DESIGN FOR AN INEXPENSIVE SOFTWARE SOLUTION TO EYE-GAZE TRACKING AS AN ALTERNATIVE COMPUTER INPUT DEVICE

by

MAYNARD EXUM

A thesis submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Electrical & Computer Engineering

Written under the direction of

Dr. Evangelia Micheli-Tzanakou & Dr. Manish Parashar

And approved by

_____

_____

_____

New Brunswick, New Jersey

January, 2008

ABSTRACT OF THE THESIS

Design for an Inexpensive Software Solution to Eye-Gaze Tracking as an Alternative Computer Input Device

By MAYNARD EXUM

Thesis Directors:
Dr. Evangelia Micheli-Tzanakou
Dr. Manish Parashar

Eye tracking has been a topic of interest in a number of different fields for many years. This thesis explores a few of the many eye-gaze tracking techniques that are currently available and also takes a broad look at the problems that plague most, if not all, of the solutions in the field. There are a few basic shortcomings in most existing eye trackers. In particular, the majority of eye trackers on the market are either too expensive, too complicated (for the average consumer) to use, or uncomfortable for the user; and frequently, tracking systems suffer from more than one of these design flaws. In addition to these issues, almost all of the systems surveyed in this thesis neglect to incorporate a solution that addresses the need to turn the tracking feature off when the user no longer wishes to control the system. The system implemented and proposed in this thesis was designed with the intention of managing each of these issues. By using an ambient light process with a feature-based algorithm, the solution promises to keep the overall cost of the system low and to make it comfortable for the user. In addition, the system uses a remote camera location and a dwell time selection method to ensure that it feels natural to the user when in use. Finally, the system incorporates a "stateful" tracking design to turn tracking on and off, in an effort to mitigate the "Midas Touch" problem which has

probably done most to impede the growth of eye tracking as a practical method for user input to date. The proposed improvements and design elements presented in this thesis expect to strengthen the argument for eye tracking as an alternative user input device on computers.

ACKNOWLEDGEMENTS

I would like to thank my parents for their constant support and encouragement through this thesis project and through all of my endeavors; and I would also like to thank them for always leading by example. This project is the result of all of your hard work.

I would like to thank Dr. Evangelia Tzanakou for her guidance and her constant encouragement to make improvements to this work. Without her dedication to this project, it truly would not have been possible.

I would like to thank Dr. Manish Parashar for his patience during this project, for his willingness to explore projects outside of his comfort zone, and for allowing me the opportunity to learn and discover in a flexible environment.

DEDICATION

This thesis is dedicated my late grandmother, Mrs. Ernestine Davis, who despite having been immobilized due to Parkinson's disease and severe arthritis, still managed to have a great impact on many people.

TABLE OF CONTENTS

Abstract ii

Acknowledgements iv

Dedication v

List of Tables ix

List of Illustrations x

Chapter 1: Introduction 1

1.1 History of Eye Tracking 1

1.2 Statement of the Problem 3

1.3 Solution to the Problem 4

1.4 Thesis Outline 5

Chapter 2: Related Work 7

2.1 Why Eye Trackers? 7

2.2 Eye Tracker Categories 9

2.2.1 Passive Ambient Light vs. Active Infrared Light Processes 10

2.2.2 Tracking Algorithm Approaches 12

2.2.3 Mounting Locations 15

2.2.4 Selection Methods 16

2.3 Previous Eye Tracking Solutions 18

2.3.1 Appearance Manifold Model – Nearest Manifold Point Search Technique 19

2.3.2 Multi-strategy Algorithm – Feature-based and Affine Motion-based Tracking Technique 20

2.3.3 Multi-strategy Algorithm – Feature-based and Markov Model-based Tracking Technique 23

2.4 Problems with Current Eye Tracking Methods 24

Chapter 3: Methods & Materials 27

3.1 Design Methodology 27

3.2 Required Components 27

3.3 The Proposed Eye Tracking Method 28

3.3.1 Application User Interface 31

3.3.2 Locating the Eye 32

3.3.3 Calibrating the System 36

3.3.4 Face Tracking 45

3.3.5 Handling Eye Blinks 48

3.3.6 A Solution to the "Midas Touch" Problem 50

Chapter 4: Results & Analysis 52

4.1 Test Methodology 52

4.2 Test Procedure & Results 54

4.3 Results Analysis 59

Chapter 5: Conclusion 63

5.1 Summary 63

5.2 Future Improvements 68

Chapter 6: Appendices 73

6.1 Appendix A – Screen Shots of Application User Interface Menus 73

6.2 Appendix B – Flow Chart Diagrams of Eye Tracking System 84

Chapter 7: References 88

LIST OF TABLES

Table 1: Results from skilled subject tests 56

Table 2: Physical characteristics of subjects under test 57

Table 3: Results from unskilled subject tests 58

Table 4: Results from second round of testing on subject 4 59

Table 5: Analysis of results from skilled subject tests 60

Table 6: Analysis of results from unskilled subject tests 61

Table 7: Analysis of results from second round of tests on subject 4 61

LIST OF ILLUSTRATIONS

Figure 1: Regions of the Eye 10

Figure 2: Grayscale Eye Image 30

Figure 3: Converted Binary Eye Image 30

Figure 4: Black-and-White Eye Image with 3 points along each side of the limbus 32

Figure 5: Black-and-White Eye Image with center points 34

Figure 6: Calibration Screen Points 38

Figure 7: Left and Right Eye Corners 46

Figure 8: Just One Eye Corner may be d 47

Figure 9: Locating the Iris on subject 2 66

Figure 10: Locating the Iris on subject 5 67

Figure 11: Main Menu 73

Figure 12: Media Controls Menu 73

Figure 13: Communication Tools Menu 74

Figure 14: Text Editor Menu 74

Figure 15: Letters A-P Menu 75

Figure 16: Letters A, B, C, D Menu 75

Figure 17: Letters E, F, G, H Menu 76

Figure 18: Letters I, J, K, L Menu 76

Figure 19: Letters M, N, O, P Menu 77

Figure 20: Letters Q-Z, and special characters Menu 77

Figure 21: Letters Q, R, S, T Menu 78

Figure 22: Letters U, V, W, X Menu 78

Figure 23: Letters Y and Z, and characters [Space] and ‘,’ Menu 79

Figure 24: Characters ‘.’, ‘?’, [Delete], and [Clear] Menu 79

Figure 25: Application Controls Menu 80

Figure 26: Volume Control Menu 80

Figure 27: Environment Settings Menu 81

Figure 28: Temperature Controls Menu 81

Figure 29: Lighting Zones Menu 82

Figure 30: Zone (1, 2 and 3) Lighting Controls Menu (lights off) 82

Figure 31: Zone (1, 2 or 3) Lighting Controls Menu (lights on) 83

Figure 32: Basic Flow Diagram of Entire Tracking System 84

Figure 33: Flow Diagram of Calibration Process 85

Figure 34: Flow Diagram of User Interface Menus 86

Figure 35: Detailed Flow Diagram of Text Editor Menu 87

Chapter 1: Introduction

1.1 History of Eye Tracking

Interestingly enough, the idea of eye tracking as an input device to a system is not at all a new endeavor. On the contrary, the study and development of eye tracking devices has been around for many years now, and it has permeated a number of varying applications. In fact, as early as the nineteenth century, researchers began to attempt to extract information from a subject's eyes while the subject performed various tasks. Many of these early devices were rather crude; for example, in 1898 Edmund Delabarre attached a plaster cap-like device directly to a subject's eye. By attaching a writing utensil to the cap, Delabarre was able to capture (for later evaluation) the eye movements of a subject while the subject read a passage of text through a hole drilled in the cap. Other researchers performed similar studies with some success, however these invasive procedures were not very practical, since they tended to inhibit, as well as obscure the subject's eye(s) while executing the study [40]. Naturally, as research in this field progressed, it would be necessary to find new ways to study a user's visual cues without the need for such invasive techniques.

In 1901, Raymond Dodge developed one of the first successful, non-invasive eye trackers. By reflecting light off of a user's cornea, he was able to record the user's horizontal (only) eye position onto a falling photographic plate [26]. He concluded from his findings, among other things, that fixations are a necessary part of information comprehension [44]. This research proved to be groundbreaking, as similar eye tracking techniques continued to be used late into the twentieth century [40]; additionally, Dodge's findings concerning fixations and saccades are still an integral component of this

field today.

In more recent times, applications requiring more precise eye movement capture techniques have been made possible with the introduction of computing devices. One of the earliest of these applications was explored when military intelligence observed that this natural human behavior could be used as an additional means of communication between man and machine. Particularly, in 1965 the US Air Force began research on a project that would "visually couple" an aircraft operator with the aircraft's weapon control system. This innovation sought to allow a fighter pilot to simply look at a potential target, thereby causing the aircraft's sensors to automatically aim (e.g. missiles) at the intended object, meanwhile leaving the pilot's hands free for other tasks [9]. This was one of the first studies in which eye movement was used as the control for a system in this manner, but given the merit of this early work and this hands-free functionality, eye tracking has become a very attractive alternative for a number of different applications.

Since that time, researchers have successfully implemented this, as well as numerous other eye-gaze applications and have demonstrated the potential for eye tracking as a practical and effective human computer interface technique. A couple of these applications include offline studies such as analysis of user interface or web page layouts for usability concerns, and psychological experiments which observe the user's eye response to particular stimuli [13]. These applications require minimal processing capabilities, because here eye movement is not tracked in real time, rather it is recorded and later synchronized with a second recording of the scene that the user was viewing. The two recordings combined help to determine the user's gaze. Conversely, there are a

second group of applications that are finding more applicability in recent times, which do require real-time feedback to the user. Some of the applications in this category include virtual reality simulations which can detect the user's eye gaze in order to only update in detail the most relevant graphics of the display [39, 42]; in-vehicle sensors which can detect changes in a user's blink pattern and spatial attention to warn the driver that he/she may be falling asleep or losing adequate concentration on the road [8, 27]; gaze-contingent displays which reveal pertinent information based on the user's perceived interest [1, 12, 36, 48]; and a number of different interfaces which allow for hands-free command-mode operation of various computing applications. Unlike the first group of applications, these require immediate feedback to the user of the system. Therefore, in these cases computational power is of the utmost importance, since a long delay time in processing eye-gaze location would not be acceptable. Up until recent times, many of these types of implementations were only theoretical, but thanks to increases in processing speeds, real-time eye tracking applications are now in fact rather reasonable.

1.2 Statement of the Problem

In recent years, eye tracking devices have been made commercially available, so that consumers may adapt them to their own applications; however, a problem with many of these commercial devices is that they generally require very sophisticated cameras and expensive, high precision hardware. As a result, most of the tracking systems that are currently available cost anywhere from $5000 - $40,000 US, and are well out of the price range of the average consumer [19, 29, 30]. In addition to the high prices, because of the complex hardware and unfamiliar components that make up these systems, they tend to

be very complicated to set up and use [31, 2, 14]. Moreover, because many of the systems do not allow the user to switch between tracking and non-tracking modes, the system constantly responds to user input, even when the user does not intend for it. Unsurprisingly, this scenario has proven to be unnatural and uncomfortable for the user; and, in light of that, an adequate solution to this problem is long overdue.

1.3 Solution to the Problem

It is the intention of this project to implement an inexpensive eye tracking system that can easily be used by anyone who has normal vision capabilities. In order to keep the cost of the eye tracking system low, it was decided to use very simple components that are already regularly used by most consumers. This includes an ordinary web cam (to do the image capture), a normal personal computer (to do the image processing or eye tracking), and a full software solution (containing the algorithm and system flow) that may be used on most Windows OS environments. Not only will the selection of off-the-shelf items keep the cost of the system low, but since these are familiar products they should also yield a system that is easy to set up and to use.

Also, since any real-time eye tracking algorithm will need to continually process image data, the complexity of the algorithm used to do the tracking will play a crucial role in the responsiveness of the system. Of course, a lower complexity algorithm will likely have to compromise some accuracy; nevertheless, since the main priority of this work was to implement an inexpensive eye tracker that would run on an ordinary computer, it was important to cost-effectively select an algorithm. The thesis reviews several different types of eye tracking algorithms, but finally settles on a feature-based

solution, which offers the low complexity required by this project.

In addition, by employing ambient light to illuminate the user's eye, as opposed to infrared light (which is used in many commercial eye tracking systems), another opportunity was presented to lower the system cost. Moreover, it is expected that the use of ambient light will make the whole user experience much more comfortable, since the setup does not require an unnatural light shining directly on the user's eye. Additionally, dwell time was chosen for the selection method from a list of alternatives including blink selection and various manual selection methods. The idea behind dwell time is to use the normal gaze pattern of the user to allow the user to make selections without necessarily thinking about it or performing any additional task. Because this application was designed primarily to accommodate disabled users, dwell time seemed to be the most viable and natural alternative of the selection methods surveyed. Finally, a solution that permits toggling the tracking mode of the system was explored. By adding this feature to the system, the goal and the hope of this thesis is to overcome the "Midas Touch" problem and to give users a system that is very easy to use.

1.4 Thesis Outline

This chapter gives an overview of the subject of eye trackers and explains the objectives of this project and what it hopes to gain. The rest of this thesis is organized as follows: Chapter 2 gives some background information on many of the various options that are currently available in the field of eye tracking, as well as what has worked and what has not. Chapter 3 explains the design of this project and the reasoning behind the decisions for each of the selections made. Chapter 4 presents experimental data gathered

while testing the application, and analyzes the effectiveness of the application for various users. Finally, Chapter 5 sums up the lessons learned from this thesis, as it discusses the strengths and weaknesses of the application and gives ideas for future improvements to this project.

Chapter 2: Related Work

2.1 Why Eye Trackers?

The subject of eye tracking has been attracting a lot of attention during the past few years. Much of this fascination has been brought about by the fact that many of the real-time applications, that were previously just fantasy, are now feasible due to increases in computational power. However, there are a couple of key reasons why eye tracking is particularly appealing when compared to alternative input device methods. First of all, as recognized by Engell-Nielsen et al. [13], the eye gaze process is a fundamental part of human physiology, in fact, more so than the "'touching'-analogy", as they put it, used by mouse operations. In other words, it is a normal reaction for one to look at an onscreen option in which he/she is interested before taking any further action to select the option, and therefore this type of system should require no training to use, and should feel very natural. Furthermore, the obvious assumption is that using eye gaze as the activation method (thereby eliminating the manual procedure normally required to select the option) should decrease selection times, based on the premise that a user must look at a particular option before performing any kind of manual operation (e.g. mouse click or key-press). This theory was tested and upheld by Sibert and Jacob [46] who found that eye gaze activation was on the average 85% faster than mouse click activation in one particular experiment where a user was required to select highlighted circles from a grid on the screen, and 31% faster in a similar experiment which required the user to select letters specified by an audio recording. The study also found that the time required for eye movement was fairly constant (regardless of distance), so the greater the on-screen distance required to move, the greater is the savings realized from the use of eye tracking

[46]. Engell-Nielsen et al. [13] also agree that if the eye gaze system delay is not considerably greater than the time required to activate the necessary mouse click(s) and/or keystroke(s), then the eye gaze activation should be noticeably faster. Of course, the resulting delay will depend greatly on the eye gaze algorithm selected, as well as other factors, which will be discussed later.

Nevertheless, most researchers agree that eye gaze tracking should allow the user to effortlessly manipulate the computer display, without the need to put his/her arm(s) in an uncomfortable position and without having to navigate a foreign object (e.g. a mouse) which may not respond correctly or quickly enough, on occasion; the user needs only to look at the screen and the pointer is automatically in position. What is more, because eye gaze tracking does not require unnatural movement, as previously stated, this input device mode may even potentially help to prevent repetitive strain injuries often resulting from prolonged keyboard and mouse/trackball use [32]. According to one estimate, repetitive-strain injuries affect between 15-25% of all computer users worldwide [45]. Probably even more compelling, a study performed over a 3-year period by Gerr et al. [15] on 632 new employees (who used a computer regularly on the job), showed that more than 50% of those surveyed had experienced musculoskeletal symptoms due to computer use during the first year after starting their new job. Clearly, this study demonstrates that the effects of mouse and keyboard use are not trivial and that, wherever possible, alternatives should be investigated. It is also important to note that, not only are these injuries very painful, but in some ignored cases they can be irreversible and debilitating [32].

A second reason eye tracking is an attractive option to traditional input device

methods is associated with the particular user group who may stand to benefit most from this technology: those with movement disabilities, particularly paraplegic persons and those suffering from ALS. ALS (or Amyotrophic Lateral Sclerosis), also known as Lou Gehrig's disease, is a neurodegenerative disease that attacks nerve cells in the brain and spinal cord resulting in muscle weakness and atrophy. The National Institute of Health estimates that approximately 20,000 Americans currently suffer from the disease and as many as 5000 new cases in the United States are diagnosed each year [35]. Due to this and other conditions, there are many people who suffer differing degrees of paralysis, leaving some in a completely "locked-in" state with little communication to the outside world. However, because of the nature of many of these conditions, often the only remaining, functioning muscle group are the eyes, which are frequently left intact. This simple fact designates anyone in this situation a prime candidate for eye tracking communication techniques, and for some "locked-in" patients, eye tracking systems may ultimately open up a whole new world.

2.2 Eye Tracker Categories

Within the realm of eye tracking, there are a variety of approaches available that have been applied with success. There are advantages and disadvantages to each as will be demonstrated, so depending on the goals of the particular application being used, one must select the technique that most closely suits his/her needs. The first category to be examined in this thesis specifies the type of light process used in the image capture. Typically, eye trackers will use either a passive, visible spectrum-light approach or an active, infrared-light approach to illuminate the eye. A second group of eye trackers that

will be reviewed in this thesis classifies the algorithm used to track the feature of interest. The eye tracker algorithm will typically either be appearance-based, feature based, model-based, motion-based, or a combination of more than one of these approaches. The third category covers the mounting location of the eye tracker, and includes local (usually meaning head-mounted) and remote options. Finally, the last category that will be considered is the selection method used, which may be pure eye movement based strategies or a combination of manual mouse/key press techniques with eye movement based activation techniques. Eye movement activation techniques are further divided into two popular activation techniques, namely dwell time activation and eye blink activation.

2.2.1 Passive Ambient Light vs. Active Infrared Light Processes

Visible ambient light, eye-tracking processes often track the region of the eye known as the limbus, which is the outer edge of the iris (the portion of the eye that characterizes the eye's color). Figure 1 shows the limbus, the iris and some of the other common regions of the eye that are generally of interest for the purposes of eye tracking.

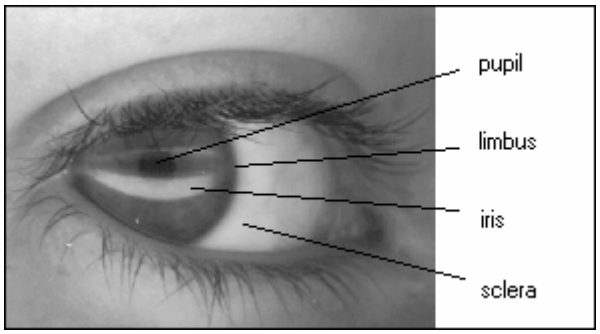

Figure 1: Regions of the Eye. Illustrates the pupil, limbus, iris, and sclera regions of an eye, useful for characterizing the eye during tracking.

The reason that passive image processes track the limbus region is because in visible spectrum light, the edge between the limbus and the sclera (the white membrane portion

of the eye) is generally most prominent. Unfortunately, one major disadvantage introduced by ambient-light eye tracking is that the limbus generally cannot be precisely detected. This is brought about by the fact that the edge of the iris is not clearly marked, rather it somewhat fades into the sclera [30]. Moreover, because of the relatively large size of the iris, the eyelids and eyelashes sometimes obstruct a clear view of the limbus. Consequently, both of these complications result in a reduced accuracy in detecting the location of the eye. Additionally, the nature of this approach is such that the uncontrolled ambient light being reflected off the image may include interference from other light sources [19]. This, in turn, can result in distortions of the feature being tracked and again lead to inaccuracies in the tracking application. However, one argument in favor of this approach is that the use of USB web cameras in ambient light generally allow for higher frame rates (normally 30 frames per second) than do the IR cameras [10]. Higher frame rates, of course, lead to smoother gaze tracking, in the case of real time, eye tracking; so this aspect provides a pointed advantage over the use of IR light.

In infrared-light, eye-tracking processes, an infrared (IR) light is shined on the user's eye in order to locate the user's pupil. A portion of the IR light ray that shines on the user's eye is reflected back through the pupil and points directly towards the light source, producing a glint in the image of the pupil. This quality helps to locate the user's eye and also provides a frame of reference from which all movements may be based. In this case, when IR light is used to illuminate the eye, the pupil is now the most salient feature of the user's eye, since it can be easily located using the glint of light reflected from the user's eye. This is advantageous considering the fact that the pupil is much more sharply defined than the iris. Also, in contrast to the passive visible spectrum-light

approach, another advantage to active IR-light approaches is the relatively small size of the pupil, as suggested previously, which is not as likely to be obstructed by the eyelids and lashes [30]. Furthermore, since the exact contour of the iris or pupil is not known ahead of time, most algorithms will estimate the shape to fit a circle (or ellipse) that will roughly characterize the region. Again, because the pupil is smaller, there is less room for error when calculating the center of the region, as compared with the iris. This provides a definite advantage for the use of IR light as opposed to visible spectrum-light processes.

On the other hand, the use of an IR-light process not only increases the overall price of the system (which, of course, is not desirable for a low-end, consumer application), but it may also require installation by professionals to ensure user safety (which would not lend the system well to general use) [3, 19]. Also, IR light, eye-tracking systems are limited to indoor use [51]. This is not a restriction of passive, ambient light approaches, so they allow for more flexibility in the design setup, including in-vehicle eye-tracking devices and other setups. Even more important, infrared light has been reported to cause discomfort to users' eyes when employed for long periods of time [19, 30]. For someone who has no other means of communication, this is clearly not an option. As a result, this study will confine its focus to visible spectrum eye tracking.

2.2.2 Tracking Algorithm Approaches

Traditional image-based passive approaches can be divided into further categories based on the eye tracking algorithm used. The nature of the algorithm used has a great influence on the accuracy of the tracking device, as well as the computational time

involved and the flexibility of the algorithm on varying image data (i.e. user eye and skin color, and changes in environment). Various eye tracking algorithms include but are not limited to appearance-based, feature-based, model-based, and motion-based approaches.

Appearance-based methods rely on building models that are developed directly by the appearance of the eye region under examination [30]. This type of method stores a large collection of images that it later uses to compare to the feature of interest. Essentially, it treats each image as a single point in an *n*-dimensional space, where *n* represents the number of images in the collection [50]. For eye tracking, it first assigns an image (of the eye) to each possible gaze position on the screen. Then upon subsequent image captures, it attempts to compare the current image with one of the stored image sample points to find the best match. This method has the advantage that it is relatively robust, since it searches for features that have already been confirmed. However, because of the large data storage requirement, it necessitates a huge amount of computation to compare images. It is worthwhile to note that this computation can be reduced by using feature extraction techniques such as those employed in [33]; however, additional processing time would be required to detect the feature under investigation. For the purposes of this research, the feature of interest (in this case, the user's eye) will be manually extracted from the total image using a simple windowing technique, which will be discussed later in this thesis. Any additional feature extraction developments may remove important information from the image needed to accurately determine the user's gaze direction, so for that reason no feature extraction techniques were investigated for this thesis.

Feature-based algorithms generally search the image for a set of facial features

and then group the features into potential candidates based on their geometric shape [28]. One common characteristic of feature-based algorithms is that they commonly require a threshold to decide when a feature is present or not [30]. Because this method is relatively simplistic, it usually does not require much computation and can be implemented in a very efficient manner. However, because these algorithms rely solely on predefined thresholds, changes in lighting conditions and head poses may degrade the accuracy of the system [51].

In contrast, model-based approaches find the best fitting model that is consistent with the image. In other words, this method does not assume an initial model, instead it allows for exploration of varying image data [43], and accordingly it is more tolerant to changing user data sets and will likely also be more adaptable to differing environmental settings. Therefore, this approach may provide for a more accurate estimation of the pupil (or iris) center, however at the cost of lower computational speeds [30] due to the time required for constructing the models.

Motion-based methods construct a difference image, which represents the divergence between the current and previous image frames, to determine the movement that occurred between the capture of the two images. Again, a threshold is used, but in this case, to match components from one image to the next and to decide which of those components have moved. In the case of eye tracking, a blink can be used as the indicator to locate the eyes; in addition, the change in direction of gaze will also trigger a difference image [10]. Motion-based methods are less likely to fail during occlusion of the eyes, but they are not as accurate as some of the other methods mentioned previously [47]. For this reason, these methods are not normally used alone [10, 47], but are instead

used in conjunction with other tracking approaches.

2.2.3 Mounting Locations

In addition to the type of light source used to track the eye image, another option to be considered is the locality of the eye tracking system with respect to the user. Systems are usually classified as either remote or head-mounted. Generally speaking, remote systems must require that the user be completely still, otherwise they will incorporate some form of face tracking algorithm that can compensate for small movements of the head. One advantage of remote systems is that web cams, which are now widely available and easy to install, can be used to provide the video capture mechanism for these systems. A second advantage is that remote systems are less intrusive than head-mounted systems which must cope with obstruction (by the camera) of the user's normal field of view; remote systems usually mount the camera a couple of feet from the user's face, while head-mounted systems usually feature a camera just centimeters from the user's eye. Although the initial expectation may be that remote systems are less restrictive and provide greater freedom, ironically, they allow for less freedom of movement than head-mounted systems. The problem with remote systems (even those which feature face tracking algorithms) is that sudden and large movements of the user cannot be handled, which ultimately leads to inaccuracies in the tracking results; and the user is essentially tethered to the view frame of the camera.

Although head-mounted systems are obviously more intrusive than the remote alternative, micro-lens cameras which are constantly getting smaller and smaller can be placed under a user's eye with little distraction. Even though the target user group of this

project generally may not have autonomy over their limbs, a subset of this group may suffer from involuntary spasms, which could compromise the accuracy of the system. In light of these findings, a head-mounted, eye-tracking device may be most ideal for these specific cases; however for the majority of users, remote camera setups will suffice.

2.2.4 Selection Methods

Since this study is concerned with using the eye tracker in real time as an active input device, it should allow the user to select options, and activate a given set of commands, just as he/she would from a mouse or keyboard. Typically, in most eye tracking systems, this is done by creating a user interface that gives the user soft-key options, which he/she may select on-screen. However, the method by which commands get activated in these systems seems to be a matter of preference.

Most tracking devices use eye movement as the trigger to activate a selection. The most notable options are eye blink activation and dwell time activation (which simply activates the object designated by the cursor after a given time period). Phillips et al. [37] and Murphy et al. [34], respectively, used eye blink activation techniques to operate a computer switch and an on-screen word writer, both for severely disabled people. Still, Jacob argues that using an eye blink to indicate the selection of an object is not an optimal solution because this scenario forces the user to think about when he/she blinks, thus detracting from the natural feel that he/she would expect from an eye tracking system. Instead he implements a dwell time solution combined with an optional, manual button press to immediately override the delay, if so desired. However, after he conducted research studies, he found the dwell time alone to be more convenient,

preferring instead a reduced dwell time to speed up system response [23]. Hansen et al. also use dwell time selection because they found it to be simple and comfortable to the user [17, 20], however many researchers have also found that using dwell time as the command activation method introduces a whole new problem. The problem is that users are not accustomed to having commands activated just by looking at screen objects; if they forget to move the cursor, they will likely activate commands unintentionally. Jacob refers to this phenomenon as the "Midas Touch", noting that everywhere the user looks, a new command will be activated. He also expresses the need for an interface which can "act on the user's input when he wants it to and let him just look around when that's what he wants..." but concludes that it is not possible to make these two distinctions [23].

A number of researchers have proposed a manual keypress for the purpose of performing an object selection. Yamato et al., for example, implemented an eye tracking device that used eye positioning to move the cursor across the screen and a mouse click to invoke an onscreen button push operation [53]. When using this method, they found object selections to be faster than when using normal mouse operations. Another study done by Zhai et al. explores the efficiency of using manual operation (of a mouse) for both fine manipulation of the cursor and object selection – using eye gaze only for coarse positioning of the cursor. They argue that traditional eye gaze tracking is flawed in the fact that it couples the user's vision with motor control, causing the resulting system to be unnatural. They propose that their method will yield a system that is more natural and accurate than eye gaze alone, while faster and less prone to physical fatigue and injury than manual mouse operation [54]. In addition, in both of these cases the "Midas Touch" ceases to be a problem, as the user must deliberately press a manual key to perform a

selection, which should shore up the inherent weakness for unintentional selections usually associated with eye gaze control. Of course, these methods are not ideal for a user with limited mobility. Also, interestingly, some researchers have found that in applications where a dwell time is employed, if it is carefully selected, the user should not even notice an unnatural consequence, but instead get the feeling that the system somehow knows what the user wants to do as soon as the user does [24, 41]. Based on the information provided, it appears that dwell time may be the best alternative for a comfortable, hands-free object selection technique. Consequently, for the purposes of this project, a dwell time solution has been applied in order to allow the user to achieve an effortless operation of the system. The dwell time has been set to 500 msec in order to exploit the natural eye movements of the user, based on the findings reported by Jacob [23] in which no case could be made for a dwell time longer than 3/4 second. Jacob notes that "people do not normally fixate one spot for that long" and that longer dwell times give the user the impression that the system has crashed [23]. In addition to this feature, functionality has also been added to cope with the "Midas Touch" problem in a novel way. By incorporating a "stateful" tracking design, this implementation expects to make the "Midas Touch" problem a mere memory. This simple solution will be discussed later in this thesis.

2.3 Previous Eye Tracking Solutions

When creating the design for this eye tracking system, it was important to take into account many of the related works that had previously been developed. By taking into consideration some of the mistakes that had been encountered by previous

researchers, this project was able to address some of the problems that have limited the use of eye gaze tracking in the past. This project also benefited by being able to simply build on some of the successful design objectives that have already been formulated. With the wealth of research available in this field, there was plenty of information on which to build. Just a few of these solutions were chosen to review in this thesis, but the solutions were intentionally selected so that they include very different design characteristics. This way, they cover a broad range of the design alternatives that are available in this field.

2.3.1 Appearance Manifold Model – Nearest Manifold Point Search Technique

Tan et al. [50] use an appearance-based or view-based algorithm to track the user's eye. They also use an IR illuminator to facilitate the eye detection process. In theory, an indefinitely large collection of images of the feature being tracked (referred to as the appearance manifold) would form a continuous set of sample points, thus allowing any test image to be matched to a point in the high-dimensional space. However, in practice this is not possible; and instead of using a densely sampled search space to model this idea, which would result in long computation times, they use an interpolation method, which approximates points on the appearance manifold by applying linear combinations to a set of neighboring sample points.

The algorithm commences by selecting the three closest sample points (in terms of similarity to the image being tracked – also known as the test point) that form a triangle. All adjacent points are also added to this selected set of sample points. The algorithm then assigns weights to these sample points, using the Least Squares method

[50], such that the error, $\in$, is minimized when Equation (1) holds true

$$\in = | x - \sum_i w_i s_i | \quad (1)$$

where x is the test image (or test point, assuming that each image is a point in the n-dimensional manifold space), $s_1 \ldots s_k$ are the image samples selected, and $w_1 \ldots w_k$ are the set of weights, which must also satisfy Equation (2), as follows.

$$\sum_i w_i = 1 \quad (2)$$

Finally, these weights may be applied as described in Equation (3),

$$Q_y = \sum_i w_i Q_{si} \quad (3)$$

in order to estimate the parameters of the theoretical sample point y, which would correspond to the test point (where Q_y is the parameter vector for point y).

To initialize the system, the application first requests that the user align the mouse pointer with several random points on the computer screen, while automatically capturing an image of the user's gaze for each point. It uses these points to build a discrete appearance manifold. In practice, the team used 252 such sample points. After analyzing their results, Tan et al. [50] found that they were able to achieve an average angular error of 0.3838 degrees, which is competitive with commercial eye tracking systems. However, the method does use a very large number of calibration points, which is not realistic for a practical user application.

2.3.2 Multi-strategy Algorithm – Feature-based and Affine Motion-based Tracking Technique

In this method, Smith et al. [47] employ a feature-based strategy in combination with a motion-based eye tracking approach. In particular, pixel intensity and skin color

are used to help locate the user's eye pupils in the feature-based portion of the strategy. Since the feature-based strategy tends to produce errors when the eyes are occluded due to certain movements and rotations, a motion-based enhancement was selected to help ensure that the system is more robust during these situations. The affine motion detector tracks the user's eyes by locating finite points at distinguishable features in the frame and then determining the distance that the pixels have moved based on the transformation that occurred between image frames. Since this method is generally not as accurate as the feature-based method, it is used only as a supplement to the afore-mentioned strategy.

The pixel intensity strategy is first applied using this method such that the system searches the pixels that formed the center of mass of the darkest pixel from the previous image frame. This pixel is identified as the pupil only if it is close enough (based on a search space around the eyes which is limited to a window of about 5 percent of the image size) to the previous pupil location; otherwise, the skin color strategy is applied. Again, a limited region is searched, this time to locate clusters of pixels that are not skin-color in nature. This allows the system to identify the user's eyes. In addition to this, clusters of pixels that match the color criteria for the user's lips are also identified in an effort to locate the lip corners. This characteristic is based on threshold parameters from observed users. Once all nonskin-color regions have been discovered, the two correctly identified eyes may be selected by comparing the slope of the line that passes through the lip corners. The line having a slope closest to that of the lip corners is selected, and the pupil centers associated with this line are acquired. The former pixel intensity strategy is then reapplied to verify the newly obtained results, since the skin color strategy is more prone to failure during occlusion of the eyes.

In addition, a third strategy is also applied concurrently to these. This strategy tracks the affine motion of the image, and based on the transformation computed, it is able to approximate the new center of the pupil. When the first two strategies do not produce acceptable results, the third strategy must be used alone. Otherwise, the result achieved from the first two strategies will be averaged with the result from the third. This average value will serve as the estimated pupil position. The system also includes a threshold that monitors the distance between the suspected pupil centers and the number of pixels contained in the supposed lips, requiring that they be within certain boundaries observed through experimentation.

Overall, the system is proven to be fairly versatile, working in various lighting conditions as well as in situations with a moving background, but this system also demonstrated its inconsistencies in performance as the results of several video clips of users gazing at various locations were examined. On a set of tests which verified the correct gaze direction of the user (including a determination of whether the user had blinked or not), the system scored anywhere from 7% - 100% correct readings. Also, because the process requires three distinct complex strategies, it requires a good deal of computation, which may cause the eye-tracking algorithm to be considerably slow in a real-time tracking situation. However, since this project was actually used for offline (video-recorded) studies, the strategy implemented in this system was indeed suitable for the application. Nevertheless, to make this design more practical for real-time use, the approach would undoubtedly require considerable rework to reduce the amount of computation.

2.3.3 Multi-strategy Algorithm – Feature-based and Markov Model-based Tracking Technique

Bagci et al. [5] propose an eye tracking method that also utilizes skin color for feature detection. In this case, the proponents use skin color for head tracking. The algorithm identifies the face as the largest cluster of skin-color pixels, where the thresholds for skin color verification are determined empirically by testing users with different skin tones. In order to track the user's eyes, first they locate four feature points corresponding to the user's nostrils and eyebrows using a set of rules based on the predetermined geometry of the features and the knowledge that the nostrils and eyebrows do not exhibit the skin-color criteria. From these data, the location of the user's iris(es) may be estimated, and a bounding box is placed around the approximate region of the user's eyes (within the image). Further, a threshold is used to distinguish between the user's iris and sclera in order to detect an edge. Using this edge as a point of reference, the translation of two images that occurs between frames will estimate the motion of the user's eye, which is tracked by using the transformation matrix given in equation (4), below.

$$\begin{bmatrix} u' \\ v' \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{4}$$

Specifically, this matrix helps solve for the position of the iris at various instances during the application. The coordinates (x, y) represent the iris in the initial frame and (u', v') are given by equations (5) and (6)

$$u = u'/w \tag{5}$$

$$v = v'/w \tag{6}$$

where (u, v) denote the coordinates of the eye in the projected frame. The elements of the matrix (shown as a_{ij}) represent the state transition probabilities of the 3D coordinate space and are modeled using a Markov framework which classifies the user's gaze into five different states (up, down, left, right, and forward), based on the position of the iris. The projected iris coordinates (u_i, v_i) may be calculated from the transformation matrix using equations (7) and (8), below.

$$u_i = a_{11}x_i + a_{12}y_i + a_{13} - a_{31}x_iu_i - a_{32}y_iu_i \quad (7)$$

$$v_i = a_{21}x_i + a_{22}y_i + a_{23} - a_{31}x_iv_i - a_{32}y_iv_i \quad (8)$$

Based on user trials, the application was able to correctly identify the location of the user's iris in 98.5% of the frames; however, the application demonstrated to be less accurate when there are frames containing closed eyes. Interestingly, this particular eye tracking application is not designed to carry out any specific task, rather its sole purpose is to simply demonstrate its accuracy in locating the user's iris, and it indicates this by placing a marker on the centers of the irises. Ideally, additional functionality would be added to the application to make it more practical.

2.4 Problems with Current Eye Tracking Methods

One of the problems with current real-time eye tracking methods is that they often use an infrared light process to illuminate the eye. While these processes are typically more accurate than visible light processes, they tend to be uncomfortable to the user, often causing the eyes to dry out after extended periods of use. As a rule of thumb, if the IR light is causing discomfort to the user‘s eyes, it is likely also causing damage to the user's cornea, lens, and/or retina. One reason for this occurrence is that the normal

defense mechanisms, which cause a person to blink when exposed to natural bright light containing infrared light, do not take place when the visible part of the light has been removed [7, 11]. As a result, the subject is unnaturally exposed to higher concentrations of light than the eyes can actually handle. This exposure may cause excessive warming of the eye tissues and can lead to a number of different eye-related medical conditions. Some of these conditions include keratitis, cataracts, retinal detachment, and glaucoma; and these conditions may ultimately result in a possible loss of focus, blind spots, and even total blindness to the subject [7].

An additional problem that all of the eye trackers surveyed in this study have in common is that they do not account for the Midas Touch problem. This problem prevents the user from manipulating the mouse pointer as he/she would normally expect, because the tracker continuously selects screen objects as the user surveys the screen. Indeed, several eye trackers [21] have used dwell time for the implicit purpose of preventing unintentional selections directly resulting from this Midas Touch phenomenon. However, dwell time actually does little to counter the effects of this problem, as observed by Hansen et al. [21], who note that in one particular study (where dwell time was used as the selection method) subjects accidentally activated screen options when they neglected to move the mouse pointer to a designated screen location. One suggestion offered in the study was to increase the dwell time to allow the user more time to react. Although users may find this solution to be effective to a certain extent, the increased dwell time also has the effect of slowing the selection process considerably. In light of this fact, a proper balance must be selected, otherwise if the dwell time is selected to be too short, then the number of erroneous selections will likely be high; then again, if

the dwell time is selected too long, it negates the speed advantage potentially obtained by using eye movement as the input device in the first place [25] and limits ease with which a user can exercise the system.

A number of systems have been proposed which certainly accomplish the task of tracking the user's eye(s), yet none of these systems provide a mechanism to distinguish when the user is just casually viewing the screen versus the case when he/she is attempting to cause the system to react to his/her gaze location (i.e. the same problem originally posed by Jacob in [23]). Thus, a solution has not been found which alleviates the Midas Touch problem; indeed, in order to make eye tracking devices more consistent with the manner people have become accustomed to using the computer, it seems necessary to address this issue. In fact, Bakman et al. consider this problem to be so pivotal that they feel until a workable solution has been developed, eye tracking for general-purpose use is doubtful [4]. This thesis intends to address each of these problems as it presents an original approach to eye tracking with simple off-the-shelf components that can easily be assembled by the average computer user.

Chapter 3: Methods & Materials

3.1 Design Methodology

One of the main goals of this project was to provide a solution to eye tracking that could be implemented from inexpensive components. This would allow the average computer user to take advantage of the technology, and give the users who need it most (i.e. disabled users) a viable and affordable alternative solution. In order to allow for this, it was important that all hardware items used in the production of the project were ordinary off-the-shelf items that could be purchased easily and economically by the consumer. In addition, software solutions were used wherever possible within the project, since software allows for seamless changes to the system by simply swapping out an existing module for a new one. As a result, this would permit subsequent updates to the system without the need to change the physical system design, thus keeping expenses low. Encouragingly, this design will also allow future developers to make their own updates to the project in order to improve its performance and utility.

3.2 Required Components

The following is a list of the components needed to build the proposed system:

- Computer with Microsoft® Windows® 98 (or better) OS
- Logitech® QuickCam® Orbit™ (or similar webcam with at least 640 x 480 pixel resolution and 300% magnification zoom feature); and included webcam driver software
- Desk mount for webcam (optional)
- J2SE v1.4.2_10 SDK with Netbeans 4.1
- Java Media Framework 2.1.1e, or later
- Executable .jar file to run the eye tracker application

- Bite bar or chin rest (optional)

3.3 The Proposed Eye Tracking Method

In an effort to keep costs low in this project and make the user's experience as comfortable as possible, it was decided to elect an ambient light, feature-based image processing approach, as opposed to using the more intrusive infrared light eye tracking process. As such, the iris has been targeted as the feature to be tracked, since it is the most prominent characteristic of the eye available in visible light.

First, an ordinary webcam was positioned in front of the user, so that the eye to be tracked is within view. Preferably, the camera resolution should be as close to the screen resolution as possible. The monitor display resolution has been set to the default 800 x 600 pixels, and a Logitech QuickCam Orbit camera is being used to capture the image with the pixel resolution set to the maximum (640 x 480 pixels). However, at this resolution and with the camera set at a comfortable distance away from the user (approximately 1 foot away, with the computer monitor roughly 2 feet away), the image of the eye does not occupy the whole camera view, so the use of a cropping window as mentioned in [38] helps to target the feature of interest, and excludes the nonessential data just as a more involved feature extraction technique would. In this case, the window resolution should be as close to the screen resolution as possible. However, since the application only captures a small window around the user's eye, the effective resolution is considerably less than the 640 × 480 pixel resolution that the camera is equipped to handle (and of course, much less than the default 800 × 600 pixel resolution of the computer screen), and this value will change from one run of the application to the next. In the current implementation of the design, the user is alerted to move a green window,

which is overlaid on the camera view, so that it encloses either his/her left or right eye. The user may position the window to his/her preference by clicking, and then dragging the mouse cursor, and releasing the mouse. In the event that the feature to be tracked does not fill the whole camera view, this windowing measure provides the benefit of reducing the image size, so that the insignificant data within the image does not have to be rendered. This results in a faster, smoother refresh of the video stream. In addition, the resulting window contains a smaller number of pixels to compute when processing the edge detection algorithm, and therefore saves valuable memory and CPU time during the actual tracking effort [38].

A black and white camera is sufficient for this task, and is actually preferred to a color camera since it is generally less expensive and performs better in low light situations [49]. Also, when it comes to visible light eye tracking, a black and white (actually gray-scale) image often appears considerably sharper than a color image. As a result, when converting a binary image from the gray scale image, it is more accurate than a converted color image. This distinction will be important during the next step in the imaging process.

A full software solution is used to capture and filter the images in an effort to save time by eliminating the need to convert data from analog to digital. The Java Media Framework makes it possible to establish the webcam as the data source that will be used to do the tracking, and to set up a data stream through which the streaming video may be accessed. Now, this makes it easy to write frames of the video stream to a .jpg file at regular intervals. The .jpg image file may be used to locate the feature of interest (in this case, the user's iris); but to make eye tracking simple, it is convenient to first convert the

image to a binary set of data. The java Graphics package makes it very easy to access and manipulate image file data, so that the digital gray-scale image may be converted to a binary image by using a threshold to classify pixels as either black or white. Figure 2, shows an example of an image that has first been cropped from the originally captured (640 × 480 pixel) image during an actual run of the application. In this example, the

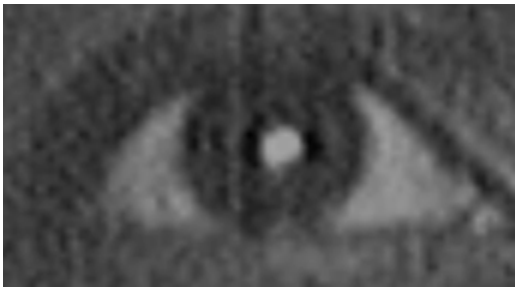

Figure 2: Grayscale Eye Image. A subject's eye taken during an actual run of the application

Figure 3: Converted Binary Eye Image. The same image after a threshold has been applied to convert it to black-and-white.

outcome is a 256 × 141 pixel image, which permits the application to run significantly faster due to the reduced data set. Figure 3 shows the resulting image after the application has performed the binary threshold conversion. In an image containing just the eye, the limbus can generally be identified as a somewhat circular-shaped black edge surrounded by the two largest white clusters of pixels in the image. In the black-and-white image above, it may be noted that the second largest white region is a glint in the center of the user's iris, caused by light reflecting off of the eye. The glint in this image would be rejected as a sclera region because, although it is positioned to the left of the accepted right sclera region, the curvature of the (possible) "limbus" at this region does not conform to the normal curvature characterized by a left limbus region (which would usually curve in the opposite direction); as a result, one of two next largest white regions would be selected. In this case, however, the limbus is simply characterized as the right

edge of the iris bordering the largest sclera region, and the data gathered from the left iris edge is discarded (as will be explained in more detail in section 3.3.2 – "Locating the Eye").

3.3.1 Application User Interface

The application user interface was designed with just a few large buttons to allow the user to easily select options on the screen. As a matter of fact, the design has four buttons, one in each quadrant of the display (upper left, upper right, lower left and lower right corners). The user interface was designed to incorporate features that would be most important to a disabled user, but also includes features that any user may find handy. The user interface includes functions such as environment control features (which currently include a temperature display panel that may be adjusted up and down; and lighting displays which may be turned on and off), a text editor that allows the user to type a message, a browser button which allows access to the internet, a volume control menu which adjusts (and mutes) the volume of the application, and a recalibration button which allows the user to recalibrate the system. At the center of the interface, there is also a window that displays a scaled image of the user's eye (currently being captured). Additionally, the main menu of the user interface includes a "quit" button to allow the user to end the tracking program, and most of the menus also include a "Previous Menu" button to allow the user to navigate to the preceding menu screen. A complete layout of all the user interface menus has been provided in Appendix A. Figure 34 of Appendix B provides a flow diagram of the user interface menus, demonstrating the result of each button keypress, while Figure 35 gives more details specific to the text editor menu

functionality. (Figures 32 and 33, incidentally, outline the flow design of the eye tracking system as a whole.)

3.3.2 Locating the Eye

Upon capturing each image, the algorithm uses the information provided concerning the relative position of the iris with respect to the sclera to locate three points along the limbus -- one near the upper eyelid, another near the lower eyelid, and a third mid-way between the two. Incidentally, either side of the iris may be tracked to locate the three points, depending on the iris's position with respect to the sclera. Figure 4 shows the three points on the right iris edge and the three points on the left iris edge, obtained from the algorithm using the image given previously in Figure 3. The first set

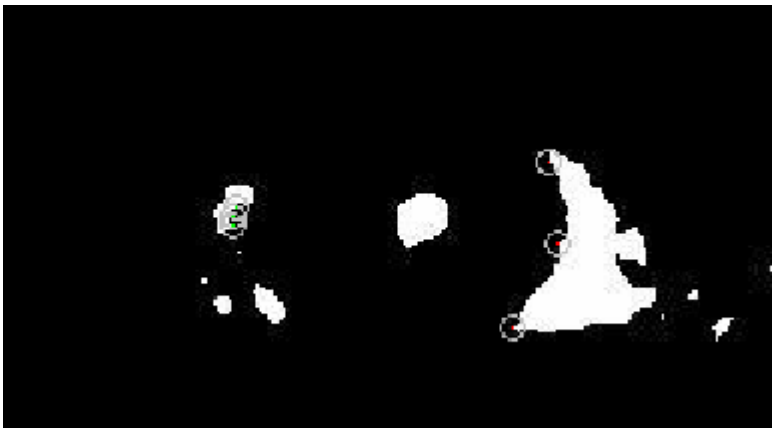

Figure 4: Black-and-White Eye Image with 3 points along each side of the limbus. The same image from Figure 3 enlarged, and illustrating the three points located automatically along the (left and right) limbus region of the eye found during an actual run of the edge detection algorithm.

of points that were located by the system are shown as red pixels with a gray circle around each of them, and appear on the right side of the iris in this image. The second set of points are green with a gray circle, and are on the left side of the iris. After detecting the three points along the limbus (from both the left and right sides), a fairly good

estimate of the center of the iris may then be attained by characterizing the iris as an exact circle. From geometry, it is clear that given any three points on the circumference of a circle, the perpendicular bisector of the first two points will intersect the perpendicular bisector of the second two points at the center of the circle. Specifically, given the points P_1, P_2, and P_3 having coordinates (x_1, y_1), (x_2, y_2), and (x_3, y_3) respectively, the midpoint of P_1 and P_2 will be defined as $(\frac{1}{2}[x_1 + x_2], \frac{1}{2}[y_1 + y_2])$; and the midpoint of P_2 and P_3 is defined as $(\frac{1}{2}[x_2 + x_3], \frac{1}{2}[y_2 + y_3])$. It can also be obtained that a perpendicular line passing through the segment connecting P_1 and P_2 has slope equal to $(x_1 - x_2) / (y_1 - y_2)$; and, a perpendicular line through the segment connecting P_2 and P_3 has slope equal to $(x_2 - x_3) / (y_2 - y_3)$. Knowing that the equation of a line having slope m and passing through the point (a, b) is $y - b = m(x - a)$, the following equations may be generated to characterize the two perpendicular bisectors:

$$y - \tfrac{1}{2}(y_1 + y_2) = (x_1 - x_2) / (y_1 - y_2) * [x - \tfrac{1}{2}(x_1 + x_2)] \quad (9)$$

$$y - \tfrac{1}{2}(y_2 + y_3) = (x_2 - x_3) / (y_2 - y_3) * [x - \tfrac{1}{2}(x_2 + x_3)] \quad (10)$$

Re-writing the two equations with y alone on one side and then subtracting them, eliminates the y variable to solve for x as shown in equation (11). Then by getting x alone on one side and subtracting equations (9) and (10), y may be determined as shown in equation (12) below.

$$x = [-\tfrac{1}{2} * (x_1 + x_2)(x_1 - x_2)(y_3 - y_2) + \tfrac{1}{2} * (y_1 + y_2)(y_2 - y_1)(y_3 - y_2) + \tfrac{1}{2} * (x_2 + x_3)(x_2 - x_3)(y_2 - y_1) - \tfrac{1}{2} * (y_2 + y_3)(y_2 - y_1)(y_3 - y_2)] / (-x_2 * y_1 - x_3 * y_2 + x_3 * y_1 - x_1 * y_3 + x_1 * y_2 + x_2 * y_3) \quad (11)$$

$$y = [-\tfrac{1}{2} * (y_1 + y_2)(y_2 - y_1)(x_2 - x_3) + \tfrac{1}{2} * (x_1 + x_2)(x_1 - x_2)(x_2 - x_3) + \tfrac{1}{2} * (y_2 + y_3)(y_3 - y_2)(x_1 - x_2) - \tfrac{1}{2} * (x_2 + x_3)(x_1 - x_2)(x_2 - x_3)] / (x_1 * y_3 - x_2 * y_3 - x_1 * y_2 + x_3 * y_2 + x_2 * y_1 - x_3 * y_1) \quad (12)$$

These (equations (11) and (12)) provide the (x, y) coordinates for the center of the circle, C. From here, the radius of the iris may also be calculated for later use throughout the application. Applying Pythagorean's theorem, the radius of the circle may be obtained by calculating the hypotenuse of the right triangle formed by the x and y coordinates of one of the three points found previously with the center of the circle. So, the radius, r, is given as:

$$r = [(x_1 - C_x)^2 + (y_1 - C_y)^2]^{1/2} \quad (13)$$

Using this approach, the approximated centroid of the iris (approaching from the right side of the eye) from the above example is illustrated in Figure 5, as a red cross-hair. The approximated

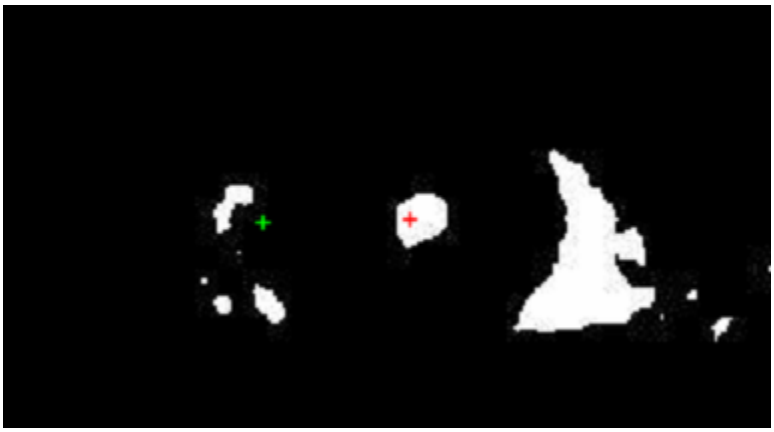

Figure 5: Black-and-White Eye Image with center points. The same image from Figure 4, illustrating the computed center points of the iris found during an actual run of the application.

centroid approaching from the left side is given as a green cross-hair in the same image. As displayed in Figure 5, the left iris center point (green cross-hair) is very close to the edge of the iris due to the distortion of the left edge of the iris. If this image were taken during an actual run of the program, the left center point would be rejected since the radius would not accurately characterize the iris, and only the right iris center point

would be used. However, if the image were taken during capture of the intial (center) calibration image, the image would likely not be accepted since the two center points are rather far apart. (This is, of course, dependant on the threshold set prior to testing.) In this case, the calibration step (and image capture) would be repeated in order to obtain a more suitable image. An image clearly displaying both sides of the iris is preferred during this calibration stage in order to more accurately resolve the user's gaze when either side of the iris is obstructed. Adjusting the brightness setting on the webcam should quickly solve this problem.

It is important to note that, although the centroid may not be precisely determined using this method (given the fact that the shape of the iris is not actually a perfect circle), the method does provide enough information to make a good estimation about the position of the eye. Additionally, the radius is calculated to be 49.77 (pixels) for this example. The radius of the iris is used simply to characterize its size during the application of the tracking algorithm. Upon capturing the center calibration image, the radius of the iris is also computed and this value is retained in a variable for later use. By defining a threshold, it can be decided whether the iris has actually been identified in subsequent iterations of the algorithm based on the original size of the iris. If the iris does not meet this criterion, then it may be concluded that the iris has not been found. In the case where the iris does not meet the necessary criterion, the eye is considered closed and the situation may be handled as either a deliberate or an inadvertent eye blink. The two different situations will depend upon the circumstances, as described in section 3.3.5 – "Handling Eye Blinks".

This process will be repeated throughout the application to locate the user's eye.

Although this approximation of the iris will ultimately lead to some inaccuracy of the algorithm, the method only requires 3 points along each side of the iris to locate the user's eye, so the computation is minimal when compared to other methods including the method used by Li et al. described in [30] which requires as many as 18 points to locate a user's pupil. This difference in computation is particularly significant, since this process of locating the user's eye will need to be repeated during subsequent iterations of the tracking algorithm. As a result, this helps to reduce the cursor delay time and makes the application appear more "user-friendly".

3.3.3 Calibrating the System

The first step in using the eye tracking application is to calibrate the system. Once the eye has been successfully located, the calibration process allows the system to determine the specific screen location with which eye gaze fixations will be associated, as well as the range of movement and the size of the iris which will be allowed as acceptable inputs. One of the common weaknesses found in many of the eye trackers implemented in the past is that they require a very large number of calibration points. For example, the appearance-based model in [50] uses 252 calibration points, and the neural network-based methods in [6] and [52] use 2000 and as many as 4000 calibration points, respectively. In both of these methods, the mouse cursor moves automatically across the screen to a number (equaling the total number of calibration points) of different predetermined screen positions, however the user is expected to visually follow the movement of the cursor until the calibration is complete. The problem with using such a large number of points is that the task of calibrating the system becomes very

tiresome to the user. As a result, most users would not be willing to endure this long process, so systems using a large number of calibration points are not likely to be successful [18]. As such, many of the high-precision, commercial eye trackers manage to calibrate their systems using only 9 points; however, this thesis proposes that just 5 points will give the system the precision needed to locate a few large objects on a specially designed user interface, making calibration very quick and easy for the user. Then it presents a second method which reduces the number to just a single calibration point, while still managing to provide a workable solution to eye tracking with the help of a practical user interface implementation which was designed and provided along with this thesis.

3.3.3.1 Selecting Calibration Points and Recording Eye Locations

In the initial system design, the calibration targets were placed at the following five screen locations: 1. the center of the screen, 2. the leftmost point of the screen (centered vertically), 3. the rightmost point of the screen (centered vertically), 4. the topmost point of the screen (centered horizontally), and 5. the bottommost point of the screen (centered horizontally). Figure 6 displays the respective locations of the calibration points on a representation of the user display, having each point marked with an “x”. The points are positioned in this manner so that the boundaries of the screen may be determined, as well as the change in position from the center to each extremity of the screen.

During the initial calibration of the system, the points are displayed to the user one at a time. The program requests the user to focus on a predetermined point at the

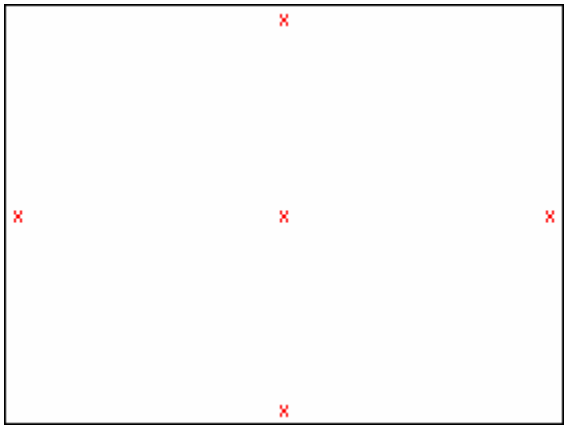

Figure 6: Calibration Screen Points. Illustrates a representation of the calibration screen showing the approximate locations of all calibration points, as they would appear on the computer monitor (although, during an actual run of the application, only one point would appear at a time). First, the center point would be displayed, then left, right, top, and bottom would follow consecutively.

center of the screen, and then on subsequent points at the left, right, top and finally the bottom of the screen. The application uses both visual and audible signals to instruct the user what to do. This is helpful because it allows a user who has no experience with the system to feel confident using it for the first time. First, the interface speaks aloud "Center", and then it displays at the center of the screen "3", "2", "1", "0". It is worth noting that the count down is currently set to start at 3, however it may be easily changed by adjusting a value in a file reserved for constant variables to give the user a sufficient but comfortable amount of time to get ready. As the user looks at the target, the application grabs the current image frame captured by the camera. The interface repeats this action for each of the remaining calibration points to capture images as the user's eye fixates on each target point. Upon capturing each image, the application locates the centroid of the iris from both the left and the right side, and then saves each coordinate pair to a Point variable for later use. During the calibration phase, the system also performs several checks to attempt to ensure that the application has correctly identified the eyes while performing the edge detection algorithm on each image. An overview of

the algorithm used to perform the calibration process is described below:

0: if (eye is closed) OR (iris cannot be detected from both sides) OR (distance of LHS iris center from RHS iris center is greater than threshold) OR (ratio between LHS iris radius and RHS iris radius is greater than threshold)
    go to 0
else
    center calibration image = current iris center

1: if ((left distance ≤ 0) OR (eye is closed))
    go to 0
else
    left calibration image = current iris center

2: if ((right distance ≤ 0) OR (eye is closed))
    go to 2
else if ((right distance < (left-to-right threshold ratio) * left distance) OR
        (right distance > (1 - left-to-right threshold ratio) * left distance))
    go to 0
else
    right calibration image = current iris center

3: if ((up distance ≤ 0) OR (eye is closed))
    go to 3
else
    top calibration image = current iris center

4: if ((down distance ≤ 0) OR (eye is closed))
    go to 4
else if ((down distance < (top-to-bottom threshold ratio) * up distance) OR
        (down distance > (1 - top-to-bottom threshold ratio) * up distance))
    go to 3
else
    bottom calibration image = current iris center

When the system captures the center calibration image, it performs a check to verify that the eye is not closed and that the iris may be detected from both sides (left and right). It also compares the center position of the iris as detected from the left side of the eye with that detected from the right side, to make sure that the distance between the two "centers" is within a given range, and that the radius of the iris detected from the left side is within

a given range of the radius detected from the right side. However, no check is performed to verify the accuracy of the position of the gaze, since there is no prior frame of reference against which to measure. However, upon capturing the left calibration image, the application performs a check to verify that the eye is open and also that the centroid of the iris (in the left calibration image) is indeed in the negative x-direction (as compared to the center calibration image); otherwise, the algorithm repeats both calibration points to ensure that an erroneous capture during the first calibration point does not compromise the accuracy of the calibration process. When the algorithm captures the right calibration image, it checks to verify again that the eye is open and that it is in the positive x-direction, this time. If both of these conditions are not met, then the algorithm repeats the current calibration point. Also, if the center-to-right image distance is too dissimilar from the center-to-left image distance (this threshold is established by an adjustable global threshold variable), then the algorithm will repeat all calibration points from the beginning. This way, the system (and user) can make certain that the left and right calibration images are fairly balanced, thus indicating a good likelihood that all three were read correctly. From here, as the top calibration image is captured, the algorithm verifies that the eye is open and that the centroid of the iris has moved in the negative y-direction. (The reason that the top calibration pixel assumes a negative direction along the y-axis as compared to the center point has to do with the way the image file is stored to an array.) Comparable to the ordering of pixels on the computer monitor, the image pixels are stored to a 2-D array starting with the top left pixel and ending with the bottom right pixel. Hence, in both cases, the y-value increases in the downward direction. If these requirements do not hold true, then the top calibration point

is repeated. Finally, the bottom calibration image is captured and the algorithm verifies that the user's eye is open and that the eye has moved in the positive y-direction, otherwise this last calibration point is repeated. And again, if the center-to-top image distance is too dissimilar to the center-to-bottom image distance, then the algorithm repeats the last two calibration points (since the first three have already been verified). Once the algorithm completes this process, the application can be confident that the calibration images are at least as equally balanced as the thresholds require. From experimental inspection, this calibration process generally allows for a granularity of approximately one cursor location for every 15 screen pixels, which means that based on an 800 x 600 pixel screen resolution, the cursor may move to 50 unique screen locations along the x-axis and 40 unique screen locations along the y-axis. This implementation worked fairly well, but partially due to the response time delay, it was somewhat confusing to accurately maneuver the mouse cursor.

To simplify the application design and make it easier to operate, it was then decided to use just one calibration point. The single calibration point allows for the user to move the mouse cursor to just one point in each quadrant of the screen, but since the user interface has only four buttons per menu, this limitation does not pose a problem for the application design. Furthermore, it greatly reduces the time required for the calibration process making the system much more user-friendly.

3.3.3.2 Mapping Eye Movement to a Screen Location

It was previously demonstrated how the eye can easily be tracked using just three points along the perimeter of the iris (the feature of interest) to locate the iris center. It

was also displayed that by referencing the centroid of the iris from the center calibration image, the relative position of subsequent eye gazes may be determined. Now, in order to do something useful with this information, it is necessary to translate these image positions to a location on the computer screen. First, the remaining calibration image locations may be used to obtain distances from the center of the screen to the screen boundaries. Using a calibration procedure having 5 calibration points, the leftmost change in gaze direction is determined by subtracting the x-coordinate of the iris centroid of the left calibration image from that of the center calibration image, as shown below in equation (14). (This value is treated as the maximum left change in gaze direction because if the user looks any further left, his/her gaze will be off the screen, and the cursor will stop at the edge of the screen.) The other maximum differences in gaze direction (given in equations (15) through (17), below) are calculated in a similar manner, subtracting each respective calibration image from the center calibration image (or vice-versa, where applicable) and then taking the absolute value of the difference. The maximum differences in left, right, up, and down eye gaze directions are as follows:

$$\Delta I_L = Ci_x - Li_x \qquad (14)$$

$$\Delta I_R = Ri_x - Ci_x \qquad (15)$$

$$\Delta I_U = Ci_y - Ti_y \qquad (16)$$

$$\Delta I_D = Bi_y - Ci_y \qquad (17)$$

where Ci_x, Li_x, and Ri_x are the center, left, and right image x-coordinates, respectively; and Ci_y, Ti_y, and Bi_y are the center, top, and bottom image y-coordinates, respectively. Now to correlate the change in gaze location (from the images) to a distance on the

screen, it is necessary to also find the maximum differences from the center of the screen to the extremities of the screen by subtracting the center calibration point from each respective calibration point. This distance will be given based on the number of pixels between the two points, and although the calibration points will be essentially fixed, the screen resolution is subject to change depending on the user's preference. Thus, the Java Toolkit package is used to resolve the number of on-screen pixels automatically, so that it is not necessary to hard-code the number of pixels using a single screen resolution value. As a result, this detail gives the application added flexibility. Incidentally, the screen calibration point distances are given as follows:

$$\Delta S_L = Cs_x - Ls_x \tag{18}$$

$$\Delta S_R = Rs_x - Cs_x \tag{19}$$

$$\Delta S_U = Cs_y - Ts_y \tag{20}$$

$$\Delta S_D = Bs_y - Cs_y \tag{21}$$

where Cs_x, Ls_x, and Rs_x are the x-coordinates taken from the centroids of the images retrieved as the user gazes at the center, left, and right screen calibration points, respectively; and Cs_y, Ts_y, and Bs_y are the y-coordinates obtained from the centroids of the images as the user gazes at the center, top, and bottom screen calibration points, respectively. From these results, the detected eye-gaze movement may be mapped to an on-screen mouse cursor movement. By dividing the on-screen distance (given in equations (18) – (21)) for each of the four directions by the change in location of the calibration images (given in equations (14) – (17)), it is possible to obtain the translation for the number of on-screen pixels which equate to one pixel in the image. This is given

below for each of the four directions, left, right, up and down (respectively), as follows:

$$L_{I \to S} = \Delta S_L / \Delta I_L \quad (22)$$

$$R_{I \to S} = \Delta S_R / \Delta I_R \quad (23)$$

$$U_{I \to S} = \Delta S_U / \Delta I_U \quad (24)$$

$$D_{I \to S} = \Delta S_D / \Delta I_D \quad (25)$$

This provides an outline for determining the respective screen locations. Once the calibration phase has ended and the application begins tracking the eye, a change in both the x- and y- coordinate locations will first be calculated from the image based on the original center calibration image. Accordingly, these distances may now be translated to a screen location (where the effective origin is taken as half the pixel width in the x-direction and half the pixel height in the y-direction) by selecting the appropriate pair of mappings from equations (22) through (25), above.

Since there is generally not a one-to-one correlation between the image and the screen resolution, the mouse cursor will tend to jump from one location to the next. Nevertheless, it is apparent that the described algorithm may successfully be used to maneuver the mouse cursor to specific areas of the screen. Thus, by using just these five calibration points, it is possible to yield a fairly accurate tracking mechanism.

Using just one calibration point makes this process much simpler. In this case when the x-coordinate is less than (or equal to) the calibrated user eye location, the system determines that the user is gazing at the left half of the screen; otherwise, it is assumed that the user is gazing at the right half. When the y-coordinate of the user's eye is found to be less than (or equal to) the calibrated user eye location, the system

concludes that the user is gazing at the upper half of the screen; otherwise, the user is gazing at the lower half. The two coordinates together establish one of the four quadrants of the display.

3.3.4 Face Tracking

Now that the translation algorithm has been established, the system can estimate any gaze position to a position on the screen. The only problem is that this estimation is based on the assumption that the user's face has remained in the same position since the time of the calibration. Without the aid of a chin rest or bite bar to keep the user's head still, this assumption is almost certainly flawed. Furthermore, for many users the idea of using a chin rest or bite bar is probably not the most comfortable arrangement. To this end, a head-movement compensation model was explored in an attempt to eliminate the need for any such accessory.

A head-movement compensation model (or face tracker) uses a stationary target on the head to attempt to locate the position (and displacement) of the user's head during gaze tracking. One obstacle presented in the proposed method is that since only the user's eye is in the camera view at any given time, the possibilities for head tracking are very limited. In [55], Zhu and Ji use the glint on the user’s eye to determine the relative position of the user’s face; however, since the proposed application does not use a controlled light source (e.g. an IR light beam) to illuminate the user’s eye, this solution is not a viable option. In order to overcome this obstacle, the current project made use of the user’s eye corners, which seemed to be an encouraging solution. The idea is that even when the user's eye moves within its socket, the corners of the user's eye should remain

motionless with respect to the user's face. Also, when the user opens and closes his/her eyes to varying degrees, the corner of the eye is expected to stay in relatively the same position, much as a hinge on a door. Therefore, any movement that is detected from the eye corners implies a movement of the user's head, and this distance and direction should be deterministic.

Using this reasoning, the eye corners were determined by locating one pair of extreme points on the top eyelid, and a second pair on the bottom eyelid. Similar to the technique used for finding the iris, the eyelids were located by first identifying the sclera regions. Since the sclera regions were found previously to locate the iris, this step of locating the eyelids was done in parallel to minimize the computation cost. Once the four points were located, the equation for the line connecting each pair of points was determined. Using the two equations, the eye corner was estimated to be the point of intersect of the two lines. Figure 7 shows an eye image taken during an actual run of the program, and displays the line segments that were created by connecting each pair of points on the left and right sides of the upper and lower eyelids. The image also shows

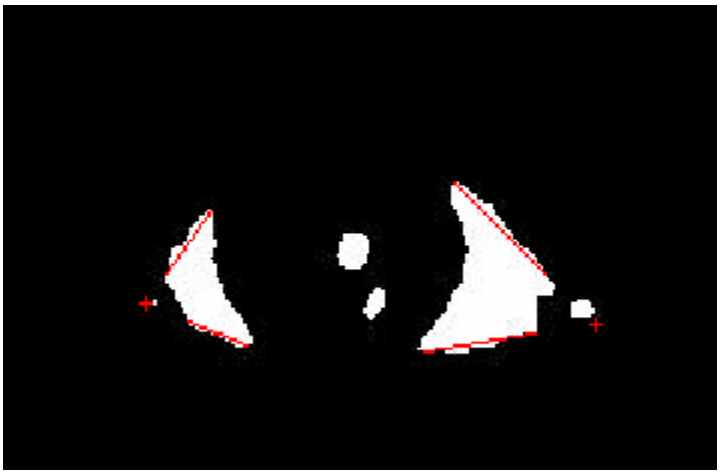

Figure 7: Left and Right Eye Corners. Illustrates the technique used for locating the user's eye corners. These eye corners are used to try to establish a stationary point of reference on the user's eye, so that (in addition to gaze movement) any movement made by the user's head may also be tracked.

(as red cross-hairs) the resulting eye corners that were established by the system during that frame. During the calibration process, both the left and right eye corners are located and used during the program to determine how far the user's head has moved from the original position. After the calibration process (while the tracker is running), the system will decide whether to use the left or right eye corner based on the same criteria used previously for determining which center point to apply. In Figure 8, only the right corner is detectable since the iris center was rejected from the left side. Once the head

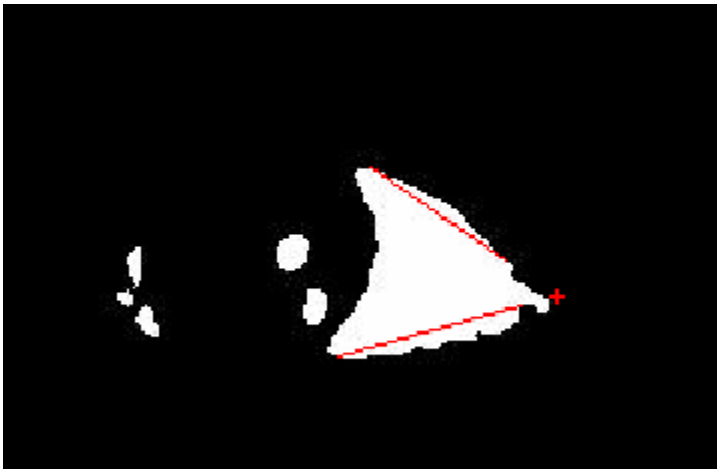

Figure 8: Just One Eye Corner may be detected. Illustrates an image of the eye where only one of the two eye corners may be accurately determined, since the left side of the iris is not clearly defined. In this scenario, the algorithm is able to use the one eye corner that is available to calculate the amount of head movement.

displacement is determined with respect to the calibration image (again using the technique established in equations 14-25), the total eye movement (including any head movement) is offset by this amount in an attempt to resolve the actual gaze. In addition, the entire bounding box is automatically repositioned (in the same direction and distance as the head was computed to have moved) to try to place the eye in the same relative position within the new image window. The expected result is that the whole eye appears to stay relatively still, while only the iris moves.

3.3.5 Handling Eye Blinks

During the execution of the first calibration point (the center point), the application also records the radius of the iris (as found from both the left and right sides) to two separate variables named *originalIrisRadiusFromLHS* and *originalIrisRadiusFromRHS*. Then during subsequent iterations of the tracking algorithm (including the remaining calibration points), it checks the corresponding (left-hand-side (LHS) or right-hand-side (RHS) radius of the iris in the current frame to verify that it does not exceed a certain threshold (which may be adjusted via a global variable). The significance of this check is that, in the event that the iris has been incorrectly identified using the scanning process described above, the appropriate action may be taken to reject the detected iris. Most incorrect reads will be the result of a blink by the user, in which case the iris will, of course, not be detectable within the image. Incidentally, all unacceptable iris readings will be treated as a blink.

In the present implementation of the system, the iris radius threshold is set to 0.6. This means that if the current iris is found to be less than 0.6 times the original iris size or greater than 1.4 times its size (i.e. ± 0.4 times the original iris size), it may be concluded that an eye blink has occurred for that frame of the video. This is assuming that neither the LHS or RHS radius is acceptable. Of course, there will be inadvertent blinks during use of the application since the user cannot control when he/she blinks, so the application attempts to ignore these. Specifically, the algorithm states that if it has been determined that the user's eye is closed during a frame of the application, then a certain variable will be incremented to keep a count of the number of consecutive eye closures. However, if the iris is successfully found (i.e. the eye is open), then this variable is reset to 0. While

the variable is less than the threshold, an image frame that has been detected to contain a blink will simply be ignored. On the other hand, if the number of consecutive iterations is equal to or greater than the threshold amount, then the application interprets this situation as a deliberate blink. Since the user will have a limited number of input mechanisms using only his/her eyes, it is convenient to use a deliberate blink as an additional communication tool. Originally, this blink operation was used to perform an object selection, but after using the application it was observed that (as also noted in [23]) this solution was not very natural to use and it also sometimes caused the user's gaze to go astray just before capturing a blink. This would result in the selection of an incorrect object, or a missed selection. Therefore, it was decided that a variation of the dwell time selection method would instead be used to issue commands. In this case, if the user gazes at a selection object for the threshold number of iterations, then the object is determined to be of interest and becomes selected automatically. This method feels more natural to the user, and further does not produce the unwelcome effects of misguiding the cursor as a result of forcing the user to interrupt his/her gaze with a blink.

Despite the fact that blink activation may not be the ideal selection method for this application, the use of an intentional blink can provide an advantage over a tracking application that does not utilize this input mode. As noted earlier, for someone with severely limited mobility, the eyes may be the only exploitable faculty to which the user has access; therefore, at the risk of introducing an unnatural movement into the application, it would likely be beneficial to utilize this signal, since it is available to the physically disabled user. Accordingly, the following proposes a blink mechanism that prospectively will not be particularly uncomfortable to the user. In addition it should

provide an additional communication input to allow the user to use the eye tracker more effectively. Furthermore, the intention is to use this deliberate eye blink mechanism to find a solution to an old eye-tracking problem.

3.3.6 A Solution to the "Midas Touch" Problem

The problem with many eye tracking applications, especially those that use dwell time activation to perform selections, is that they are in a constant tracking mode state, whether the user is looking to perform an action or not. This situation causes the phenomenon where the user performs selections unintentionally, because the user interface cannot distinguish the user's intent to select an object from normal looking. Other researchers attempted to solve this problem by proposing a manual button for selecting on-screen objects; however, this solution is not ideal because it assumes that the user has a natural ability to control his/her hands and arms. Clearly, for some this may not be the case, so in keeping true to the hands-free design of this research, this project proposes a tracking-mode, toggle switch that may be controlled with a simple blink to allow the system to be controlled completely without the need for any manual interaction. Of course, it is necessary to devise some measure to distinguish between an involuntary blink and one that is done with intent to communicate some information. One solution requires the user to employ a long blink to issue a command. Although this action may be somewhat unnatural for the user, unlike the blink activation method mentioned previously, this blink response would be used only minimally when the user decides to switch in and out of tracking mode and would not cause a serious distraction to the user. Applying this method, if the eye has been determined closed for a number of iterations

equal to or greater than the blink threshold, then the tracking mode would be toggled between on and off. This would allow the user to casually view the screen when tracking mode is off, and to actively control the system when tracking mode is on.

Chapter 4: Results & Analysis

4.1 Test Methodology

The user interface provides a number of useful functionalities for the user, but for the purposes of evaluating the accuracy and efficiency of this application, the text editor function was selected as the basis for analyzing the system performance. This feature was selected because it provides a measurable task that must be performed using a sequence of steps. The text editor UI was designed to make text entry as simple as possible. The first screen of the text editor menu is composed of four user areas, including one button that returns the user to the previous menu. Also essential to the menu is a text area (which displays the user's text input) and two buttons that allow the user to select any of the 26 letters of the alphabet (plus a few additional important characters).

In addition to the 26 letters of the alphabet, the current implementation also provides functionality for a space character, period, comma, question mark, delete character (which deletes the last character entered), and a clear character (which clears the entire text area), for a total of 32 characters. (Additional characters may be added to future revisions of the software design, but would come at a cost of an additional selection operation, as will be demonstrated later in this section.) Since there are only two remaining user input areas, they must somehow provide text entry for all 32 characters.

Initially, the menu was designed such that one of the two remaining buttons was dedicated to the task of scanning through the entire alphabet. The other button was used to select the current character and display it to the user text area. However, because only

one button was being used to scroll through all 26+ characters, the process sometimes required more than 26 selection operations to get to the desired character. Additionally, the system response delay often caused the user to overshoot the character requiring him/her to scan through the entire alphabet all over again. To alleviate both of these problems, a second strategy needed to be devised. Because of the limited number of inputs available, it was important to design a text input menu that is easy to use, but required as few selection operations as possible. This constraint led to the decision to use a divide-and-conquer technique to allow access to all of the characters in O(log n) selection operations. More specifically, the characters are first divided in half between the two text entry buttons on the first text editor menu screen, placing characters A through P on one button and Q through Z plus supplementary characters on the second button. Pressing either of these buttons will advance the user to a screen having four characters on each of the four buttons, for a total of 16 characters from which to choose. Then, pressing one of the four buttons on the next text editor screen takes the user to a third screen having just one character on each button. Thus, when a user selects a button, the application returns to the first text editor screen and updates the user text area by appending the selected character to the end of the text string. Overall, this divide-and-conquer technique was found to be much easier to operate and it allows for the input of any character in exactly $\text{ceil}(\log_4(\text{ceil}(n/2))) + 1$ selection operations, where in this case n equals 32 (the total number of characters provided in the application). As a result, any character may be typed to the screen using just three selection operations. However, initially one problem still remained with this implementation. In particular, since text characters occupy all four buttons on the second and third text editor screens, there is no

button reserved for navigation to the previous menu, as in all other cases. In order to return to the first screen of the text editor menu, the user must navigate through the character selection screens until he/she selects a single character. Consequently, if the user accidentally selects a button that does not contain the desired character to be typed, he/she cannot return to the text entry screen without entering an unwanted character. To remedy this aberrant behavior, a time-out period was implemented such that if no character button has been selected within the specified time period, the menu will automatically return to the text entry screen, without the need to enter an unwanted character. (For more details on the correct text editor menu operation, refer to the flow chart diagram provided in Figure 35 of Appendix B.) Results from testing this implementation are provided in the following section.

4.2 Test Procedure & Results

The study consisted of five participants. Using the eye tracker text editor menu, they were each asked to type out a short sentence. The subjects were timed to measure how long it took to type the sentence, "I am hungry." The time was started as soon as the subject was placed on the text editor menu, and the time was stopped right after the subject correctly typed the last letter of the sentence. (Subjects were not asked to navigate to the text editor menu themselves, however they were required to re-enter the menu if they accidentally exited the text editor while performing the test, and in these instances the timer continued running and the (exit menu) button press was counted as an error.) Incidentally, the head-movement compensation feature that was discussed in section 3.3.4 was not used during testing because it was found to respond inconsistently

and it caused the system to be far less accurate. As a result, all subjects were requested to sit very still, as any movement after the calibration procedure would throw off the accuracy of the tracker. During the test interval, all incorrect button presses were recorded to attain an overall, selection error rate for each run. (It is important to note that an incorrect selection did not necessarily always lead to an incorrect character entry; instead an incorrect selection was presumed to represent any button press that could not lead to a correct character entry. This could also include a button press that did actually produce an incorrect character entry.)

During the first phase of the test, the first subject was tested for 10 regressions in an attempt to measure the consistency and repeatability of the system on the same subject. The subject was also very familiar with the system, and had used it many times prior to testing; so, this subject qualified as a skilled user. In each of the 10 regressions, the subject was able to successfully complete the task. The results from the 10 test runs were tabulated and are shown in Table 1. Although the subject was able to complete the task, he noticed that in some instances, before he could select an intended character, the screen timed out and returned to the text editor menu. In addition to this, on occasion the system froze (and stopped responding altogether for several seconds), forcing the subject to have to wait to continue typing. Sometimes this happened just as the subject was attempting the final button press, which would have resulted in a character entry. Other times system pauses led to incorrect character selections after the system resumed normal operation. These occurrences wasted a good deal of time, and inevitably caused the text entry times to be slower, and perhaps even caused the error rates to be higher than they may have been otherwise.

During the second phase of the test, the four remaining participants were tested for just one run. The remaining subjects were purposely selected such that they had dissimilar physical attributes. Here, the objective was to verify the robustness of the

Table 1: Results from skilled subject tests. Displays the results from testing the first subject through 10 regressions of the test procedure.

Regression #	**Time** (min:sec)	**# Incorrect Selections**
1	2:30	1
2	2:10	1
3	2:10	0
4	5:53	7
5	3:10	3
6	2:27	4
7	4:06	5
8	3:09	2
9	2:48	3
10	2:03	2

system's tracking algorithm when tested on a diverse user group. Table 2 illustrates the physical characteristics of all participants tested in the study (including the subject from the first phase of testing).

None of the remaining subjects had ever used the system before, so they may all be classified as unskilled users. Each of the four subjects was given instructions on how to use the system and specifically on how the text editor menu works. Then each subject was given a few minutes to become familiar with the text editor menu by allowing the

participant to manipulate it manually. Additionally, before the subjects could actually use the eye tracker, it was also necessary to adjust the camera "Brightness" setting to make sure that the system was able to correctly detect a clear black-and-white image of

Table 2: Physical characteristics of subjects under test. Lists the physical characteristics of all 5 subjects to capture any possible trends or limitations in the system detection model.

Subject #	**Gender**	**Skin color**	**Eye color**	**Other characteristics** (which may have an influence on the system)
1	male	brown	brown	none
2	male	olive	green/gray	wide eyes
3	female	dark brown	brown	glasses
4	female	white	brown	none
5	male	light brown	brown	growth on iris

the user's eye. (This step was generally necessary anytime a new user used the system, or if the position or lighting of the system had changed.) Once the camera settings had been sufficiently adjusted, the application was initiated. None of the unskilled users were able to successfully complete the requested task. Next tested, subject 2 was able to type the characters, "I k ", but after being unable to successfully control the cursor any further, the subject gave up. Although this subject sat fairly still compared to some of the others, he found that he had difficulty using the system. The time was officially stopped after 10 minutes had elapsed. Subject 3 struggled quite a bit with the test. She had a hard time keeping her head in the position where she started and was only able to type the characters, "Ca". The time was finally stopped after 10 minutes, since the subject could not make any additional progress. Subject 4 was also able to type just two letters, "Ai" and after 10 minutes the time was stopped. Finally, subject 5 was unable to type any

characters at all, despite the fact that he was able to sit relatively very still. Table 3 captures the results from these four test runs.

Table 3: Results from unskilled subject tests. Displays the results from testing the four unskilled system users.

Subject #	**Time** (min:sec)	**# Incorrect Selections**	**Actual Text Output**
2	stopped after 10:00	12	I<space><space>k<space>
3	stopped after 10:00	5	Ca
4	stopped after 10:00	10	Ai
5	stopped after 10:00	3	

Since the unskilled users were not able to successfully navigate the system during the first round of testing, it seemed necessary to demonstrate that a user could master the eye tracker after a little practice. For this purpose, one of the unskilled subjects was selected to retest. Subject 4 was selected because she most greatly differed from the skilled subject in skin color. The subject was allowed to use the eye tracker for about a half hour for five days before repeating the test. During the practice sessions, the subject was not requested to perform any particular task, but was asked to simply try out the different functions on the application to become more familiar with it and to get more comfortable selecting the on-screen buttons. After the practice sessions were complete, subject 4 was given 10 trials in which to complete the original task. The results from this study are shown in Table 4. As shown in the table, the subject was much more successful during the second round of testing. She was able to complete the task on each attempt, and felt she had more control over the application during the second round of testing.

Table 4: Results from second round of testing on subject 4. Displays the results from testing a subject after a moderate amount of training.

Regression #	**Time** (min:sec)	**# Incorrect Selections**
1	6:11	7
2	7:26	12
3	3:43	4
4	6:24	9
5	8:33	13
6	5:58	6
7	6:17	8
8	8:42	11
9	5:21	6
10	4:30	5

4.3 Results Analysis

After testing all five of the participants, it was clear that having some practice made a distinct difference in the subject's ability to effectively use the system. This was not only evidenced by the contrast in results between the skilled tester and the other testers, but also by the fact that most of the unskilled participants began to gain more control in making accurate selections as time went on. Subject 4 noted that by the time the test ended, she was beginning to feel more confident about the system responding correctly to her inputs. One possibility is that the unskilled test group were not as adept at sitting motionless as was the skilled tester. However, it was concluded that all the

subjects tested could consistently turn the tracking mode On/Off. Perhaps, with more practice the unskilled group of testers would also be able to type complete sentences and interact with the user interface more effectively.

Be that as it may, even the results from the skilled subject left much room for improvement, as character selections produced an error rate of 0.25 errors per character selection, as can be seen in Table 5. In the table, the per character statistics were

Table 5: Analysis of results from skilled subject tests. Displays an analysis of the system test results from the skilled user.

Avg. Time for total task (with std. dev.)	**Avg. # Incorrect Selections for total task** (with std. dev.)	**Avg. Time per** (correct) **character**	**Avg. Error Rate per** (correct) **character**
3:03 ± 1:07	2.8 ± 1.99	16.64 ± 6.09 sec./char.	0.25 ± 0.18 errors/char.

obtained by dividing the average time and error rate (found in Table 1) by the total number of characters in the test sentence, "I am hungry" (or 11 characters). Here, it may be observed that the error rate per character entry is actually rather high, at 25%. However, it should also be noted that this error rate includes 3 selections for every character. Accordingly, in order to find the effective error rate per selection, it is necessary to divide the number of errors by the total number of selections, in this case 33. As a result, the actual average error rate observed while testing the skilled subject was closer to 8.5%.

The results from the unskilled user group were far less impressive showing error rates ranging anywhere from 600% to 1000%. The respective error rates from the unskilled user test group may be found in Table 6. (Note that the error rate per selection in this case is not a good measure for comparison between subjects, since the subjects all

Table 6: Analysis of results from unskilled subject tests. Displays the error rates obtained from the text entry results of the unskilled user group.

Subject #	**Error Rate per** (correct) **character entry**	**Error Rate per selection operation**
2	6 errors/character	2.4 errors/selection
3	(cannot be defined)	2.5 errors/selection
4	10 errors/character	5 errors/selection
5	(cannot be defined)	(cannot be defined)

achieved different levels of completion of the task with differing degrees of correctness.) A better measure, for comparison purposes (in this case), would be the success rate per selection; however, since the total number of screen selections that were actually executed during the test was not recorded, this statistic is not available.

The results from the second round of testing are displayed in Table 7. These

Table 7: Analysis of results from second round of tests on subject 4. Displays analysis of the system test results from subject 4.

Avg. Time for total task (with std. dev.)	**Avg. # Incorrect Selections for total task** (with std. dev.)	**Avg. Time per** (correct) **character**	**Avg. Error Rate per** (correct) **character**
6:19 ± 1:31	8.1 ± 2.91	34.45 ± 8.27 sec./char.	0.74 ± 0.26 errors/char.

results show that the error rate per character selection is still a very high 74%. However, when taking into account once again the number of screen selections required for every character selection, one can see that the effective error rate per screen selection is actually much lower, at 25%. Although the results were not as convincing as those from the skilled user, they do give some indication that as a user learns to use the system, his/her success rate will improve. Since the subject only used the system for less than a week,

the results show merit that with more application, the system may prove to be a practical solution.

Chapter 5: Conclusion

5.1 Summary

Eye tracking has been a topic of interest in a number of different fields for many years now. Researchers have applied the study of eye tracking to passive applications to gain useful information about both individual and more general human behaviors. By gathering data from natural eye movements, experimental media applications are able to use the information to deliver a more personalized media experience to their consumers. Other applications, particularly in the realm of simulators and virtual reality programs, are using eye tracking to take advantage of smart image processing, whereby areas on a screen that are not of interest to the user will not be rendered as clearly, thus saving valuable clock cycles in processing time. These are just a few of the many applications that are being explored through this technology, and new useful discoveries are being made all the time. After all, the eyes are important communication tools; and the most interesting thing about this communication is that people do it without even thinking about it. For this reason, an entirely different set of applications has also gained quite a bit of interest. This set of applications use active eye tracking to provide input to various devices to perform real functions in a new way. Some research in this area is being done specifically to provide communication tools for those who have no other means of communication. Other researchers also suggest that the fundamental qualities of eye movement to communicate information should allow for anyone to make use of these applications, and with even greater ease than manually operated devices. Yet, due to a few critical limitations with these types of applications, they are still not being widely used today. One of the most significant problems with this technology is that reliable

systems are still far too expensive for the average consumer. An additional problem with active eye tracking applications is that due to a phenomenon coined as the "Midas Touch" problem, they are not natural to the user. This thesis has proposed an original eye tracking implementation that it feels has addressed both of these issues, while also providing a reliable system. The system was specifically designed to give paralyzed users a means of communication, but it was developed with the hope that a wider audience may also find a practical use for this application.

5.2 Discussion

The data gathered as a result of evaluating this eye tracking system indicate that with a little refinement the software design presented in this thesis may be very practical for use by almost anyone. The success rate obtained from the skilled user study, (at 92% accuracy on average) is in line with many of the results reviewed in the comparison studies (found in Chapter 2 of this thesis). In addition, the intermediate user (subject 4, after training) showed some improvement from the initial study with a success rate of 75%. Of course the time required to perform selections, particularly in the text editor menu, has to improve; but, this may be corrected to some extent by revisiting the time-out sequence currently implemented in these screens.

There were, however, a few limitations uncovered as a result of testing. First, all of the subjects found the time-out period on the character entry screens to be a hindrance. In some occasions, the screens timed out before the subject could make his/her selection. In order to resolve this problem, the time-out period could conceivably be increased to try to reduce the occurrence of this situation, but it was found during earlier tests of the

application that a longer time-out period becomes very bothersome when a user has made an incorrect selection, and finds that he/she must wait for a long time to return to the text editor menu. In fact, even the current time of 6 seconds can become a burden to a more skilled user. An obvious solution to this problem would be to add a "Previous Menu" button to the character selection screens, and remove the time-out sequence altogether. Of course, this would translate into additional selections per character entry. To be exact, given the current character set, each character selection would require 4 button presses instead of the current 3. At a glance, this disparity does not seem like much, but when added to every character selection operation, it may cause selection times to be significantly greater. However, after factoring in the wasted time (caused by the problems with the time-out functionality mentioned previously), it may pay to implement such a change.

The system also experiences some critical lock-ups that prevent the user from using the system as expected. These lock-ups most likely occur due to the large number of resources required by the program, in addition to the large sets of data input/output continuously being fed and retrieved from the processing unit. Any efforts to ensure that the program runs more efficiently will likely require new developments in this area to help eradicate these lock-up occurrences.

Additionally, because of the camera sensitivity to lighting, the tracker is somewhat difficult to set up, and it requires a good deal of attention to this detail in order to get it to work properly. Also, by design, the tracking algorithm relies on a group of predetermined characteristics in order to make a determination on whether the iris has indeed been discovered. While testing subjects 2 and 5, these criteria proved to be too

discriminating as the tracker had a difficult time recognizing the iris and even passing the calibration process. While reviewing some of the images taken during calibration, it was observed that, for these two cases, the shape of the eye images might have caused some problems for the eye detection algorithm. In the first case, subject 2 had relatively wide eyes, to the extent that even when he looked straight ahead, the bottom rim of his iris was above his eyelid. This caused problems during calibration, as the calibration routine expected to find an eye image with two separate sclera regions (when the user was focusing at the center of the screen). In this subject's case, there was only one large sclera region that wrapped around the lower edge of the iris. Figure 9 shows an image of subject 2's eye, including the iris detection points and the iris center located during tracking. In the second of these two cases, subject 5 had a growth (possibly a cataract) on

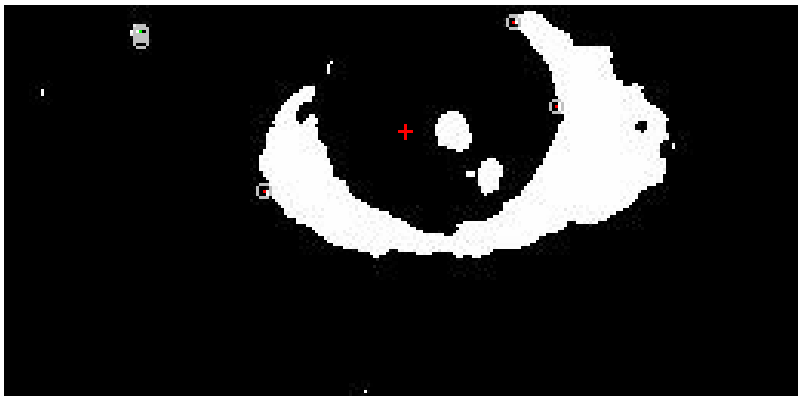

Figure 9: Locating the Iris on subject 2. The system struggled to locate the iris on subject 2 because the subject did not have two clearly defined sclera regions. The third iris point from the right sclera region is actually on the left side of the iris and not at all on the iris edge. The left iris points have incorrectly been located (clustered together) at the top left of the image.

his eye that overlapped part of the outer edge of the iris. This growth altered the shape of the user's iris (at least in many of the images), so that the iris appeared concave on one side. This distortion made it difficult for the system to detect his iris correctly and, as a result, made it difficult for him to use the eye tracker. Figure 10 illustrates the eye of

subject 5, highlighting the iris perimeter points and centers detected from both the left and right sides of the iris. From the image, one can observe that the two center points are

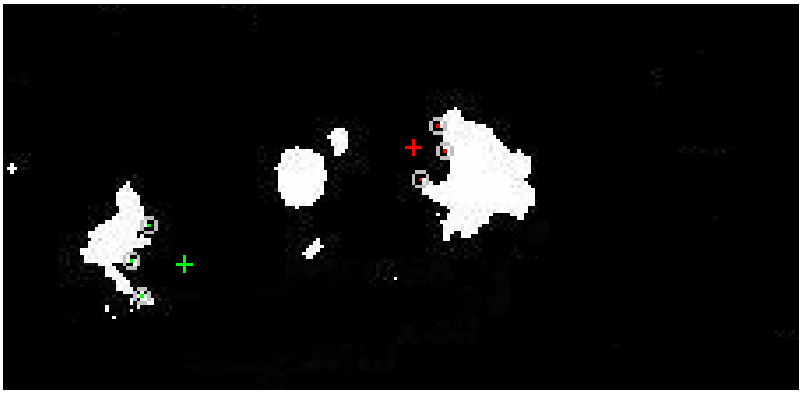

Figure 10: Locating the Iris on subject 5. The system had a difficult time locating the iris on subject 5 because this subject had a growth on his iris which obstructed its outer edge. The right side of the iris clearly has an indentation which causes the tracker to incorrectly estimate its center.

very inaccurate and very far apart. Obviously, this system had a hard time identifying the iris in both of these cases, and to improve the effectiveness of the eye tracker for a broader spectrum of users, both of these limitations will need to be addressed. Some careful revisions to the tracking algorithm may help to overcome both of these flaws.

It was also interesting to note that, based on the image files recorded from subject 2, the light eye color did not appear to present a problem; on the contrary, the black-and-white images showed a clear contrast between the sclera and the iris regions. Moreover, the light skin tones of subjects 2 and 3 also did not seem to impair the efficacy of the eye tracker. After the necessary adjustments were made to the camera settings, the images (in both cases) displayed an adequate demarcation between the skin and the sclera of the eye, allowing the iris to be detected. Finally, the images obtained from subject 4 also suggested that the glasses worn by this user did not cause any problems in distinguishing the important features.

5.2 Future Improvements

Although there were definitely some successes in the design of this eye tracking system, there is clearly a great deal of room for development to this project. An obvious improvement to the current implementation would be the inclusion of the head-movement compensation model. Using this enhancement, it would be possible to track the user's head in addition to his/her eye(s) using the eye-tracking algorithm proposed in this thesis. This would eliminate the need for a bite bar or chin rest by allowing for some small head movements, and would likely be more comfortable to the user. More notably, concerning the experiments conducted in this thesis, a head movement compensation model would remove the unreasonable expectation that a user can sit completely motionless while operating the system. This type of model should reduce the large number of errors caused by inadvertent movements.

During the implementation of the current project, as discussed in section 3.3.4, a head movement compensation model was actually attempted. The strategy aimed to track the two corners of the user's eyes, which can be assumed to stay reasonably in place (with regard to the face). As a result, by tracking the corners of the user's eyes, the movement of the user's head may also be determined. The eye corners were located by approximating the shapes of the two sclera regions as triangles and using the corresponding points on the triangles (nearest to the eye corners) as the eye corners themselves. Unfortunately, this model worked only marginally at best, so it was not used in the final version of the application; however with additional improvements to the model, it may provide a workable face tracking solution. The only disadvantage of this option is that a mobile user would be limited to a narrow range of motion. For example,

if the user moves his/her head too much, he/she will move outside of the fixed frame of the camera, thus the head-movement compensation model would not be effective in this situation. On the other hand, there are a number of webcams available on the market that have been designed to track the user's face automatically, so that when the user moves his/her head the camera may find the user's new position and adjust itself accordingly. To perform this task, many of these cameras use a built-in, feature-based tracking technique, which recognizes features such as the user's eyes and lips to surmise the position of the face (similar to the algorithm described in sections 2.3.2 and 2.3.3). In fact, the camera used for this study also includes this technology; however, in order to make use of this feature, it would have been necessary to accurately obtain the amount of rotation made by the camera each time it locates the user's face. This would have required some hacking into the webcam logic and would have likely introduced additional inaccuracies from the mechanical movement of the webcam itself. In addition, in order for the camera to obtain an image with sufficient resolution for this application, the user must be somewhat close to the camera (and with the zoom feature on the maximum setting); as a result, the face tracking mechanism generally will not work. Due to all of these factors, the face-tracking feature was not employed for this project.

Alternatively, a head-mounted camera could be used to allow for even more freedom of movement to the user. There are currently several CCD cameras that are small enough to be mounted to the head with very minimal imposition to the user, and which have sufficient resolution (640x480 pixels) to capture a very detailed image of the eye. The camera could easily be mounted to the frame of a pair of glasses with the camera positioned just below one of the user's eyes, as described in [3]. This method

was also attempted on the current software implementation with a miniature, high-resolution, black-and-white, CCD camera (Supercircuits PC224XP), but the software could not be adapted to work with the camera due to an incompatibility between the image format of the camera and that expected by the java class (javax.media.format.VideoFormat) used throughout the program designed with this thesis. However, it is possible that with additional changes to the program, a camera such as this may be used with this eye tracker application.

However, in order to take advantage of this approach, a head-mounted monitor would also be necessary to ensure that both the camera and monitor are fixed with the user's point of view. This detail may be handled by using a personal LCD screen similar to the DV920 Video Eyewear made by ICUITI, and available for approximately $550 US. The camera could be attached to the frame of this device as it would be attached to an ordinary pair of glasses, and this design would allow the user to move about freely without the concern for disrupting the accuracy of the eye tracker. The setup would be a little more costly than the proposed method (between $700 and $800 total, for the camera, a USB video capture device for the camera, if necessary, and the personal LCD screen), but it would likely be much more comfortable and flexible for the user and perhaps even more accurate, due to the increased resolution of the actual eye image and to the capacity that the head-mounted device is able to keep the system more stationary than would a chin rest.

One problem the designer of this system would have to overcome is the computation time. With the potential for higher quality images, the computation time may increase significantly, making the system virtually unusable. This problem could be

eliminated rather easily by reducing the resolution of the image (using the settings options available to the camera) to a size comparable to that of the cropped images used in this project, however that would erase any potential gain in accuracy that may be achieved by using the close-range, head-mounted camera, in the first place. Another option may be to use a more efficient, programming method to analyze the pixels of each image, possibly partitioning the large array of image pixels into several smaller arrays and then analyzing them in parallel. This will, of course, require more processing power, but with the steady increases in CPU power in home computers, this may be a viable option in the near future.

Another opportunity for improvement to the current design is in the text editor functionality. The current text editor is somewhat cumbersome because it takes three selection operations to select each letter, assuming there are no errors. Accordingly, any reduction in the time it takes to type a message would greatly improve this process. One method that may be applied is a predictive text entry algorithm like the techniques commonly used in text messaging on cell phones. By using a combination of natural language patterns and/or user-specific, commonly used words and phrases, the text editor is able to offer a prediction of the next word the user may intend to type based on the characters typed so far. In previous studies, similar methods have proven to reduce the number of key presses by as much as 25% [16, 22]. Needless to say, this would dramatically reduce the amount of time required to type a message.

In addition, it may also be beneficial to develop application user interfaces as well as web pages specifically for the purpose of eye tracking operations in mind. These pages could be designated as eye tracker compliant or compatible, and would feature

components that make the selection of links and commands more accessible to an eye tracking system. These components would include enlarged selection targets, and interfaces that can easily be undone if an incorrect option is accidentally selected. Ideally, at some point, eye tracking web pages would be added to the web, in a similar manner to the way modified WAP pages have been provided to accommodate the smaller screens of mobile phones and PDAs. In addition, (for the purpose of developing an eye tracking interface), it might be worthwhile to reserve some pages as tracking mode pages, while other pages are simply passive viewing pages. Tracking mode pages would always remain in tracking mode in order to make their use more convenient for the user. For example, a (read only) text page may allow the user to read and scroll down as given in [23] so that the cursor does not necessarily need to move with every eye movement, but may detect when the user is viewing the last line of text, and automatically scroll down. However, other screens may lend themselves better to an adjustable (on/off tracking mode) setting. For instance, an HTML page with selectable links may be more user-friendly if the user can switch in and out of tracking mode, to allow both easy reading of the page and selection of objects. By implementing “smart” pages, this would allow disabled and able-bodied users alike to navigate just as consumers have become accustomed to the use of a mouse to perform screen selections.

Chapter 6: Appendices

6.1 Appendix A – Screen Shots of Application User Interface Menus

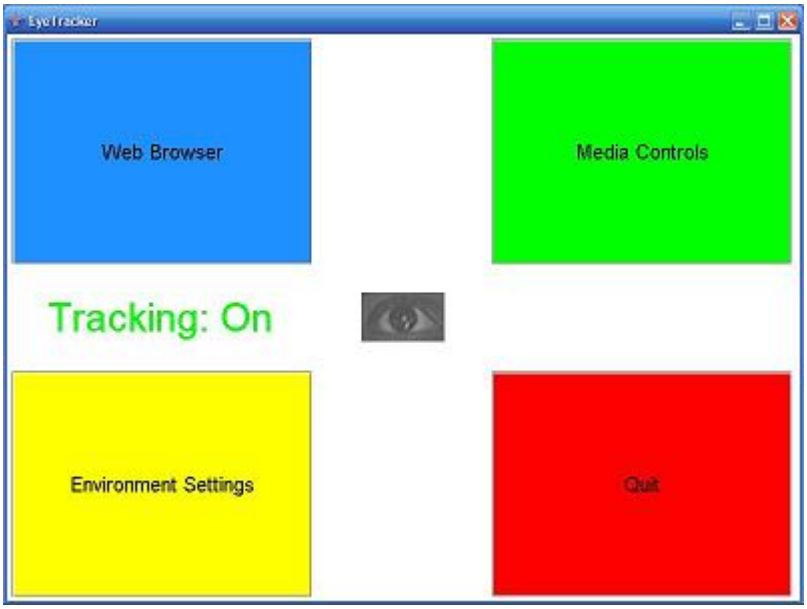

Figure 11: Main Menu

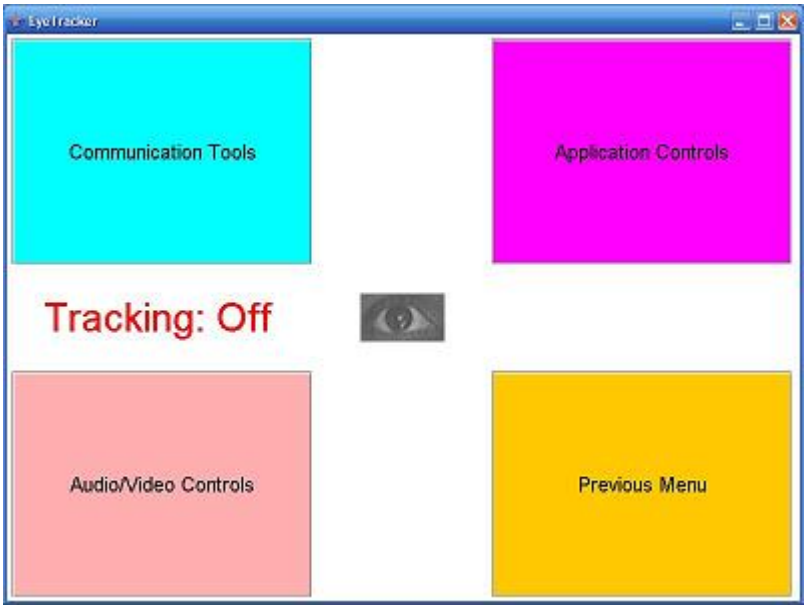

Figure 12: Media Controls Menu

Figure 13: Communication Tools Menu

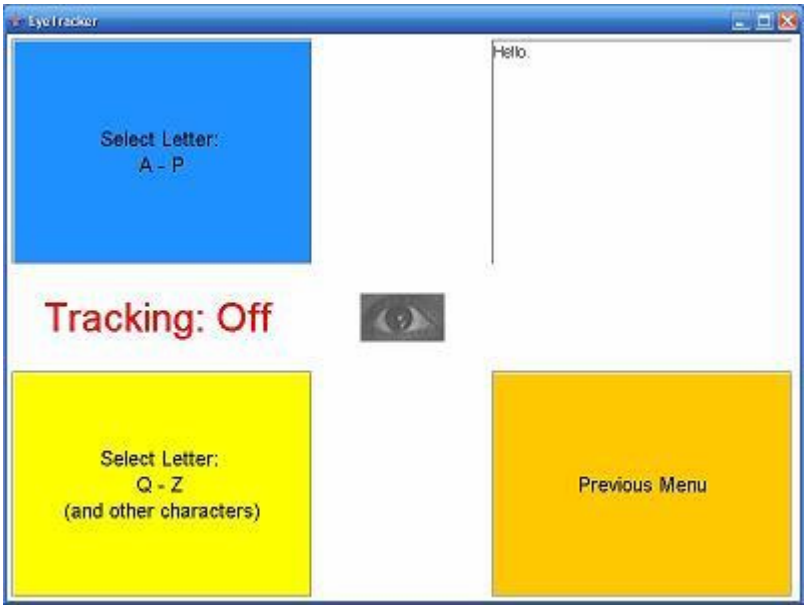

Figure 14: Text Editor Menu

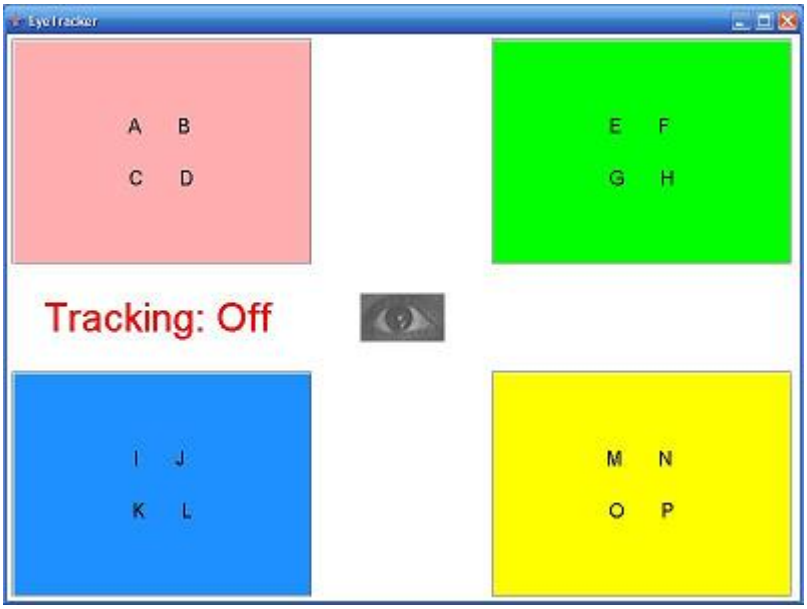

Figure 15: Letters A-P Menu

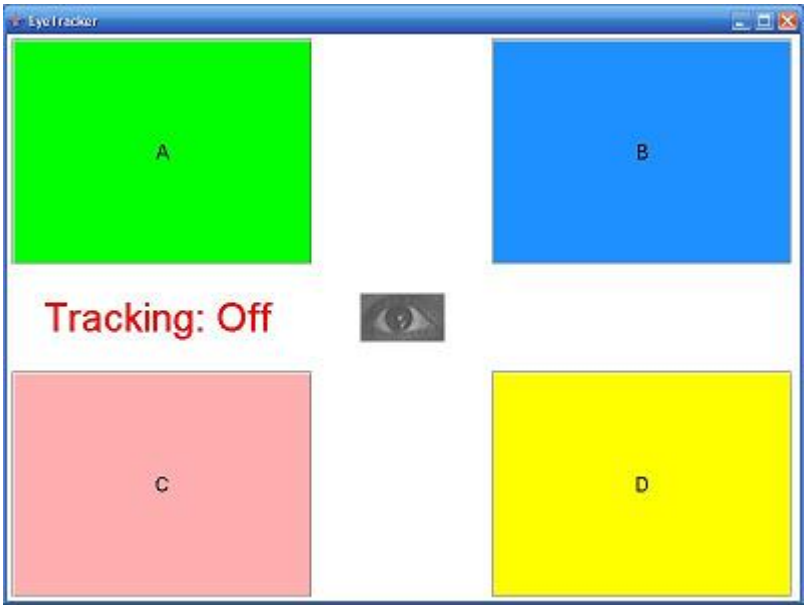

Figure 16: Letters A, B, C, D Menu

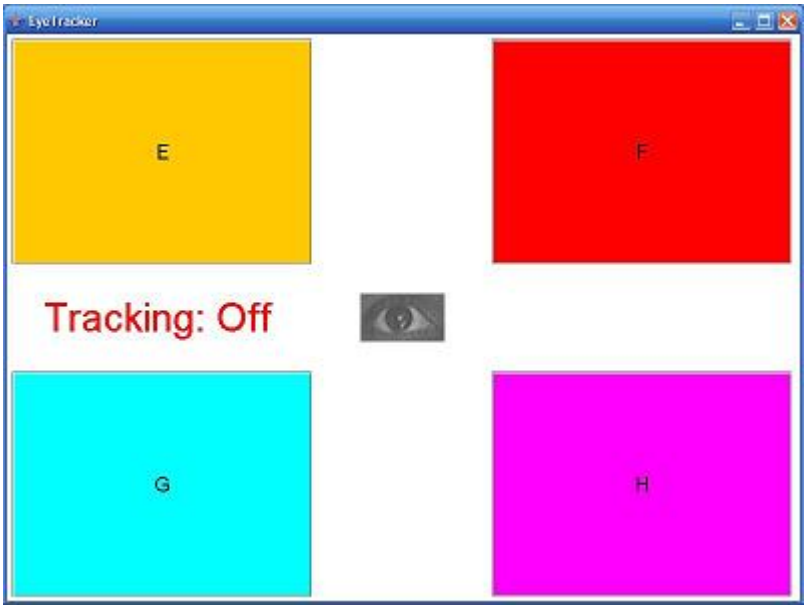

Figure 17: Letters E, F, G, H Menu

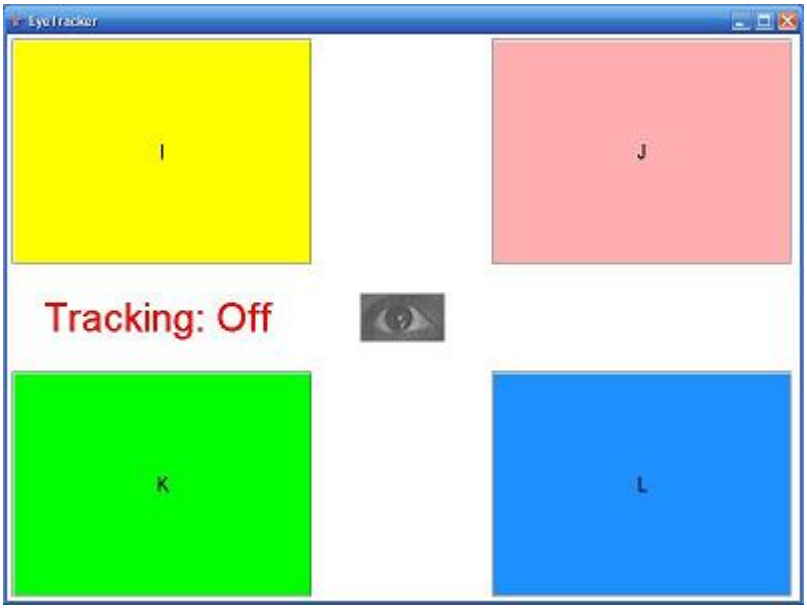

Figure 18: Letters I, J, K, L Menu

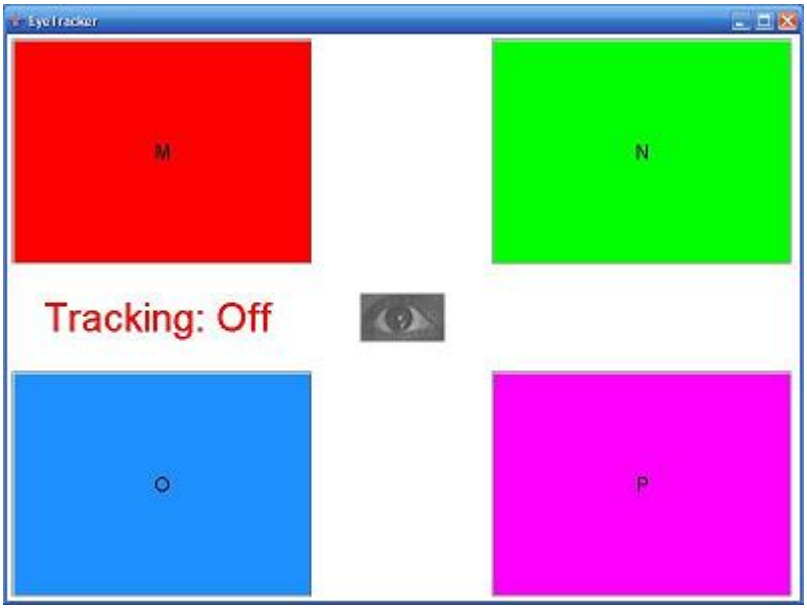

Figure 19: Letters M, N, O, P Menu

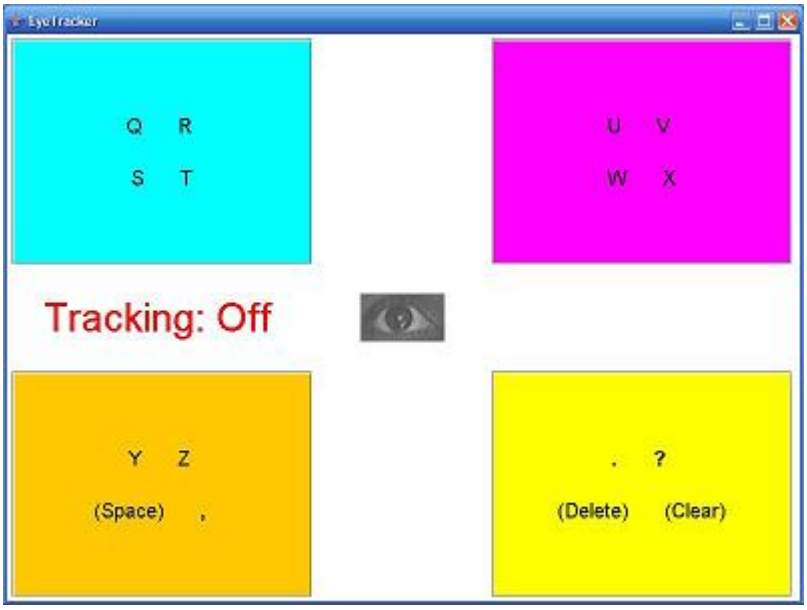

Figure 20: Letters Q-Z, and special characters Menu

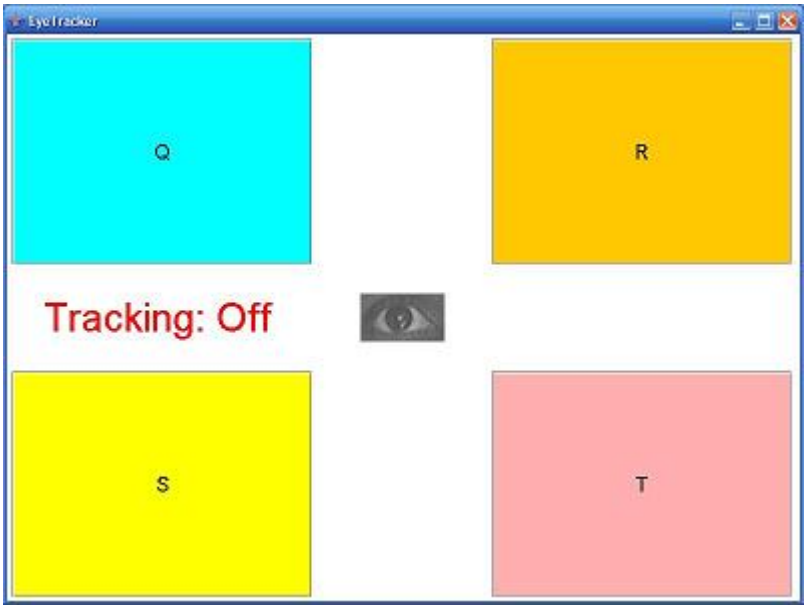

Figure 21: Letters Q, R, S, T Menu

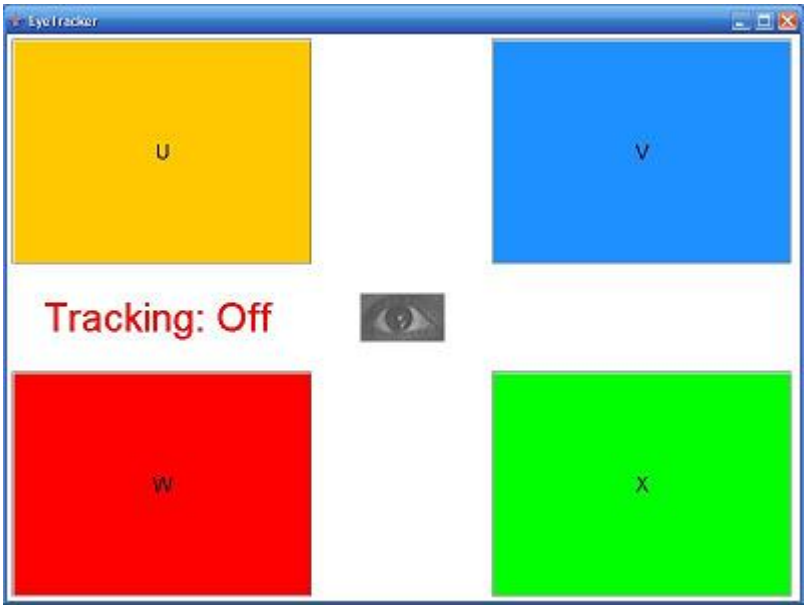

Figure 22: Letters U, V, W, X Menu

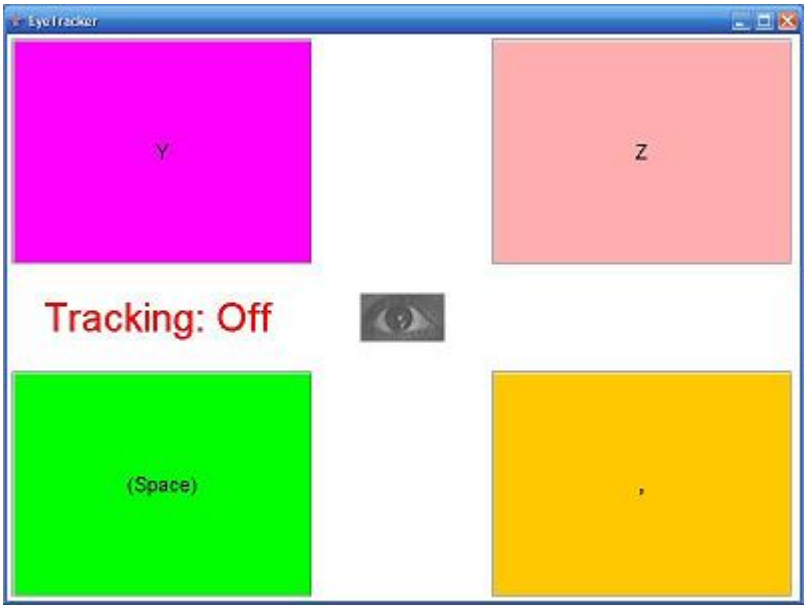

Figure 23: Letters Y and Z, and characters [Space] and ‘,’ Menu

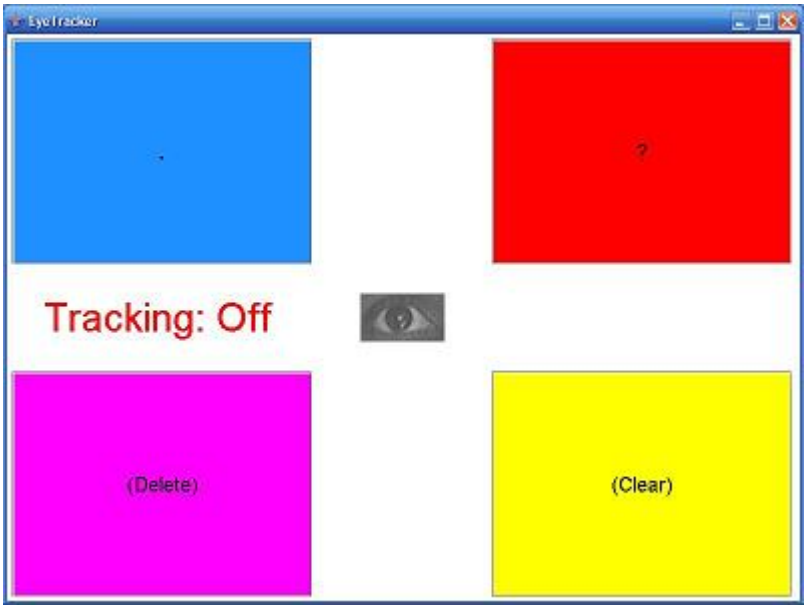

Figure 24: Characters ‘.’, ‘?’, [Delete], and [Clear] Menu

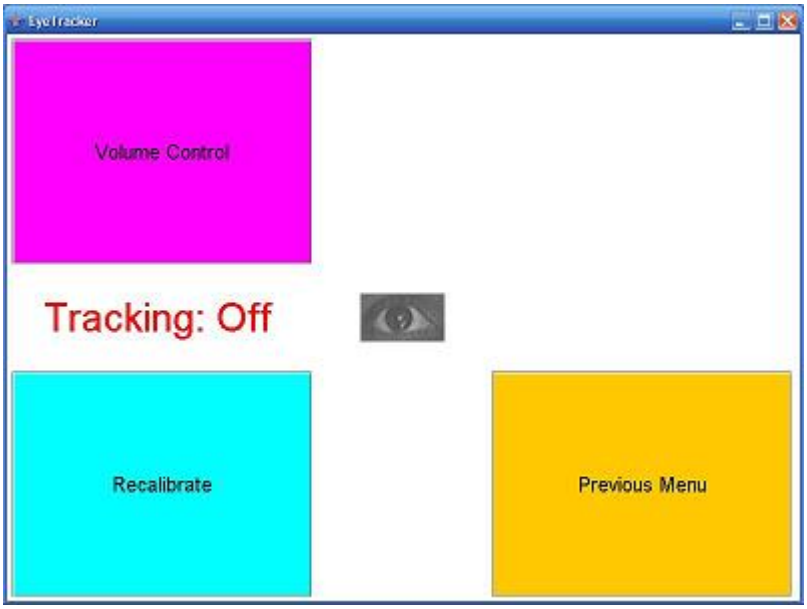

Figure 25: Application Controls Menu

Figure 26: Volume Control Menu

Figure 27: Environment Settings Menu

Figure 28: Temperature Controls Menu

Figure 29: Lighting Zones Menu

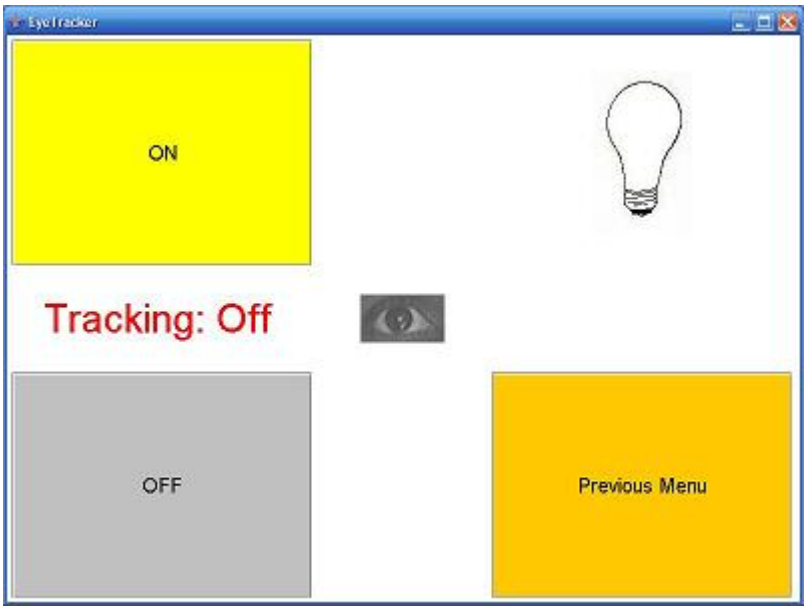

Figure 30: Zone (1, 2 and 3) Lighting Controls Menu (lights off)

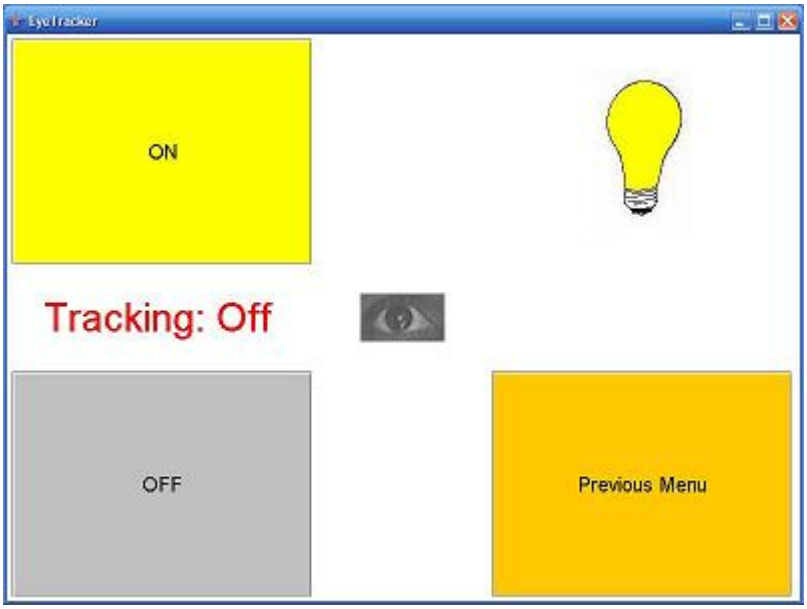

Figure 31: Zone (1, 2 or 3) Lighting Controls Menu (lights on)

6.2 Appendix B – Flow Chart Diagrams of Eye Tracking System

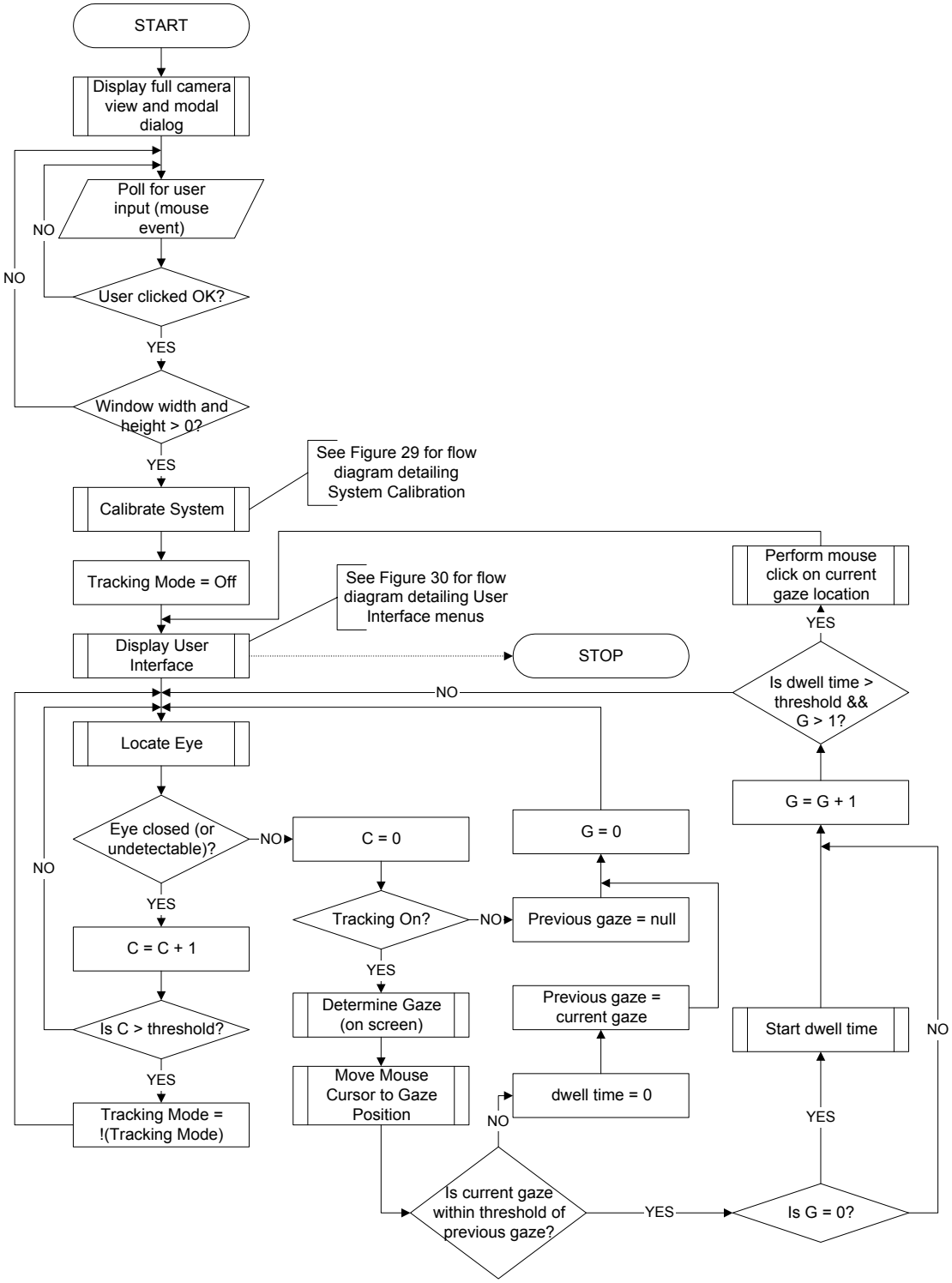

Figure 32: Basic Flow Diagram of Entire Tracking System

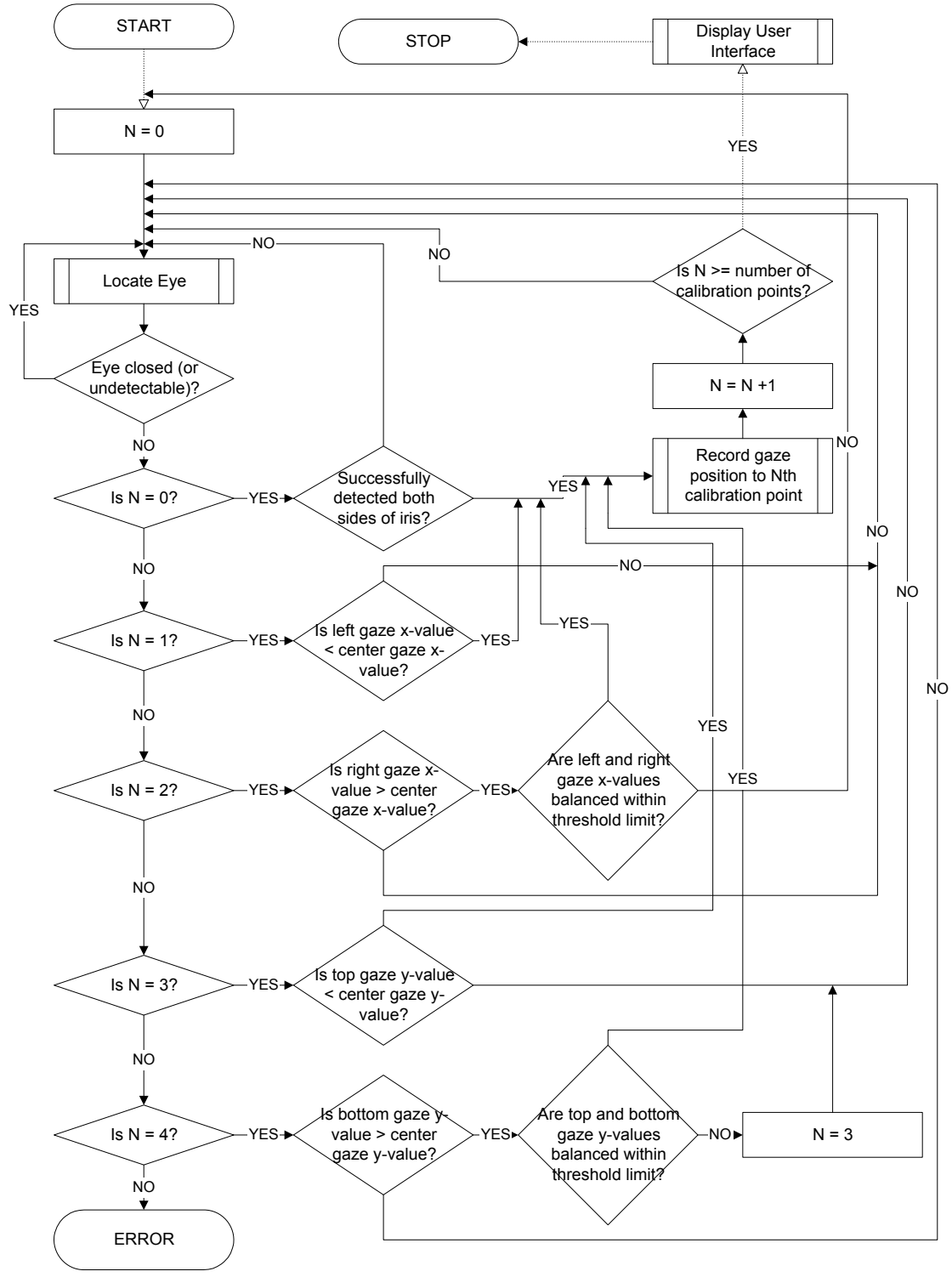

Figure 33: Flow Diagram of Calibration Process

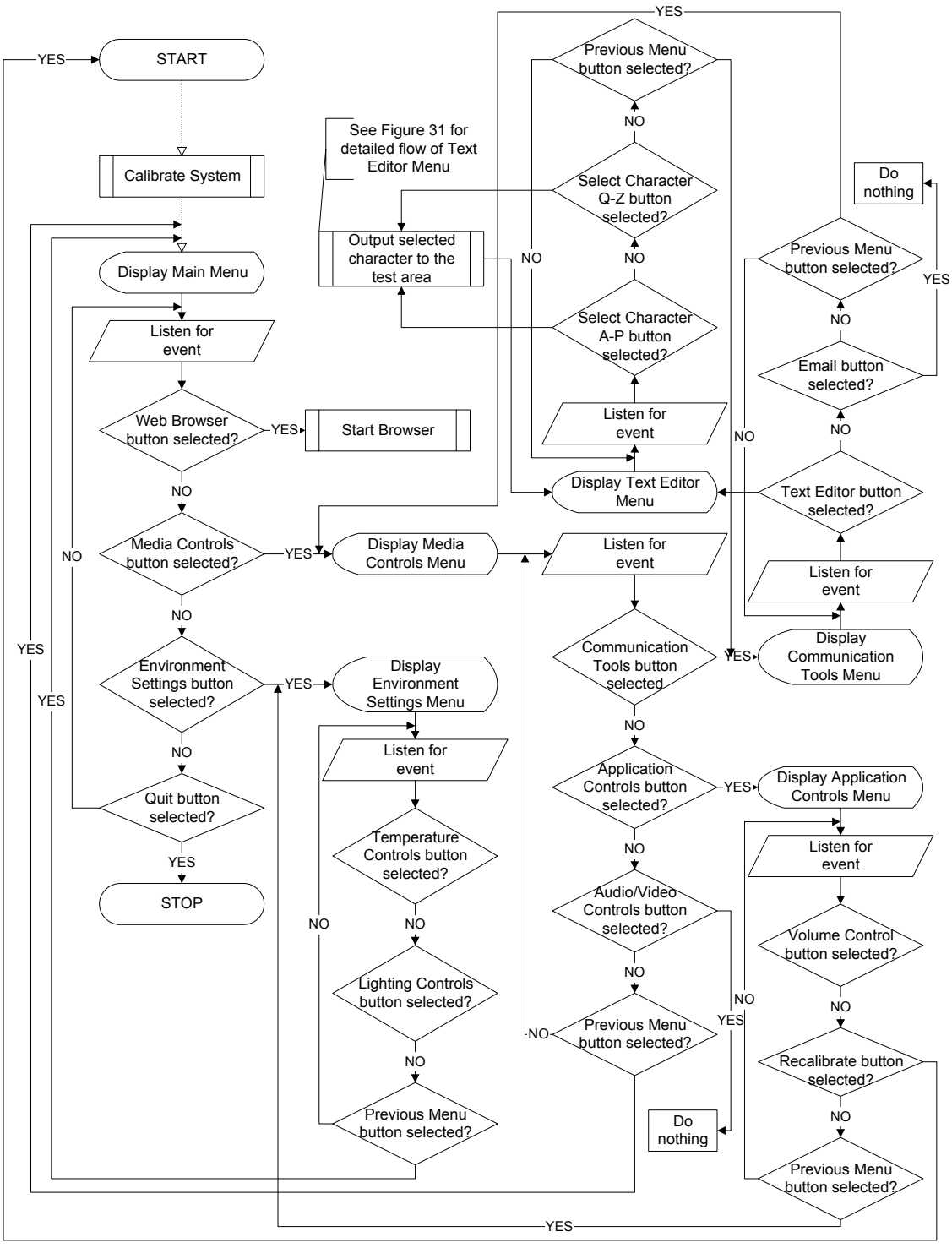

Figure 34: Flow Diagram of User Interface Menus

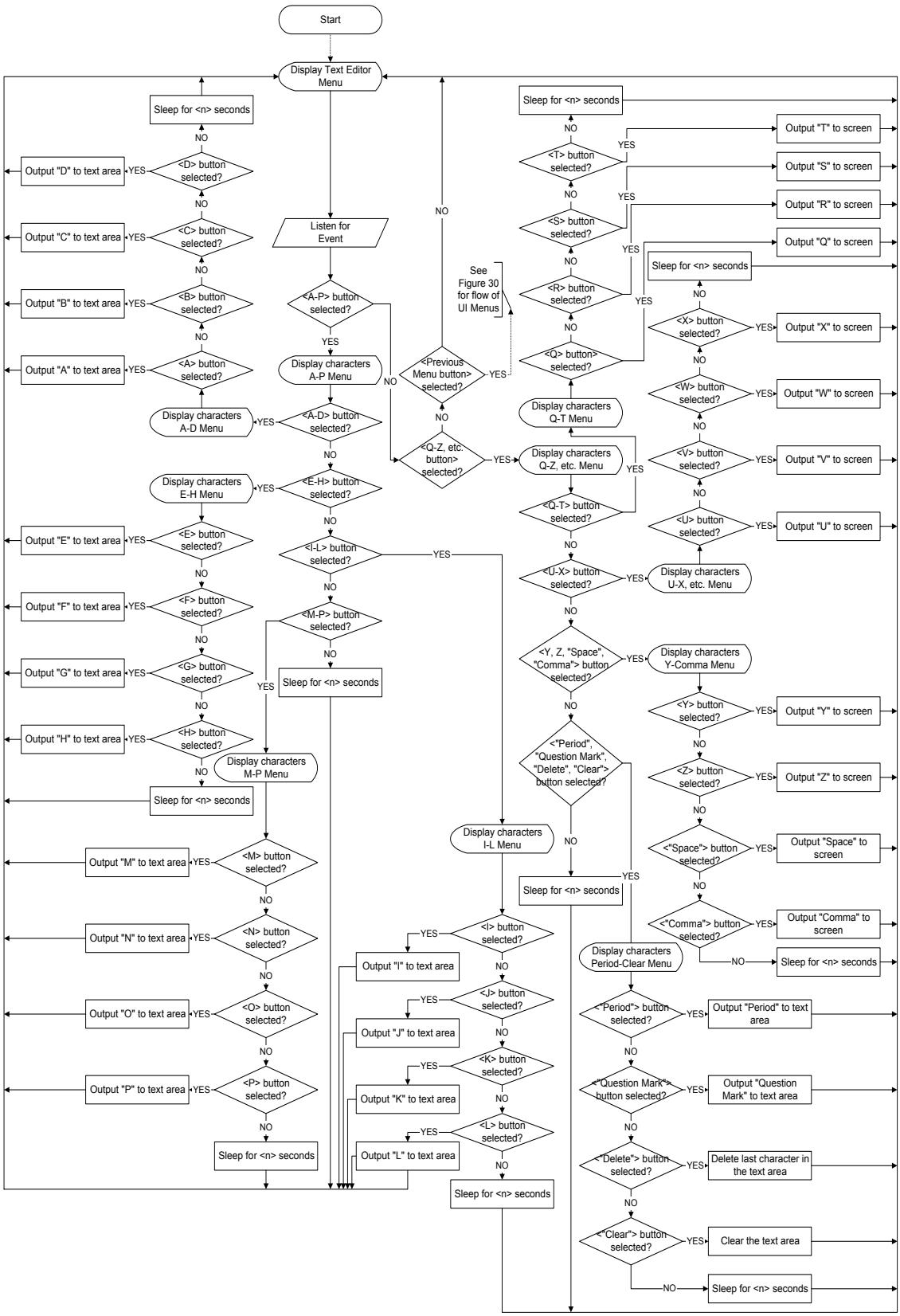

Figure 35: Detailed Flow Diagram of Text Editor Menu

Chapter 7: References

[1] Abdel-Malek, A., Bloomer, J. (1990). "Visually Optimized Image Reconstruction," In *Proceedings of the International Society for Optical Engineering (SPIE)*, vol. 1249, pp. 330-335.

[2] Amir, A., Zimet, L., Sangiovanni-Vincentelli, A., Kao, S. (2005). "An embedded system for an eye-detection system," In *Computer Vision and Image Understanding* (CVIU98), no. 1, pp. 104-123.

[3] Babcock, J. S., Pelz, J. B. (2004). "Building a Lightweight Eyetracker," In *Proceedings of the 2004 Symposium on Eye Tracking Research & Applications*, pp. 109-114.

[4] Bakman, L., Blidegn, M., Wittrup, M., Larsen, L. B., Moeslund, T. B. (1998). "Enhancing a WIMP based interface with Speech, Gaze tracking and Agents," In ICSLP-1998, paper 0766.

[5] Bagci, A. M., Ansari, R., Khokhar, A., Cetin, E. (2004). "Eye Tracking Using Markov Models," In *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3, pp. 818-821.

[6] Baluja, S., Pomerleau, D. (1994). "Non-intrusive Gaze Tracking Using Artificial Neural Networks," In *Carnegie Mellon University CS Department Technical Report* (CMU-CS-94-102).

[7] Baxter, Richard. "Laser Safety Training Manual," In http://chemistry.uchicago.edu/safety/LaserSafety.pdf, University of Chicago.

[8] Bergasa, L. M., Nuevo, J., Sotelo, M. A., Barea, R., Lopez, M. E. (2006). "Real-time System for Monitoring Driver Vigilance," In *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 63-77.

[9] Birt, J. A., Furness, T. A. III (1974). "Visually Coupled Systems," In *Air University Review*, vol. 20, no. 3, pp. 28-40.

[10] Chau, M., Betke, M. (2005). "Real Time Eye Tracking and Blink Detection with USB Cameras," Boston University Computer Science Technical Report No. 2005-12, pp. 1-10.

[11] Derwent House Company. "Eye Safety with Near Infra-red Illuminators," In http://www.derwentcctv.com/datasheets/safety.pdf.

[12] Eaddy, M., Blasko, G., Babcock, J., Feiner, S. (2004). "My Own Private Kiosk: Privacy-Preserving Public Displays," In *Eighth International Symposium on*

Wearable Computers, 2004, vol. 1, pp. 132-135.

[13] Engell-Nielsen, T., Glenstrup, A. J., Hansen, J. P. (1995). "Eye-Gaze Interaction: A New Media - not just a Fast Mouse," In *Handbook of Human Factors/Ergonomics*, Asahura Publishing Co., Tokyo, Japan, pp. 445-455.

[14] Ewing, K. (2005). "Studying web pages using Eye Tracking," A Whitepaper sponsored by Tobii Technology, pp. 1-13.

[15] Gerr, F., Marcus, M., Ensor, C., Kleinbaum, D., Cohen, S., Edwards, A., Gentry, E., Ortiz, D., Monteilh, C. (2002). "A Prospective Study of Computer Users: I. Study Design and Incidence of Musculoskeletal Symptons and Disorders," In *American Journal of Industrial Medicine 41*, pp. 221-235.

[16] Gong, J., Tarasewich, P., Hafner, C. D., MacKenzie, S. I. (2007). "Improving Dictionary-Based Disambiguation Text Entry Method Accuracy," In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems* (CHI '07), pp. 2387-2392.

[17] Hansen, D. W., Hansen J. P., Nielsen, M., Johansen, A. S. (2002). "Eye Typing using Markov and Active Appearance Models," In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, pp. 132-136.

[18] Hansen, D. W., MacKay, D., Nielsen, M., Hansen, J. P. (2004). "Eye Tracking Off the Shelf," In *Proceedings of the Symposium on Eye Tracking Research and Applications* (ETRA 2004), ACM Press, p. 58.

[19] Hansen, J. P., Hansen, D. W., Johansen, A. S. (2002). "Bringing Gaze-based Interaction Back to Basics," In *Proceedings of the Symposium on Eye Tracking Research & Applications*, pp. 15-22.

[20] Hansen, J. P., Torning, K., Johansen, A. S., Itoh, K., Aoki, H. (2004). "Gaze Typing Compared with Input by Head and Hand," In *Proceedings of the Eye Tracking Research & Applications Symposium* (ETRA 04), pp. 131-138.

[21] Hansen, J. P., Johansen, A. S., Hansen, D. W., Itoh, K., Mashino, S. (2003). "Command Without a Click: Dwell Time Typing by Mouse and Gaze Selections," In *Proceedings of INTERACT 2003*, IOS Press, pp. 121-128.

[22] How, Y., Kan, M.-Y. (2005). "Optimizing predictive text entry for short message service on mobile phones," In *Proceedings of Human Computer Interfaces International* (HCII 05).

[23] Jacob, R. J. K. (1990). "What You Look At is What You Get: Eye Movement-Based Interaction Techniques," In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 11-18.

[24] Jacob, R. J. K. (1991). "The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What You Get," In *ACM Transactions on Information Systems*, vol. 9, no. 3, pp. 152-169.

[25] Jacob, R. J. K. (1994). "New Human-Computer Interaction Techniques," In Brouwer-Janse, M. & Harrington, T. (Eds.), *Human Machine Communication for Educational Systems Design*, pp. 131-138.

[26] Jacob, R. J. K., Karn, K. S. (2003). "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises" (Section commentary), In J. Hyona, R. Radach, & H. Deubel (Eds.), *The Mind's Eyes: Cognitive and Applied Aspects of Eye Movements*, pp.573-605.

[27] Ji, Q., Zhu, Z., Lan, P. (2004). "Real-time Non-intrusive Monitoring and Prediction of Driver Fatigue," In *IEEE Transactions on Vehicular Technology*, vol. 53, no. 4, pp. 1052-1068.

[28] Ko, J.-G., Kim, K.-M., Ramakrishna, R. S. (1999). "Facial Feature Tracking for Eye-Head Controlled Human Computer Interface," In *Proceedings of the IEEE Region 10 Conference*, vol. 1, pp. 72-75.

[29] Li, D., Babcock, J. S., Parkhurst, D. J. (2006). "openEyes: a low-cost head-mounted eye-tracking solution," In *Proceedings of the Eye Tracking Research & Applications Symposium* (ETRA 06), pp. 95 – 100.

[30] Li, D., Winfield D., Parkhurst, D. J. (2005). "Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches," In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR05), vol. 3, p. 79.

[31] Manhartsberger, M., Zellhofer, N. (2005). "Eye tracking in usability research: What users really see," In *Usability Symposium 2005*, vol. 198, pp. 141-152.

[32] Marxhausen, P. (2005). In Univ. of Nebraska-Lincoln / Electronics Shop RSI Web Page, http://eeshop.unl.edu/rsi.html.

[33] Micheli-Tzanakou, E. (2000). Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence, CRC Press Inc., Boca Raton, FL, USA.

[34] Murphy, R. A., Basili, A. (1993). "Developing the User System Interface for a Communications System for ALS Patients: The Results," In *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting, p. 1034*.

[35] National Institute of Neurological Disorders and Stroke (2006). Amyotrophic

Lateral Sclerosis Fact Sheet, In http://www.ninds.nih.gov/disorders/amyotrophiclateralsclerosis/detail_amyotrophiclateralsclerosis.htm.

[36] Parkhurst, D., Culurciello, E., Niebur, E. (2000). "Evaluating Variable Resolution Displays with Visual Search: Task Performance and Eye Movements," In *Proceedings of the 2000 Symposium on Eye Tracking Research and Applications*, pp. 105-109.

[37] Phillips, G. N., Bibles, L. D., Currie, D. E., Topperwien, C. L. (1985). "Pneumatic Pressure Switch for Communication," In *Rehabilitation Engineering Society of North America*, pp. 265-267.

[38] Previty, J. (1985). "A Preliminary Design Proposal for an Eye Movement Detection System," Department of Electrical Engineering, Rutgers University, New Brunswick, NJ.

[39] Reddy, M. (1995). "Musings on Volumetric Level of Detail for Virtual Environments," In *Virtual Reality: Research, Development and Application*, vol. 1, no. 1, pp. 49-56.

[40] Richardson, D. C., Spivey, M. J. (2004). "Eye tracking: Characteristics and methods," In Wnek, G. & Bowlin, G. (Eds.) *Encyclopedia of Biomaterials and Biomedical Engineering*. New York: Marcel Dekker, Inc., pp. 568-572.

[41] Richardson, D. C., Spivey, M. J. (2004). "Eye-Tracking: Research Areas and Applications," In Wnek, G. & Bowlin, G. (Eds.) *Encyclopedia of Biomaterials and Biomedical Engineering*. New York: Marcel Dekker, Inc., pp. 573-582.

[42] Rolland, J. P., Yoshida, A., Davis, L. D., Reif, J. H. (1998). "High-resolution Inset Head-mounted Display," In *Applied Optics 1998*, vol. 37, no. 19, pp. 4183-4193.

[43] Salvucci, D. D. (2000). "An Interactive Model-Based Environment for Eye-Movement Protocol Analysis and Visualization," In *Proceedings of the Eye Tracking Research and Applications Symposium*, pp. 57-63.

[44] Scherffig, L. (2005). "It's in Your Eyes: Gaze Based Image Retrieval in Context," Edited by Hans H. Diebner, Institute for Basic Research, Karlsruhe.

[45] Sharan, D. In Repetitive Strain Injuries information page, http://www.deepaksharan.com/cri_intro.html.

[46] Sibert, L. E., Jacob, R. J. K. (2000). "Evaluation of Eye Gaze Interaction," In *Proceedings of the SIGCHI conference on Human factors in Computing Systems*, pp. 281-288.

[47] Smith, P., Shah, M., da Vitoria Lobo, N. (2003). "Determining Driver Visual Attention with One Camera," In *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 4, pp. 205-218.

[48] Starker, I., Bolt, R. A. (1990). "A Gaze-responsive Self-disclosing Display," In Proceedings of the Conference on Human Factors in Computing Systems (CHI '90), pp. 3-9.

[49] Stiehl, W. D., Lieberman, J., Breazeal, C., Basel, L., Lalla, L., Wolf, M. (2005). "The Design of the Huggable: A Therapeutic Robotic Companion for Relational, Affective Touch," MIT Media Lab, Massachusetts Institute of Technology, Cambridge, MA.

[50] Tan, K.-H., Kriegman, D. J., Ahuja N. (2002). "Appearance-based Eye Gaze Estimation," In *Proceedings of the Sixth IEEE Workshop on Applications of Computer Vision*, pp. 191-195.

[51] Witner Hansen, D., Satria, R., Sorensen, J., Hammoud, R. (2005). "Improved Likelihood Function in Particle-based IR Eye Tracking," In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR05), vol. 3, p. 5.

[52] Xu, L.-Q., Machin, D., Sheppard, P. (1998). "A Novel Approach to Real-time Non-intrusive Gaze Finding," In *British Machine Vision Conference*, 1998.

[53] Yamato, M., Monden, A., Matsumoto, K., Inoue, K., Torri, K. (2000). "Button Selection for General GUIs Using Eye and Hand Together," In *Proceedings of the Working Conference of Advanced Visual Interfaces*, pp. 270-273.

[54] Zhai, S., Morimoto, C., Ihde, S. (1999). "Manual and Gaze Input Cascaded (Magic) Pointing," In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI'99), pp. 246-253.

[55] Zhu, Z., Ji, Q. (2005). "Eye Gaze Tracking Under Natural Head Movements," In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR05), pp. 918-923.