Pi for an internal control or by communication using USB, GPIO or WIFI module (if applicable).

The Ethernet communication between the processor and the display requires a local IP address allocator and communication servers in both terminals. The IP address allocator is a simple router, which can also be removed once the development is complete. The suitable server for the communication is the SSH server<sup>25</sup>, or the tightVNC server. The former can enable the PC to communicate with the processor using a command line, in a window shown as Figure 16.



```

pi@raspberrypi ~ $ g++ -g -Wall -o code_test code.c `pkg-config opencv --libs --cflags`
code.c: In function 'int main(int, char**)':
code.c:44:44: warning: suggest parentheses around assignment used as truth value [-Wparentheses]
code.c:160:10: warning: variable 'row2chang' set but not used [-Wunused-but-set-variable]
code.c:186:10: warning: variable 'row3chang' set but not used [-Wunused-but-set-variable]
pi@raspberrypi ~ $ ./code_test
Xlib: extension "RANDR" missing on display ":1.0".
pi@raspberrypi ~ $ ./code_test
HIGHGUI ERROR: V4L: index 1 is not correct!
HIGHGUI ERROR: V4L: index 1 is not correct!
Xlib: extension "RANDR" missing on display ":1.0".
pi@raspberrypi ~ $ ./code_test
HIGHGUI ERROR: V4L: index 1 is not correct!
HIGHGUI ERROR: V4L: index 1 is not correct!
Xlib: extension "RANDR" missing on display ":1.0".
pi@raspberrypi ~ $ ./code_test
HIGHGUI ERROR: V4L: index 1 is not correct!
HIGHGUI ERROR: V4L: index 1 is not correct!
Xlib: extension "RANDR" missing on display ":1.0".
pi@raspberrypi ~ $

```

Figure 13: The Putty server on PC

The tightVNC server is described as "...a free software package that shows the desktop of a remote PC and allows to control it with local mouse and keyboard, just like it was a local computer."<sup>26</sup> This server acts as the display for the system. It can show the desktop of the Raspberry Pi and control it with the PC's input.

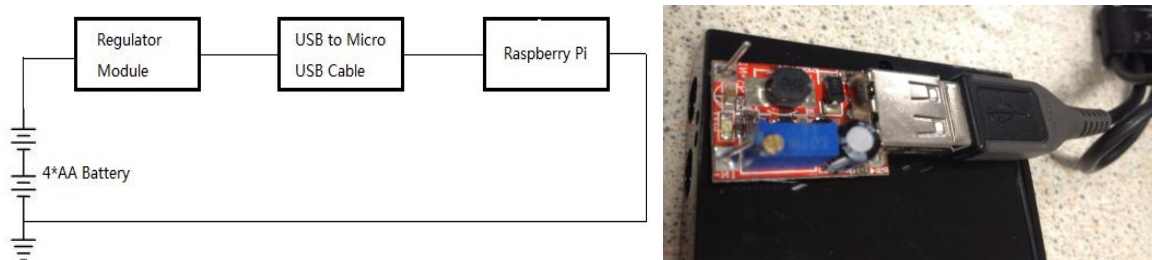
The installation of this set of communication is by: first setting up the Raspbian system on Raspberry Pi. This is done by using a Win 32 disk imager to write the Raspbian system into the SD card, and initialize the system with a keyboard and a HDMI display. After initialization the processor is ready for communication using Ethernet and SSH server. The second step is to use a IP scanner to search for the processor's allocated IP address, and use the SSH server Putty to open the Raspberry Pi's command line and install the tightVNC terminal server on Raspberry Pi, and also install the terminal server on PC. After installation and password setting the PC can access to the desktop of Raspberry Pi for displaying and controlling. The code for installing tightVNC server is

```
$ sudo apt-get install tightvncserver
$ tightvncserver
$ vncserver :0 -geometry 1920x1080 -depth 24
$ nano svnc.sh
    #!/bin/sh
    vncserver :0 -geometry 1920x1080 -depth 24 -dpi 96
$ chmod +x svnc.sh
$ ./svnc.sh
sudo bash
sudo nano /etc/init.d
chmod 755 /etc/init.d/vncboot27
```

### 5.1.3 Portable battery solution

For the selected battery module's components, the implementation will be to connect the battery case's wire with the DC-DC convertor and prepare a USB to micro USB lead for connection between the convertor and the power input of Raspberry Pi.

Figure 17 show the connection between the battery and the Raspberry Pi.



**Figure 14 Hardware structure of the battery module**

## 5.2 Software development on PC

### 5.2.1 Installation of software and compiling tools

The software and compiling tools are OpenCV library version 2.4.3 and Microsoft Visual Studio 2010 express. The library requires Cmake for installation. The detailed installation steps are as follow:

*Install Visual Studio 2010 Express on computer*  
*Extract the OpenCV package*



*Open Cmake*

*Select*

*Source code: (directory)/opencv*

*Binaries: (directory)/opencv/vs2010*

*Configure: VS10*

*Click "Generation"*

*Open "OpenCV.sln" in the generated file with Visual Studio 2010*

*Click "Build solution" for "OpenCV (61 projects)", "All\_Build", "Install" when the "Solution configure" is "Debug"*

*Click "Build solution" for the same choices when the "Solution configure" is "Release"*

*Close Visual Studio 2010 and put the (directory)/install file into the directory for the library and rename as "opencv" (C:\opencv)*

*Open "Computer System Properties -> Advanced system settings -> Environment variables -> System variables"*

*Create new variable by click "New"*

*Name:*

*OPENCV\_BUILD*

*Value:*

*(the installed directory of the library -> build) C:\opencv\build*

*Go to "Path" in the same directory and Click "edit"*

*Add "; %OPENCV\_BUILD%\x86\vc10\bin "*

*Click "OK"*

*Open Visual Studio 2010*

*Choose C++ environment, and create an empty project under "Win32 console application "*

*Open "Project manager -> debug|Win32"*

*Click "Add property sheet"*

*Under "OPENCV\_DEBUG"*

*Right click to "Properties", and go to "C/C++ -> edit additional include directories"*

*Add " \$(OPENCV\_BUILD)\include "*

*Click "OK"*

*Go to "Linker -> general -> additional library directories -> edit "*

*Add " \$(OPENCV\_BUILD)\x86\vc10\lib "*

*Go to "Input -> additional dependencies -> edit"*

*Add "*

*opencv\_core243d.lib*

*opencv\_imgproc243d.lib*

*opencv\_highgui243d.lib*

*opencv\_ml243d.lib*

*opencv\_video243d.lib*

*opencv\_features2d243d.lib*

*opencv\_calib3d243d.lib*

*opencv\_objdetect243d.lib*

*opencv\_contrib243d.lib*

*opencv\_legacy243d.lib*

*opencv\_flann243d.lib*

*,"*

*Open "Project manager -> release|Win32"*

*Do the same steps above with only changing the .lib dependencies from "....243d.lib" to "....243.lib"*

*Open "Solution explorer -> (project name) -> source files -> add new files"*

*Write code for the system*



### 5.2.2 Image processing before applying algorithm

The pre-requirement for using the chosen Hough circle algorithm is the input of a gray-scale image. In order to change the source image, which is in RGB format, into gray scale format, the function of [cvCvtColor] is used.

**Void cvCvtColor(const CvArr\* src, CvArr\* dst, int code);** in the program written as  
cvCvtColor(src\_Img, test\_Img, CV\_BGR2GRAY);<sup>28</sup>

In order for a more accurate gray scale image, in other words, to minimize the influence of noise pixels, one of a common filtering algorithm [cvSmooth] is applied to the image.

**Void HoughCircles(Mat& image, vector<Vec3f>& circles, int method, double dp, double minDist, double param1=100, double param2=100, int minRadius=0, int maxRadius=0)**<sup>29</sup>; in the program written as cvSmooth(test\_Img, test\_Img, CV\_GAUSSIAN, 3, 5);

It is worth mentioning that the Hough circle algorithm will release the storage of the input image after returning the results, so in order to support the double use of the algorithm, it is necessary to clone the image. **void\* cvClone(const void\* structPtr);**

The first Hough algorithm require a binary image with black valued 0 and white valued 255 instead of a gray scale image. The conversion between binary and gray scale can be achieved by setting a threshold between 0 and 255 and change pixels with value less than threshold to 0 and more than threshold to 255.

**double cvThreshold(const CvArr\* src, CvArr\* dst, double threshold, double maxValue, int thresholdType)**<sup>30</sup>

### 5.2.3 Testing accuracy with pictures and videos

For testing with pre-stored pictures and videos, the necessary steps is opening the pictures or the videos, capturing images from the pictures or videos, allocating memory storage to the image, and send the image into the pre-process stage and then the algorithm. The results are stored in the form of plain text into a .txt document. It is also shown on screen by opening a window showing the source image and drawing the detected circles on the image. The relevant code for the above function is:

```
/*before detection*/
FILE *data;
data=fopen("pupil.txt", "w+"); //open file to write
IplImage *src_Img = NULL; //create memory storage for image
src_Img=cvLoadImage(filename,1); //open image
cvNamedWindow("test", CV_WINDOW_AUTOSIZE); //open window for display
/*after detection*/
float * pp=(float *)cvGetSeqElem(cir,k); //cir is the returned sequence from the algorithm
CvPoint pt=cvPoint(cvRound(pp[0]),cvRound(pp[1]));
cvCircle(src_Img,pt,cvRound(pp[2]),CV_RGB(0,255,0),5); //Draw circle on the image
fprintf(data, "%f\t%f\t%f\n",pp[0],pp[1],pp[2]); //print result to file
```

### 5.2.4 Testing with real time eye movements

In the case of using the real time eye movement as the input of the system, it is necessary to rewrite the memory storage with the new image every frame, and define an exit for the program to terminate. The method of capturing the image from camera is also different from capturing from file. The relevant code for the described functions is:

```
CvCapture* pCapture = cvCreateCameraCapture(1); //the bracket's number is 0 for PC, 1 for R-Pi
cvSetCaptureProperty(pCapture,CV_CAP_PROP_FRAME_WIDTH,320); //force the size of the image
cvSetCaptureProperty(pCapture,CV_CAP_PROP_FRAME_HEIGHT,240); // into half of original value
while((src_Img = cvQueryFrame( pCapture ))) // for multiple capture of the images
{
    /* main code for detection*/
    char c=cvWaitKey(1);
    if (c==27) break; //press 'Esc' to close the program
}
```

## 5.2 Software development on R-Pi: Installation of software and compiling tools

The installation of OpenCV library version 2.4.3 and GCC compiler of the Raspberry Pi's system Raspbian is as follow:

*Open PC and Raspberry Pi*

*Open Putty and tightVNC server on PC end for remote control (R-Pi end's VNC server is set to run on startup)*

*Command on the Putty window:*

```
Sudo raspi-config
Sudo apt-get-y install build-essential cmake cmake-qt-gui pkg-config libpng12-0 libpng12-dev
libpng++-dev libpng3 libpnglite-dev libpngwriter0-dev libpngwriter0c2 zlib1g-dbg zlib1g zlib1g-dev
pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools libjpeg8 libjpeg8-dev libjpeg8-dbg libjpeg-progs
ffmpeg libavcodec-dev libavcodec52 libavformat52 libavformat-dev libgstreamer0.10-0-dbg
libgstreamer0.10-0 libgstreamer0.10-dev libxine1-ffmpeg libxine-dev libxine1-bin libunicap2
libunicap2-dev libdc1394-22-dev libdc1394-22 libdc1394-utils swig libv4l-0 libv4l-dev python-numpy
libpython2.6 python-dev python2.6-dev libgtk2.0-dev pkg-config
Wget http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.3/OpenCV-
2.4.3.tar.bz2/download
Tar- xvj OpenCV-2.4.3.tar.bz2
rm OpenCV-2.4.3.tar.bz2
cd OpenCV-2.4.3/
mkdir build
cd build
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D
BUILD_PYTHON_SUPPORT=ON -D BUILD_EXAMPLES=ON ..
make
sudo make install
sudo nano /etc/id.so.conf.d/opencv.conf
(add) "/usr/local/lib " (at the end of file)
Sudo nano /etc/bash/bashrc
(add) "PKG_CONFIG_PATH = $PKG_CONFIG_PATH : /usr/local/lib/pkgconfig export
pkg_config_path" (at the end of file)
```

## 6. The evaluation of the system's performance

The testing arrangement is adjusted after the second bench inspection to separately analyze the influences of the enhancement of the algorithm, the accuracy in eye closure, and the influence of the surrounding environment. 3 different set of testing is designed for these different aspects and other variables irrelevant to these aspects are tried to be kept minimal, though not thoroughly eliminated because of the dynamic characteristics of the eye's movement.

### 6.1 Power consumption

The power rating<sup>31</sup> for the Raspberry Pi is 700mA. The power consumption of a web camera is usually less than 500mA. The 4\*AA batteries provide an average of 4000mAh of power. The theoretical minimum working time for the system (without any controlled device attached) will be about 3.3 hours ( $4000\text{mAh}/(700\text{mA}+500\text{mA})$ ). The actual power consumption is measured by using rechargeable AA batteries providing 4000mAh for the system and leave it running until the voltage is too low for supplying the system. The power consumption measurement is repeated 3 times for average performance.

Test	Working time
1	3 hours 57 minutes
2	4 hours 02 minutes
3	3 hours 51 minutes

**Table 6: Record of system working time**

The results from the actual testing suggest that the actual power consumption is slightly smaller than the theoretical maximum value. The average working time of the system is 3 hours 3 hours 56 minutes, nearly 4 hours, and is acceptable for a portable system.

### 6.2 Weight, cost and size of the whole system

The weight, cost and size of the individual components of the system is listed below:

component	Weight (gram)	Cost (GBP)	Size (cm <sup>3</sup> )	Required in final system
Raspberry Pi	45+100(case)	25	9*6*3	YES
Battery module	90	7	5*4*2	YES
Router	70	5	10*7*2	NO
USB-micro USB lead	20	1	40 (cm)	YES
Ethernet cable*2	30*2	2	40(cm)*2	NO
Head-mounted camera	200	13	30*20*10	YES
PC	1500	>500	40*30*3	NO
Final system	450	46	6500	-

**Table 7: record of system weight and size evaluation**

The optimum size and weight can reach as small as 6500 cm<sup>3</sup> (and less than 500 cm<sup>3</sup> if the cap is not considered), and less than 450 grams

### 6.3 Outlier removal before data handling--Error rate of obvious detection mistakes

According to the above analysis of the algorithm's enhancement, there will still be existence of errors in the returned data, which may bring great influence to the further analysis of the data. In order to reduce the influence, a outlier removal section can be added to the data analysis stage after the detection stage. An outlier is defined as

*An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.*<sup>32</sup>

In this system, the outliers are defined as the group of data that is largely different with the neighboring groups. The cause of the large difference may be a quick saccade of the pupil that does not reflect any information, or a wrong detection of the pupil by the algorithm. The adopted outlier removal method is the Z-value method for numerical data. This method is based on the assumption that the detected data for a fixation in a period less than 15 seconds will tend to be in a normal distribution. As the tested eye may move in small range during recording, it is acceptable that the data changes in this small range. The calculation for a Z score is:

$$z = \frac{x - \mu}{\sigma}, \text{ Where } \mu \text{ is the mean of the population; } \sigma \text{ is the standard deviation of the group of data.}^{33}$$

The quantity Z shows the distance between the raw data and the group mean in units of the standard deviation. Z is negative when the raw data is below the mean, positive when above.

Because the evaluation is based on pupil fixation as explained before, the data exceeding 3 times of  $\sigma$  can be seen as an obvious outlier from the other components in the same group, thus removed from the data analysis. In practical applications of this system, these components should be treated as useless in gesture matching. Table 8 shows the statistics for the outlier removal section on the testing data.

Actual pupil movement: fixation without blinking					
Number of data: 150 for each environment, each participant (15 seconds, 10 fps)					
environment		P1	P2	P3	Average
Natural light	Outlier	3 (2.0%)	1 (0.7%)	2 (1.3%)	1.3%
	Useless data	5	4	7	3.5%
Dim	Outlier	1 (0.7%)	0 (0.0%)	1 (0.7%)	0.5%
	Useless data	1	0	1	0.5%
Bright	Outlier	3 (2.0%)	3 (2.0%)	0 (0.0%)	1.3%
	Useless data	3	3	1	1.5%
Dark	Outlier	4 (2.6%)	3 (2.0%)	2 (1.3%)	2.0%
	Useless data	4	3	4	2.4%
Note: Useless data is the sum of outlier and invalid detection					

**Table 8: Statistics of performance and outlier percentage**

## 6. 4 Influence from external environment—natural light, bright light, dim and dark

The environment's influence to the system is mainly analyzed by comparing the detection results of the same person and nearly the same fixation in the environment of four different environments, with main difference in light levels. The real time testing will inevitably introduce random noise from the change of focus distance error, light level shift, testing participants' movement, etc. It will be hard to distinguish the error introduced by the environment with the error caused by the system itself. The testing design of this system is based on the system's general performance under the roughly sorted environment types. The detailed performance measure is based on the percentage of useful data, and the percentage of data's variation. Considering the fact that each test has different eye-to-camera distance, the average value of the radius as well as the standard variation of the radius of the detected pupil will not be of relevance between the groups of test. The variation of the pupil may be best evaluated by the variation of radius divided by the average value, which represents a percentage of the variation. The detailed data is shown in Table 9.

<b>Actual pupil movement: fixation without blinking</b>					
<b>Number of data: 150 for each environment, each participant (15 seconds, 10 fps)</b>					
<b>environment</b>		P1	P2	P3	Average
<b>Natural light</b>	$\sigma$ (standard deviation)	3.418086	6.9242	4.56475	-
	$\mu$ (mean value)	44.5477	52.47857	71.1126	-
	$\sigma/\mu$ (%)	7.7	13.2	6.4	7.3
	Useful data (%)	96.7	97.3	95.3	96.4
<b>Dim</b>	$\sigma$ (standard deviation)	1.061	3.546898	2.845784	-
	$\mu$ (mean value)	51.210	67.31269	65.9242	-
	$\sigma/\mu$ (%)	2.1	5.3	4.3	3.9
	Useful data (%)	99.3	100	99.3	99.6
<b>Bright</b>	$\sigma$ (standard deviation)	1.594	4.79502	3.72467	-
	$\mu$ (mean value)	39.147	54.42426	63.9448	-
	$\sigma/\mu$ (%)	4.1	8.8	5.8	6.23
	Useful data (%)	98	98	99.3	98.4
<b>Dark</b>	$\sigma$ (standard deviation)	3.223	2.629957	5.450828	-
	$\mu$ (mean value)	46.147	65.21503	67.06713	-
	$\sigma/\mu$ (%)	7.0	4.0	8.1	6.37
	Useful data (%)	97.4	98	97.4	97.6

**Table 9: Statistics of system performance and variation of data**

The testing result suggests that under different environments, the system's performance will vary slightly. The system works comparatively better under a dim light environment, with the variation percentage of 3.9% and useful data percentage of 99.6%. The system underperforms when the environment is less stable (i.e. the natural light environment), with the variation percentage of 7.3% and useful data percentage of 96.4%. As a conclusion, the system is able to detect the pupil with an average accuracy of 98.0%. Figures and charts in appendix show more details of the groups of data.

## 6.5 Changes brought by improving the algorithm

### 6.5.1 Accuracy tested by 20 preset images of the eye

The improvement of the algorithm has stages of single usage of algorithm, pixel counting, double usage of algorithm, speed enhancement. The accuracy of the performance is tested by 20 preset images with a random position of the pupil. Figure 18 shows two of the random pupil position images.



**Figure 15: Sample image for accuracy testing**

The result is evaluated by the following standard:

- The center of the detected circle is within the actual pupil area
- The radius of the detected circle is not larger or smaller than 50% of the actual pupil

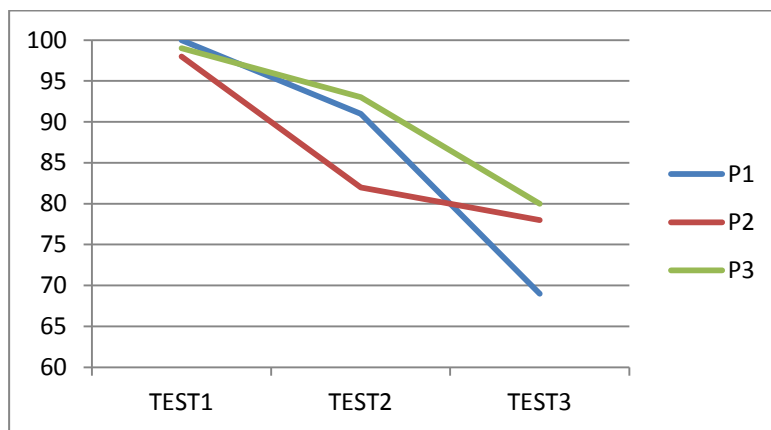
The detailed result of the test is show in Table 10

	Single Hough	Double Hough	Pixel counting	Speed raise
Picture 1	Y	Y	Y	Y
Picture 2	Y	Y	Y	Y
Picture 3	Y	Y	Y	Y
Picture 4		Y	Y	Y
Picture 5	Y	Y	Y	Y
Picture 6			Y	Y
Picture 7		Y	Y	Y
Picture 8	Y	Y	Y	Y
Picture 9	Y	Y	Y	Y
Picture 10		Y	Y	Y
Picture 11	Y	Y	Y	Y
Picture 12	Y	Y	Y	Y
Picture 13	Y	Y	Y	Y
Picture 14				
Picture 15		Y	Y	Y
Picture 16	Y	Y	Y	Y
Picture 17	Y	Y	Y	Y
Picture 18	Y	Y	Y	Y
Picture 19	Y		Y	Y
Picture 20	Y	Y	Y	Y
<b>Notes:</b>				
1.the improvements of the system follow the order of “from left to right” and cannot be ignored when testing for the accuracy.				
2. the evaluation is based on the system of Raspberry Pi				

**Table 10: Accuracy testing results**

The main reason of the early versions of the system cannot detect some of the pupils is because it is set to take the circle largest vote as the pupil. From the testing result, the number of correctly detected picture raise from 14 in the Single use of algorithm stage, to 17 in the Double use of algorithm stage, and to 19 in the Pixel counting stage. It is seen that by reducing the size of the output image does not influence the accuracy of the system, by having 19 out of 20 pupils detected, the same as in the Pixel counting stage. The real time measurement of the accuracy is hard to evaluate because it is difficult to analyze every frame's accuracy by human eye. The method of building general assumptions based on changing variables and matching the actual performance with the assumption is used.

The main assumption made is that the system will detect more pupils in a same duration if the participant keeps the eye open, and will detect fewer pupils if the participant blinks more. Figure 19 show the actual record of data from the three participants. Test 1 for each participant is required to keep the eye fixed without blinking for 10 seconds; test 2 requires casual blinking during the recording; test 3 requires deliberately frequent blinking during the recording. According to the requirements, each test has 100 groups of data. Figure 19 presents the number of groups with detected pupil from the data.



**Figure 16: Detected useful data percentage with different input**

According to the data, the results of the actual performance are consistent with the assumption and thus the system's accuracy in real time detection is concluded as acceptable.

Further for the closure measuring, 3 groups of the video are recorded for eye closure testing, with the participants required to close their eyes regularly (normally once every 5 seconds) and the test lasts for 20 seconds. The following statistics are gathered during this test:

Group 1: The participant's eyes close at the time  $t = 7, 10, 13$  (seconds), each time lasting for about 1 seconds. The detected closure is in data group (80-84), (120-127), (158-168).

Group 2: The participant's eyes close at the time  $t = 5, 11, 17$  (seconds), each time lasting for about 1 seconds. The detected closure is in data group (61-67), (135-144), (206-211).

Group 3: The participant's eyes close at the time  $t = 3, 8, 14, 19$  (seconds), each time lasting for about 1 seconds. The detected closure is in data group (37-44), (98-104), (170-180), (231-2239).

The system's running speed as tested by stop watch is about 12 frames per second.

The total groups of data for the 20 seconds vary from 240 to 250, as it is counted by stopwatch controlled by hand. The counting results show that the closure detection is generally consistent to the observation directly to the participant's eyes. The duration and the detection time is observed to be accurate.

### 6.5.2 Speed test (r-pi's specific algorithm speed, changing image size's speed)

The system's working speed is detected by recording the running time with a stopwatch for detecting 150 frames of picture. As the system is designed for pupil diameter monitoring and pupil gesture controlling, the working speed should be larger than 5 frames per second so that the movements of the pupil can be correctly recorded without missing most of the movements. Table 11 shows the recording of system running time with different output image resolution, and on different platforms.

Output resolution	Running time on PC	Running time on Raspberry Pi
<b>640*480 test1</b>	22 seconds	Cannot display correctly
<b>320*240 test1</b>	8 seconds	24 seconds
<b>320*240 test1</b>	9 seconds	24 seconds
<b>320*240 test1</b>	8 seconds	23 seconds
<b>160*120 test1</b>	6 seconds	14 seconds
<b>160*120 test2</b>	6 seconds	15 seconds
<b>160*120 test3</b>	6 seconds	15 seconds
<b>Notes: the time recorded is for processing 150 frames of picture, without any preset frame rate from the system (i.e. the system's delay is set to 1ms)</b>		

**Table 11: System speed with different output resolutions**

Based on the above recording, the suitable outputting resolution for PC is 320\*240, with a processing rate of about 17 frames per second; and the suitable outputting resolution for Raspberry Pi is 160\*120, with a processing rate of about 10 frames per second.

## 6.6 Difference between PC and R-Pi

### 6.6.1 Detecting accuracy for eye closure

For the comparison of the accuracy between the two platforms, the final system with the speed raise, double use of algorithm, and the pixel counting sections is tested with a human eye observation for performances.

The observation results suggest that the system performance on PC platform is better than on Raspberry Pi, with less delay from the eye's action to the system's reaction, but the accuracy of the system has little difference between the two platforms. During the 20 seconds of real time observation, the participant is required to regularly close the eye. There will be around 10 times of closure in 20 seconds, and according to observation, the system running on both PC and Raspberry Pi successfully detected all of the closures without obvious delay.



### 6.6.2 Detecting speed

The same method for eye closure detecting accuracy is used in this section. The evaluation is based on human eye observation and can only describe qualitative performances. The participant is required to look at different directions or close the eye regularly during the test, and around 10 different pupil changes happened during the observation. According to observation, the system running on PC successfully detected all of the pupils in different position without obvious delay. This means that the system can detect the movement accurately and re-detect the pupil quickly after the eye changes the gaze direction. The system running on Raspberry Pi also detected all of the positions successfully, however, it will need a human eye observable time to react and re-detect the pupil when the eye opens from a closure. The duration of this observable time is less than 1 second.

## 6.7 Cognitive workload testing for the system

A cognitive workload test for the final system implemented on Raspberry PI is held for a general performance observation. The test is accomplished by one participant doing activities of plain text reading and website looking.

The participants are required to read a randomly selected plain text, or look at a randomly selected website for 15 seconds, during which the data of the eye is collected by the device. 2 participants' data are collected for the analysis. Figure 20 shows the result of the test.

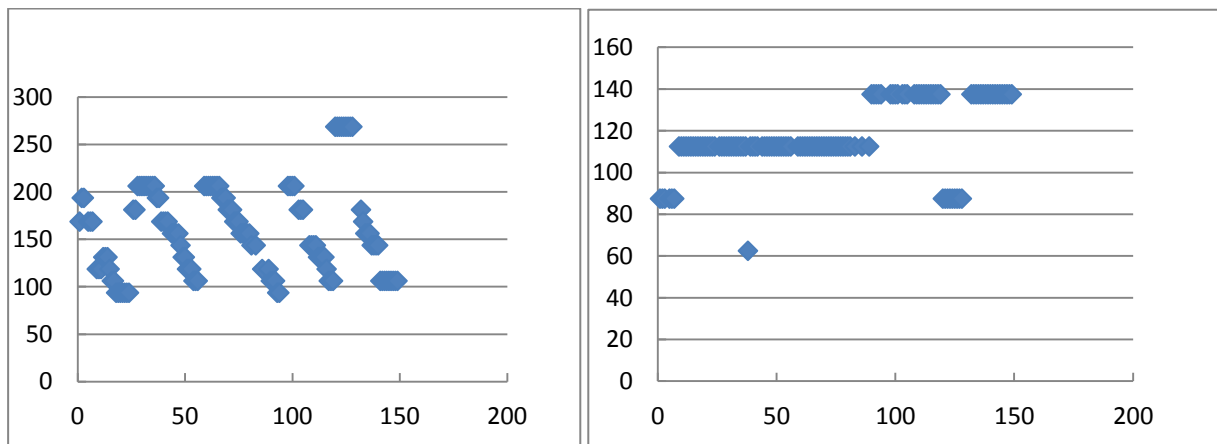


Figure 17: X-Y position recording of reading text

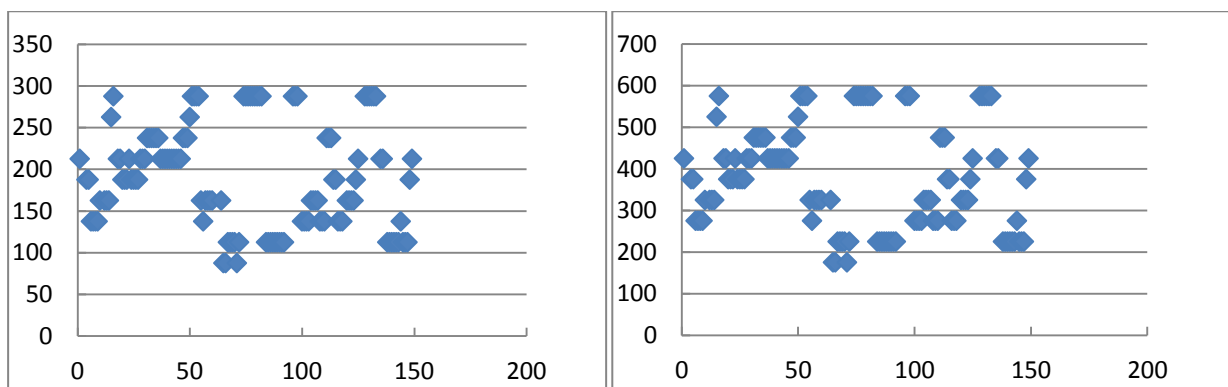


Figure 18: X-Y position recording of looking at a website

This test is held under a stable low light level environment and the participant's head kept still. The potential variables are therefore kept minimal for analyzing the difference between these two activities. Based on the result of the test, the text reading activity has a more regular trend of data change while in the activity of looking at a website, the eye will conduct more sudden moves that might be a saccade without any information acquired.

## 7. Conclusion

This report introduced the design, development and improvement of a portable pupil tracking system that can be further used in the applications such as pupil diameter monitoring, gaze tracking, and eye gesture controlling. The system is based on the Linux micro computer Raspberry Pi that is capable of combining the buffering images and data, processing images, and outputting functions into the card-sized computer. Current stage of accomplishment is able to detect pupil parameters and eye closures. Once the gesture matching section for controlling peripheral devices is added to the system, it can also be used for an eye gesture controlling application.

There is existing well established pupil tracking systems based on computers (ITU gaze tracker) but the portability and suitability for using the system without PC support and in the outdoor environment is limited because of the system is already packaged and simplified for specific systems and applications. The system introduced in this report can work independently without the control of the computer, as the computer only acts as the display and debugging tool during the development of the system.

The algorithm for this system is adopted from the Hough circle in the OpenCV library, and through the improvement of double using Hough circle detection, pixel counting for the circle, and raising speed, the returned data from the algorithm is in the format of time buffered array into .txt file and can be easily altered into a FIFO (first-in-first-out) or FILO (first-in-last-out) buffer for further applications. The system also records the input video with the detected area marked with circles for later monitoring. The speed raise for the Raspberry Pi computer ensures the system works with a frame rate of 10 and thus suitable for applications requiring a robust response from the action to the reaction. The efforts are all made for a optimum accuracy and speed for a low cost and multiple-application compatible system.

The evaluations of the system are focused on the accuracy of closure and pupil position, as well as the system's response speed to inputs. The results from the testing data evaluation suggest the system can deliver convincing detection of the pupil with a satisfying response speed. The currently completed system can detect the pupil diameter for applications such as cognitive workload monitoring in different environments, both indoor and outdoor. The further developments for other applications can also use the evaluation results in this report as the source of analyzing the change in accuracy and speed for the system.

## 8. Future work

The system is designed to work without PC control and without displaying the output. The program is set to run at startup of the Raspberry Pi system. It is vulnerable because of the reason that the focus of the currently adopted web camera needs hand adjustment for a focus at the eye. It is possible that the focus will not be correctly set if not monitoring from a display is added to the system. One of the future works will be solving this problem. It can be solved by adding a WIFI module to the system for a communication with the PC, or by adding a low cost LCD screen in the system. Also changing the adopted model of camera with one that has auto-focus-changing function will perfectly solve this potential problem.

The second future development lies on the speed issue of the Raspberry Pi platform. As the evaluation of the system's performance with real time eye movement suggested, the current system has a human observable delay on the Raspberry Pi platform. As the bottle neck of the speed issue is the image outputting on screen, this problem may be solved by designing a module to close the display without stopping the program. The first aspect of future development may eliminate this delay if the LCD display has a higher data communicating rate than the current Ethernet transmission with tightVNC server.

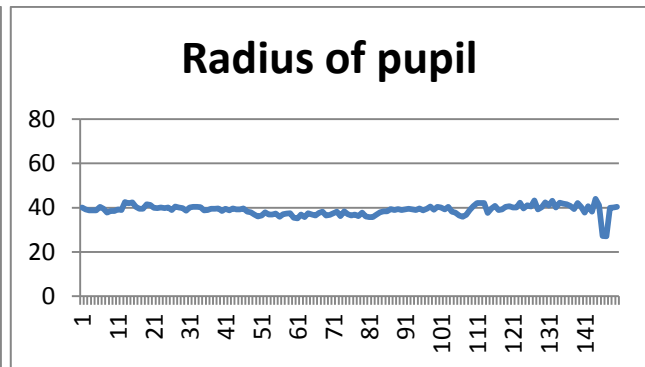
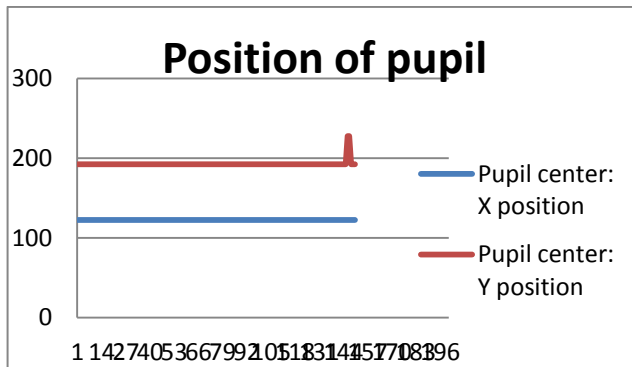
The third aspect for the future work is building a more application friendly output format, that enable this system to work with other systems for applications such as gaze analyzing, controlling with gaze, or pupil behavior identifying. The current output format is .txt file with arrays of pupil diameters. A better solution may be a FIFO buffer of certain time duration's pupil diameters.

The fourth aspect of potential future development is the exploration of using the current system for an eye gesture controlling application. As the Raspberry Pi has multiple types of I/O ports for communication, this system has the potential of being the core control module for other systems, for example, a mobile robot system, or a website surfing application controlled by gaze.

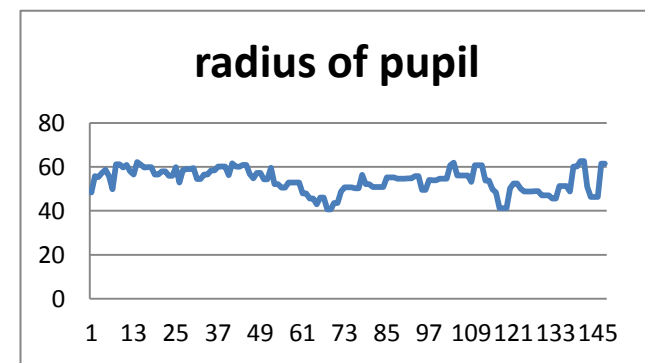
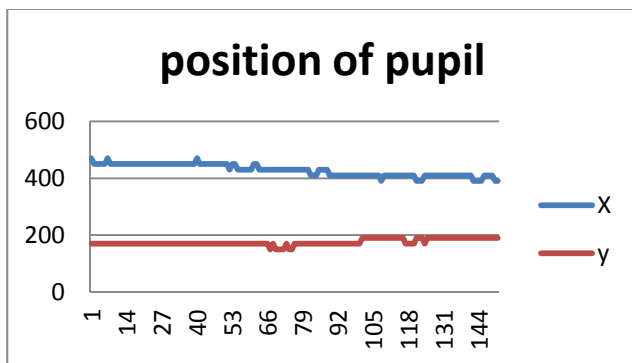
## Appendix A: Test data

### B1: Fixation under bright light level

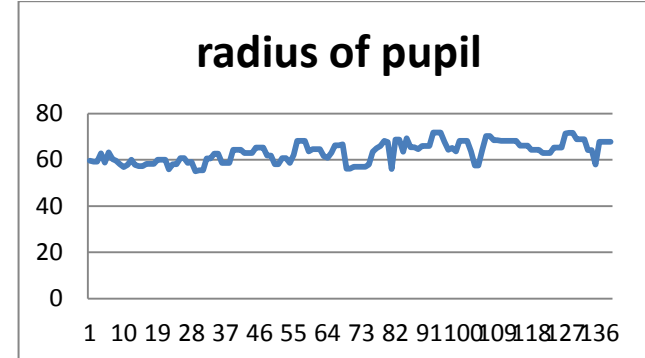
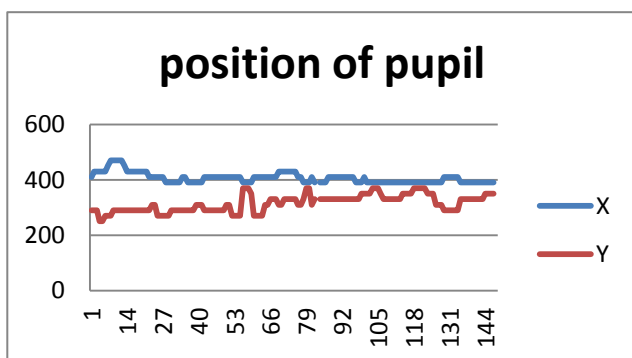
#### Participant 1



#### Participant 2

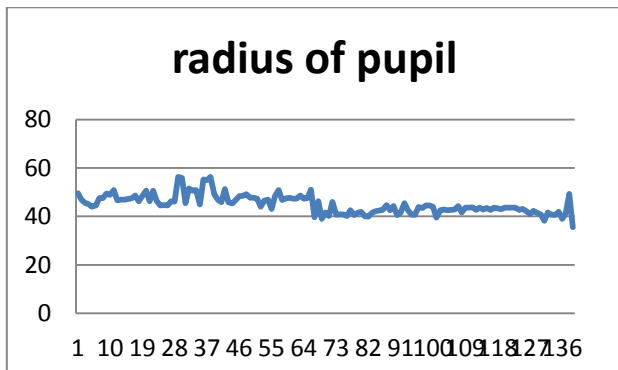
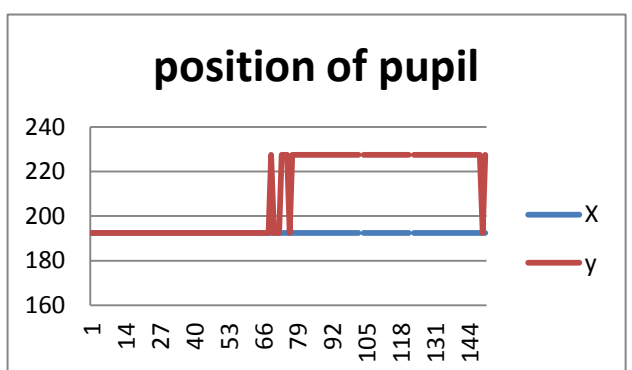


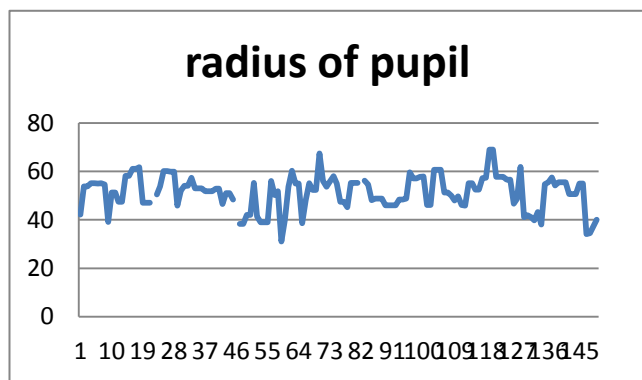
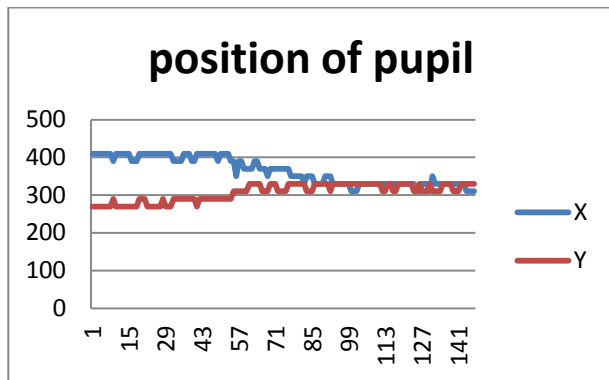
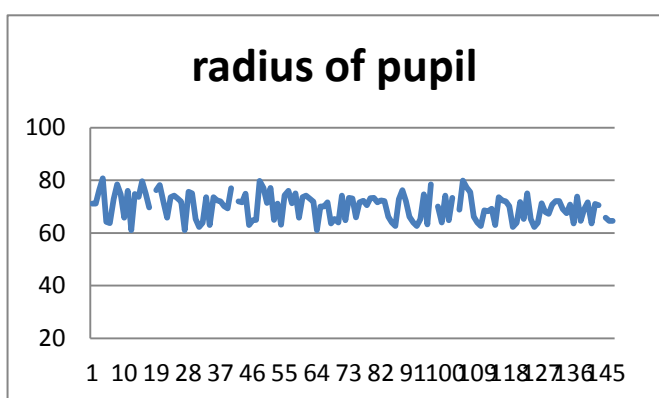
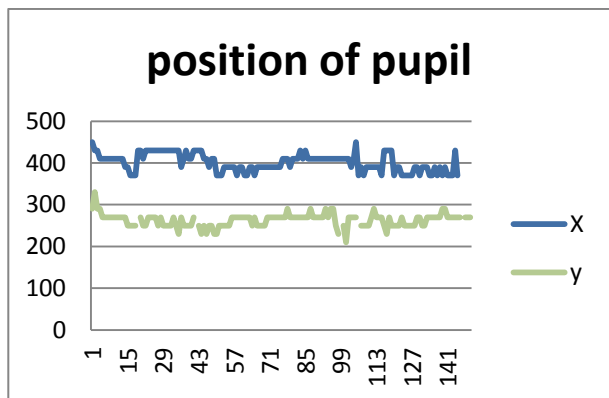
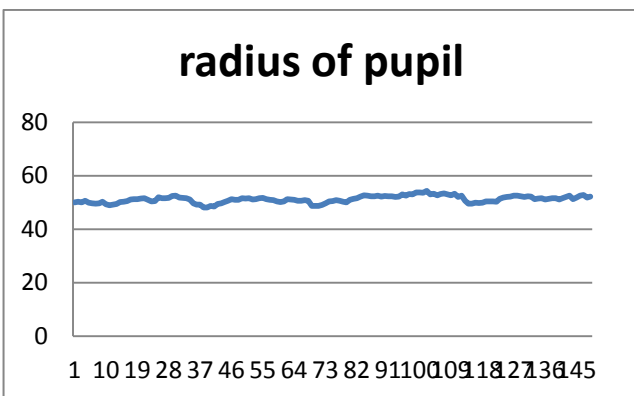
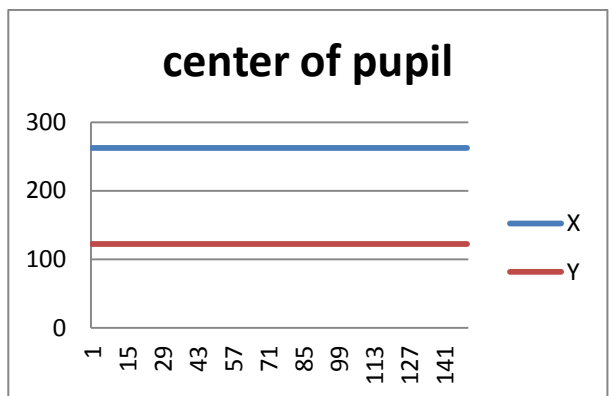
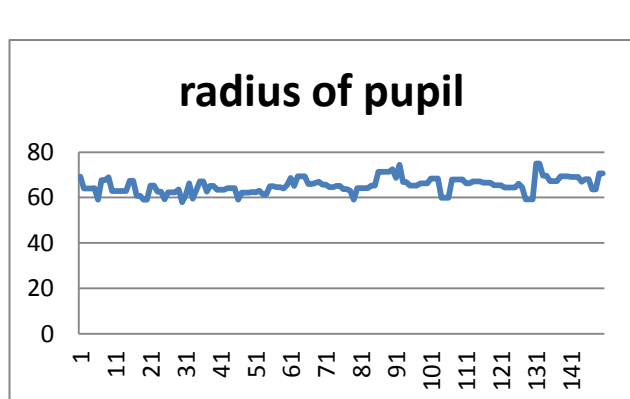
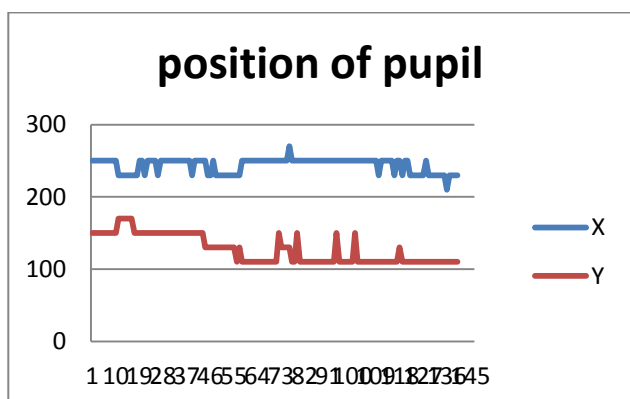
#### Participant 3

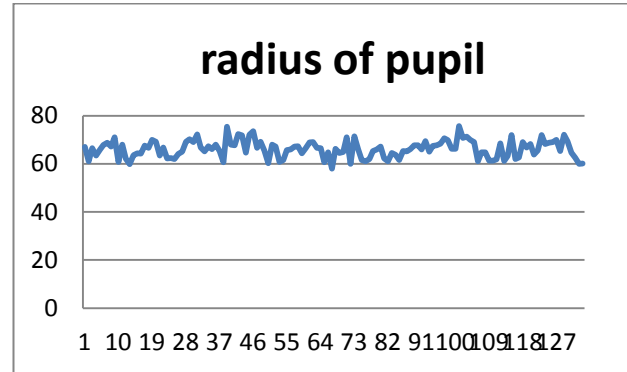
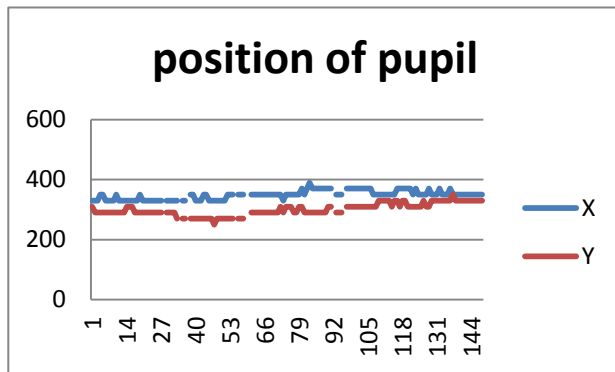
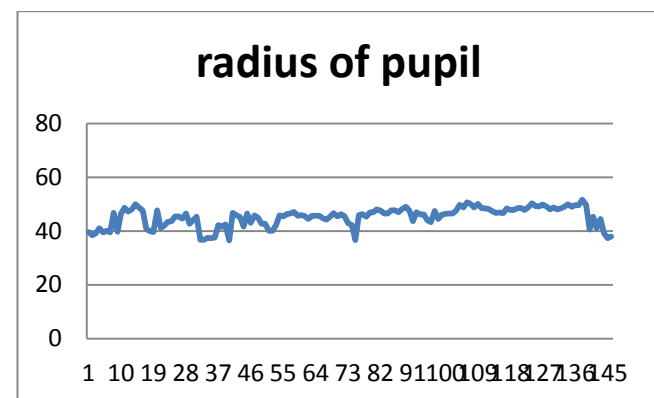
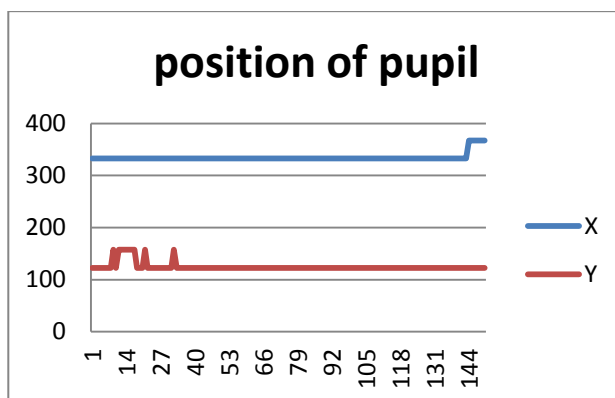
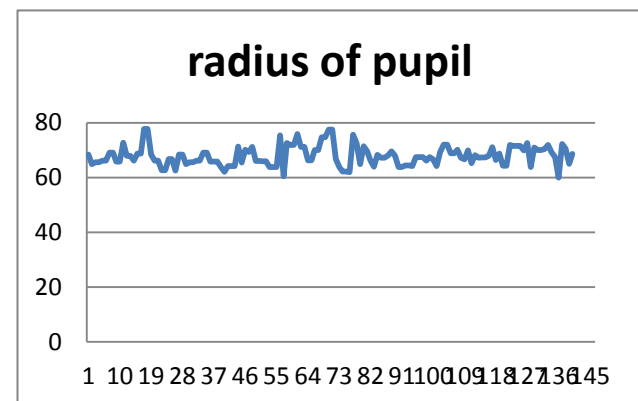
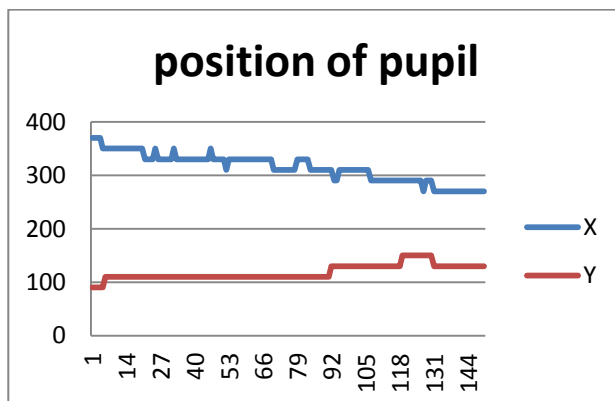
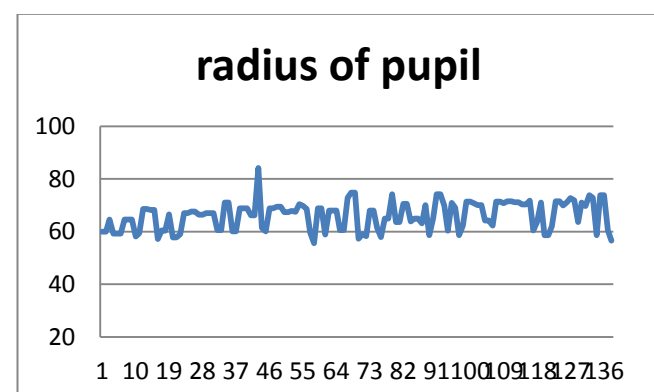
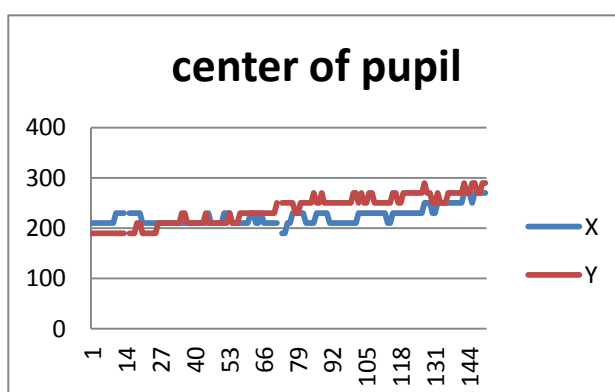


### B2: Fixation under natural light

#### Participant 1



**Participant 2****Participant 3****B3: Fixation under dim light****Participant 1****Participant 2**

**Participant 3****B4: Fixation under dark light****Participant 1****Participant 2****Participant 3**

## Appedix B: statements

### B1: Safety Statements

Written risk assessment form had been filled in before the project began. All the hardware work in this project is done according to the control measures required in the form. For the potential user of the system, to prevent strangulation, all of the electricity wires are fixed inside the shell of the camera. The length of flexible wire is carefully controlled under the length which is enough to surround anyone's neck. All sharp parts on the system are covered by soft material to prevent penetration. No voltage higher then 5V is on the system, so it does not have an electric shocks danger.

The maximum permissible exposure (MPE)<sup>6</sup> of 880 nm infrared light is 196mW/cm . The illumination diameter of the eye is 3cm approximately, according to the infrared LED datasheet, the infrared LED works at 200mW. Thus,

$$S = \pi * R^2 = \pi * \left(\frac{3}{2}\right)^2 \approx 7 \text{ cm}^2$$

$$E = 196\text{mW} / 7\text{cm}^2 \approx 28 \text{ mW/cm}^2 < \text{MPE}$$

So the IR LED will not have thermal damage to the eye.

### B2: Statements about ethic

Written risk assessment form had been filled in before any experiment began. All the volunteer experiments are done under the guideline of the form.

Three volunteers are picked to do the experiment. All of them are students in University of Birmingham. Their experiment data is kept in my computer anonymously and will be destroyed when the project is finished finally. No data call-back is required from any volunteers till the day this report is finished.

## Reference

- <sup>1</sup> Tobii UX Live Solution Makes Real-time Tracking Easy, Affordable and a New Must-have Tool for Web Designers and Developers  
[http://www.mynewsdesk.com/us/pressroom/tobii\\_technology/pressrelease/view/tobii-ux-live-solution-makes-real-time-eye-tracking-easy-affordable-and-a-new-must-have-tool-for-web-designers-and-developers-854808?utm\\_source=rss&utm\\_medium=rss&utm\\_campaign=Subscription&utm\\_content=pressrelease](http://www.mynewsdesk.com/us/pressroom/tobii_technology/pressrelease/view/tobii-ux-live-solution-makes-real-time-eye-tracking-easy-affordable-and-a-new-must-have-tool-for-web-designers-and-developers-854808?utm_source=rss&utm_medium=rss&utm_campaign=Subscription&utm_content=pressrelease)
- <sup>2</sup> The Eye Tribe- Eye control for mobile devices  
<http://theeyetribe.com/>
- <sup>3</sup> A. Liaghatdar, K. Kangarloo, F. Farokhi, 2011 "Pupil Localizing in Video images the First Step Toward Eye Monitoring" International Conference on Multimedia Technology (ICMT) pp: 3163-3166.
- <sup>4</sup> M. Asadifard, J. Shabanzadeh, 2010 " Automatic Adaptive Center of Pupil Detection Using Face Detection and CDF Analysis " Proceedings of the International MultiConference of Engineers and Computer Scientists , Vol I.
- <sup>5</sup> S. Panev, O. Bombarov, S. Sokolov, 2008 " IR Based Pupil Tracking Using Particle Filtering" International Scientific Conference Computer Science 2008
- <sup>6</sup> O. V. Komogortsev, J. I. Khan, 2007 "Kalman Filtering in the Design of Eye-Gaze-Guided Computer Interfaces" J. Jacko (Ed.): Human- Computer Interaction, Part III, HCII 2007, NCS 4552, pp. 679 689, 2007. Springer-Verlag Berlin Heidelberg
- <sup>7</sup> C. Jian-nan, Z. Chuang, Q. Yan-jun, L. Ying, Y. Li, 2011 "Pupil Tracking Method Based on Particle Filtering in Gaze Tracking System" International Journal of the Physical Sciences Vol. 6(5), pp. 1233-1243
- <sup>8</sup> Z. Zhu, Q. Ji, K. Fujimura, 2002 "Combining Kalman Filtering and Mean Shift for Real Time Eye Tracking Under Active IR Illumination" IEEE , 1051-4651/02
- <sup>9</sup> Q. Ji, X. Yang, 2002 "Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance" Real-Time Imaging 8, 357 377
- <sup>10</sup> [http://en.wikipedia.org/wiki/Eye\\_tracking](http://en.wikipedia.org/wiki/Eye_tracking)
- <sup>11</sup> Xuan Zhang, I. Scott MacKenzie, Evaluating Eye Tracking with ISO 9241 – Part 9, York University
- <sup>12</sup> E.C. Lee, K.R. Park, A robust eye gaze tracking method based on virtual eyeball model, Machine Vision and Applications 20 (5)(2009) 319–337
- <sup>13</sup> Ji & Zhu, 2004, , Batista 2005, Ji & Yang, 2002, Zhu & Ji, 2005, Li et al., 2004, Morimoto et al., 2000, Haro et al., 2000
- <sup>14</sup> L. Young and D. Sheena. Methods & designs: Survey of eye movement recording methods. Behavioral Research Methods & Instrumentation, 7(5):397-429, 1975.
- <sup>15</sup> Batista, 2005, Ohno & Mukawa, 2004, Ji & Bebis, 1999, Ji & Yang, 2002
- <sup>16</sup> :<http://www.samsung.com/global/business/semiconductor/product/application/detail?iald=836&productid=7125>
- <sup>17</sup> Starburst: A robust algorithm for video-based eye tracking Dongheng Li, Derrick J. Parkhurst Human Computer Interaction Program, Iowa State University, Ames, Iowa, 50011



<sup>18</sup> <http://thirtysixthspan.com/openEyes/starburst.pdf>

<sup>19</sup> L. Swirski et al, Robust real-time pupil racking in highly off-axis images

Meunier, F. (2009) On the Automatic Implementation of the Eye Involuntary Reflexes Measurements Involved in the Detection of Human Liveness and Impaired Faculties.

<sup>20</sup> The look-up table that defines the relationship between the boundary coordinates, orientation, and the Hough parameters.

<sup>21</sup> The structure CvSeq is a base for all of OpenCV dynamic data structures. It is grow-able. The basic structure of CvSeq is:

```
{ int flags; int header_size; CvSeq* h_prev; CvSeq* h_next; CvSeq* v_prev; CvSeq* v_next; int total;
int elem_size; CvMemStorage* storage; CvSeqBlock* first}.
```

[http://docs.opencv.org/modules/core/doc/dynamic\\_structures.html#cvseq](http://docs.opencv.org/modules/core/doc/dynamic_structures.html#cvseq)

<sup>22</sup> OpenCV documentation

[http://opencv.willowgarage.com/documentation/cpp/imgproc\\_feature\\_detection.html](http://opencv.willowgarage.com/documentation/cpp/imgproc_feature_detection.html)

<sup>23</sup> the final position of a point ( $\mathbf{R}_f$ ) relative to its initial position ( $\mathbf{R}_i$ ), and a displacement vector can be mathematically defined as the difference between the final and initial position vectors:  $\mathbf{s} = \mathbf{R}_f - \mathbf{R}_i = \Delta \mathbf{R}$ . From Wikipedia

<sup>24</sup> OpenCV documentation

[http://opencv.willowgarage.com/documentation/c/miscellaneous\\_image\\_transformations.html](http://opencv.willowgarage.com/documentation/c/miscellaneous_image_transformations.html)

<sup>25</sup> PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers

<http://www.putty.org/>

<sup>26</sup> <http://www.tightvnc.com/release-2.6.php>

<sup>27</sup> [http://elinux.org/RPi\\_VNC\\_Server](http://elinux.org/RPi_VNC_Server)

<sup>28</sup> OpenCV documentation

[http://opencv.willowgarage.com/documentation/c/miscellaneous\\_image\\_transformations.html](http://opencv.willowgarage.com/documentation/c/miscellaneous_image_transformations.html)

<sup>29</sup> OpenCV documentation

[http://opencv.willowgarage.com/documentation/c/image\\_filtering.html](http://opencv.willowgarage.com/documentation/c/image_filtering.html)

<sup>30</sup> OpenCV documentation

[http://opencv.willowgarage.com/documentation/c/miscellaneous\\_image\\_transformations.html#threshold](http://opencv.willowgarage.com/documentation/c/miscellaneous_image_transformations.html#threshold)

<sup>31</sup> In electrical and electronic engineering, the power rating of a device is a guideline set by the manufacturer as a maximum power to be used with that device. This limit is usually set somewhat lower than the level where the device will be damaged, to allow a margin of safety. [http://en.wikipedia.org/wiki/Power\\_rating](http://en.wikipedia.org/wiki/Power_rating)

<sup>32</sup> Grubbs, F. E.: 1969, Procedures for detecting outlying observations in samples. *Technometrics* 11, 1–21.

<sup>33</sup> [http://en.wikipedia.org/wiki/Standard\\_score](http://en.wikipedia.org/wiki/Standard_score)