GAZE-ENHANCED USER INTERFACE DESIGN

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Manu Kumar

May 2007

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____
(Terry Winograd) Principal Advisor

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____
(Scott Klemmer)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

_____
(Shumin Zhai)

Approved for the University Committee on Graduate Studies.

Abstract

The eyes are a rich source of information for gathering context in our everyday lives. A user's gaze is postulated to be the best proxy for attention or intention. Using gaze information as a form of input can enable a computer system to gain more contextual information about the user's task, which in turn can be leveraged to design interfaces which are more intuitive and intelligent. Eye gaze tracking as a form of input was primarily developed for users who are unable to make normal use of a keyboard and pointing device. However, with the increasing accuracy and decreasing cost of eye gaze tracking systems it will soon be practical for able-bodied users to use gaze as a form of input in addition to keyboard and mouse. This dissertation explores how gaze information can be effectively used as an augmented input in addition to traditional input devices.

The focus of this research is to augment rather than replace existing interaction techniques. Adding gaze information provides viable alternatives to traditional interaction techniques, which users may prefer to use depending upon their abilities, tasks and preferences. This dissertation presents a series of novel prototypes that explore the use of gaze as an augmented input to perform everyday computing tasks. In particular, it explores the use of gaze-based input for pointing and selection, scrolling and document navigation, application switching, password entry, zooming and other applications. It presents the results of user experiments which compare the gaze-augmented interaction techniques with traditional mechanisms and show that the resulting interaction is either comparable to or an improvement over existing input methods. These results show that it is indeed possible to devise novel interaction techniques that use gaze as a form of input without overloading the visual channel and while minimizing false activations.

The dissertation also discusses some of the problems and challenges of using gaze information as a form of input and proposes solutions which, as discovered over the course of the research, can be used to mitigate these issues. Finally, it concludes with an analysis of technology and economic trends which make it likely for eye tracking systems to be produced at a low enough cost, that when combined with the right interaction techniques, they would create the environment necessary for gaze-augmented input devices to become mass-market.

The eyes are one of the most expressive features of the human body for non-verbal, implicit communication. The design of interaction techniques which use gaze-information to provide additional context and information to computing systems has the potential to improve traditional forms of human-computer interaction. This dissertation provides the first steps in that direction.

Acknowledgments

I would like to thank my advisor Terry Winograd for letting me chose my own path for my research and most importantly for guiding me along the way and educating me in the ways of academia and academic research. Terry was always very encouraging and helped me to look at problems from different perspectives. Working with Terry has been a wonderful experience and I could not have hoped for a better advisor for my Ph.D.

Andreas Paepcke was always ready and willing to roll up his sleeves and help with designing studies and analyzing data. Andreas taught me all I know about statistical analysis and helped to make my work better in many ways. Scott Klemmer has truly brought new energy and perspectives to the HCI group and his presence has increased the quantity and the quality of the work being done at Stanford. I am also grateful to Pat Hanrahan and Brian Wandell for serving on my orals committee and to Dan Boneh for the opportunity to co-author a paper with him.

Shumin Zhai, David Beymer and Arnon Amir from IBM Almaden Research have been a constant source of encouragement and served as a sounding board providing advice and feedback at critical junctures of this research. I have truly enjoyed our discussions on both eye tracking technology and applications.

Shumin Zhai and Rob Jacob wrote the seminal papers that became the foundation of my work and I thank them both for their contribution to the field and for their interaction with me. I would like to recognize Andrew Duchowski and Roel Vertegaal for their contribution to the field and for organizing the Eye-Tracking Research and Application symposium, where I met several outstanding individuals in the field of eye-tracking.

Bob Dougherty helped me in the initial prototyping phases of attempting to build my own eye tracker. The financial support of Stanford Media X and the School of Engineering Equipment Matching grant was instrumental in my ability to have an eye tracker on my desk and also to fund the final year of my Ph.D. I would especially like to acknowledge Ellen Levy and Kathy Lung for their support.

Colin Johnson and Greg Edwards of EyeTools, Inc. were always willing to help and share information. Nico Vroom of Tobii Technology, AB provided a loaner eye tracker when our unit needed to be returned to Sweden for repair and helped to keep my research on schedule.

Sergei Vassilvitskii helped me analyze and wrap my head around the pros and cons of various algorithms. Samuel Ieong saved me countless hours by showing me how to program the magic macros and tricks in Excel to analyze the data from user studies. Michael Bernstein shared the code for his Master's thesis with me, which served as an example for me to get up to speed on C#. Rohan Puranik joined as an undergraduate research assistant and assisted with prototyping and data analysis.

I would also like to thank Ron Yeh for the early brainstorming sessions in which I was able to share my crazy thoughts of wanting to control things with my eyes. Ron's willingness to tolerate my harebrained ideas and engage in stimulating discussions provided the encouragement for me to actually consider giving them a shot. Thank you also to Taemie Kim, one of the best research partners I have worked with, for her crucial contribution to our debut short-paper at CHI on the Dynamic Speedometer; my current and past office mates Ian Buck, Doantam Phan and Bryan Chan for putting up with me for many years; Jeff Klingner, who was always willing to help me brainstorm solutions to problems as I walked up and down the hall; Leith Abdulla for providing the nourishment (aka junk food) and the engaging conversation to help in procrastination and thinking; Bjoern Hartmann and Dan Maynes-Aminzade, who were willing to assist and share their endless creativity; David Akers, Brian Lee, Heidy Maldonado, Wendy Ju, Merrie Ringel Morris, Kayvon Fatahalian, Mike Houston, Joel Brandt and the many other members of the Gates 3B HCI and Graphics family who have always been very supportive.

Thank you to Heather Gentner and Ada Glucksman — without whom things would never run so smoothly. They shield us from the immense bureaucracy that a large university is and help ensure we can get things done. John Gerth's emails at odd hours of the night are testament to the hard work it takes to secure our computing resources.

A special thanks to Ren Ng – who on one hand made it more of a challenge for me to finish my dissertation by providing the distraction of a cool new startup — something I tried hard to stay away from for years, since I knew I would get distracted! In all fairness, he also provided the motivation for me to finish. Interacting with Ren over the past year and a half has made me realize that my true passion lies in the world of startups and entrepreneurship and I am looking forward to returning to that passion after completing the Ph.D.

I would also like to thank my parents who have always afforded me the freedom to follow my own path. I know that they miss me very much and would rather have me home, but they still encourage me to do what I want; my sister, who by being available at home enables me to stay in the United States and my grandmother, Pushpa Guglani, who was a remarkable lady till the very end. She encouraged me to think free and to follow my dreams. I dedicate this Ph.D. to her for she would have been very proud.

Most of all thank you to my fiancée, Hana Konfrštová, who has put up with the endless hours I have spent in front of the computer and my never ending requests for her to massage my arms which were sore from RSI. We've both had an incredibly busy year, filled with several critical milestones, and it would not have been possible to make it through it without each other's support.

Contents

List of Tables

List of Illustrations

1 Introduction

The eyes are a rich source of information for gathering context in our everyday lives. We use our eyes to determine who, what, or where in our daily communication. A user's gaze is postulated to be the best proxy for attention or intention [116]. Using eye-gaze information as a form of input can enable a computer system to gain more contextual information about the user's task, which in turn can be leveraged to design interfaces which are more intuitive and intelligent.

Eye gaze tracking as a form of input was primarily developed for users who are unable to make normal use of a keyboard and pointing device. However, with the increasing accuracy and decreasing cost of eye gaze tracking systems it will soon be practical for able-bodied users to use gaze as a form of input in addition to keyboard and mouse – provided the resulting interaction is an improvement over current techniques. This dissertation explores how gaze information can be effectively used as an augmented input in addition to traditional input devices.

The focus of this research is to augment rather than replace existing interaction techniques. Adding gaze information provides viable alternatives to traditional interaction techniques, which users may prefer to use depending upon their abilities, tasks and preferences. This dissertation presents a series of novel prototypes that explore the use of gaze as an augmented input to perform everyday computing tasks. In particular, it explores the use of gaze-based input for pointing and selection, scrolling and document navigation, application switching, password entry, zooming and other applications. It presents the results of user experiments which compare the gaze-augmented interaction techniques with traditional mechanisms and show that the resulting interaction is either comparable to or an

Figure 1. Logo for the Gaze-enhanced User Interface Design (GUIDe) research project.

improvement over existing input methods. These results show that it is indeed possible to devise novel interaction techniques that use gaze as a form of input without overloading the visual channel and while minimizing false activations.

The dissertation also discusses some of the problems and challenges of using gaze information as a form of input and proposes solutions which, as discovered over the course of the research, can be used to mitigate these issues. Finally, it concludes with an analysis of technology and economic trends which make it likely for eye tracking systems to be produced at a low enough cost, that when combined with the right interaction techniques, they would create the environment necessary for gaze-augmented input devices to become mass-market.

The eyes are one of the most expressive features of the human body for non-verbal, implicit communication. The design of interaction techniques which use gaze-information to provide additional context and information to computing systems has

the potential to improve traditional forms of human-computer interaction. This dissertation provides the first steps in that direction.

1.1 Thesis Statement

The keyboard and mouse have long been the dominant forms of input. Contemporary computer systems are still plagued by the asymmetrical bandwidth problem [52], where the bandwidth from the computer to the user is far greater than the bandwidth from the user to the computer.

In this dissertation we build upon the insight presented by Jacob in [52] to investigate the possibility of introducing the movements of a user's eyes as an additional input medium. We posit that gaze information, i.e. information about what the user is looking at, can be used as a practical form of input i.e. a way of communicating information from the user to the computer. The thesis statement of this work is:

"Gaze information can be used as a practical form of input."

In this research, we explore the design space of interaction techniques that use gaze information for everyday computing tasks. While some of the interaction techniques presented have the potential to supplant traditional input devices such as the mouse (Chapter 3), our goal is not to replace traditional input devices but to provide viable alternatives which users may choose to use depending upon their tasks abilities and preferences. Other sections of this dissertation explore augmenting existing interaction techniques with eye gaze. In particular, using gaze in conjunction with the keyboard and the mouse to design effective interaction techniques.

We chose the realm of desktop interactions, since they are broadly applicable to all types of computer users. In addition, the technology for desktop eye tracking systems has improved sufficiently to make it a viable input modality. The cost of these systems remains an issue, but current technology and economic trends indicate that low cost eye tracking should be possible in the near future.

1.2 Contributions

This dissertation presents a series of novel prototypes we built and experiments we conducted as a basis for the formulation of design guidelines for improving the usability and utility of gaze-based interaction techniques. The major contributions presented in this thesis are:

Gaze-based interaction techniques: We present several novel interaction techniques which explore the use of gaze as an augmented input to perform everyday computing tasks. In particular, we explore the use of gaze-based input for pointing and selection, scrolling and document navigation, application switching, password entry, zooming and other applications. We present the results of user experiments which compare the gaze-based interaction techniques with traditional mechanisms and show that the resulting interaction is either comparable to or an improvement over existing input methods. These results show that it is indeed possible to devise novel interaction techniques that use gaze as a form of input without overloading the visual channel and while minimizing false activations.

Technologies for gaze input: We discuss some of the problems and challenges of using gaze information as a form of input and propose solutions which, as discovered over the course of the research, can be used to mitigate these issues. In particular, we present techniques for filtering and smoothing gaze data, improving eye-hand coordination for gaze plus trigger activated interaction techniques and the providing focus points to help improve the accuracy of eye tracking and the user experience for using gaze-based interaction techniques. This dissertation also introduces some ideas for improving eye tracking technology and systems.

Design guidelines for gaze-based interaction: Based on our experiences in designing, implementing, and evaluating gaze-based interaction techniques, we identify key design challenges for supporting effective gaze-based interaction. We formulate design guidelines relating to these challenge areas, including appropriate uses for gaze-based interaction.

1.3 Dissertation Roadmap

The remainder of this dissertation is organized as follows:

In Chapter 2, we discuss the motivation for this work and provide some background information including the history of eye tracking, how state of the art eye trackers work and the challenges for using gaze as a form of input.

Chapters 3-8 present the gaze-based interaction techniques developed as part of this dissertation. Each chapter provides a self-contained section on using eye–gaze for a particular task.

Chapter 3, on pointing and selection, describes the design, evolution and evaluation of a new pointing technique which uses a combination of eye gaze and keyboard.

Chapter 4, on gaze-enhanced scrolling techniques, presents several different techniques for gaze-based scrolling including augmenting manual scrolling techniques with gaze information and automatic scrolling techniques, which control the onset and speed of scrolling based on the user's gaze and the use of off-screen targets for gaze-based document navigation and control.

Chapter 5, on application switching, describes the design and evaluation of a gaze-based technique for switching between applications. This technique extends Apple's concept of Exposé by using gaze-based selection of the desired application window rather than clicking on it with a mouse.

In Chapter 6, we discuss the use of a gaze-based password entry to reduce the risks of shoulder surfing.

Chapter 7, on zooming, presents the results of our attempts to implement gaze-contingent semantic zooming and explains why the obvious implementations of such a system fail to work.

Chapter 8 discusses a number of other smaller gaze-based applications, which have interesting uses, but are too small to merit a chapter for themselves. We also introduce new ideas for gaze-based interfaces in this section.

Chapter 9, on improving gaze input, discusses some of the challenges for using gaze-input and presents solutions to these challenges. In particular it presents a saccade detection and fixation smoothing algorithm, discusses approaches

to mitigate eye-hand coordination problems when using a combination of gaze plus trigger based input and discusses the use of focus points to help focus the users gaze and improve the user experience for gaze-based interaction.

In Chapter 10, we present a discussion on why current eye tracking systems are prohibitively expensive for mass-market use and propose technology and business model changes to enable the emergence of low-cost, mass-market eye trackers. We conclude with a summary of the design challenges for gaze input and a corresponding set of design guidelines which help to mitigate these challenges.

2 Background

This chapter presents the background material and related work relevant to this dissertation. It specifically looks at the motivation behind our research, the history of eye tracking, current state of the art in eye tracking and the challenges for using gaze as a form of input. It should be noted that we do not present a detailed analysis of related work in this section. Related work that is relevant to each interaction technique is presented at the beginning of the corresponding chapter.

2.1 Motivation

Computers have become an integral component of our lives. Whether at work, home or anywhere in between, we spend increasing amounts of time with computers or computing devices. Computers have not been around for a very long time – the origins of the Personal Computer can be traced back to the early 1980s when Xerox, IBM and Apple introduced their respective personal computers. However, even in this short time span increasing amounts of repetitive strain injuries (RSI) [7, 33] have emerged from overuse of the keyboard and mouse.

Repetitive strain injuries develop over periods of long and continuous overuse often extending over several years. The surge in computer-related RSI amongst technology professionals has been recognized in recent years. As more and more professions adopt computers as a primary tool, the number of cases of repetitive strain injuries is expected to increase dramatically. While the keyboard and mouse both contribute to computer-related RSI, most people suffering from RSI find that mouse use causes more strain and pain than using the keyboard [86] (Figure 2).

Figure 2. Tendonitis: a form of repetitive strain injury (RSI) caused by excessive use of the keyboard and particularly the mouse.

This impending epidemic of computer-related repetitive strain injuries coupled with the author's personal desire to develop new forms of interaction which would help to alleviate some of the stress and pain of RSI became one of the key motivators for exploring alternative forms of input for computer systems.

Alternative input modalities such as speech, which do not rely solely on the use of the hands, have been in use for a long time. However, while speech recognition may be suitable for some tasks, it is not a silver bullet for all tasks. In particular, using speech for a pointing task does not provide provides users with much useful functionality [85]. In addition, the accuracy, privacy, and social issues surrounding the use of speech interfaces make them less than optimal for use in everyday computing scenarios.

For our research, we chose to investigate the possibility of using a more subtle form of input — eye gaze.

2.2 Gaze as a Form of Input

Jacob [53] and Zhai [116] present an overview of why one would want to use eye movements for interactive input. We synthesize some of their comments in the list below:

- The eyes are a fast, convenient, high bandwidth source of information. Eye movements have been shown to be very fast and very precise.

- The eyes require no training – it is natural for the users to look at the object of interest. In other words, the control-display relationship is already well established in the brain.
- A user's eye gaze serves as an effective proxy for his or her attention and intention. Since we typically look at what we are interested in or look before we perform an action, eye gaze is the best non-invasive indicator for our attention and intention. In fact the problem of lack of eye-contact in video conferencing [29] shows just how much humans perceive by observing the eyes of others.
- The eyes provide the context within which our actions take place.
- The eyes and the hands work well in coordination.

Jacob in his 1990 paper *What You Look at Is What you Get* [52] introduced several gaze-based interaction techniques for *object selection, continuous attribute display, moving an object, eye controlled scrolling text, menu commands* and *listener window.* In this paper Jacob states that *"what is needed is appropriate interaction techniques that incorporate eye movements into the user-computer dialog in a convenient and natural way."* In a later paper, in 2000, Sibert and Jacob [98] conclude that: *"Eye gaze-interaction is a useful source of additional input and should be considered when designing interfaces in the future."* Jacob's seminal work in eye tracking laid the foundation for this research on the use of gaze as a form of input.

In his paper on MAGIC pointing [118] Zhai states that "*to load the visual perception channel with a motor control task seems fundamentally at odds with users' natural mental model in which the eye searches for and takes in information and the hand produces output that manipulates external objects."* By his statement Zhai affirms that the eyes should be used for the purpose of looking and should not be overloaded with the unnatural task of doing actions, since that is counter to the evolutionary function of the eyes. By contrast, the hands are meant for performing actions. Zhai goes on to state that: *"Other than for disabled users, who have no alternative, using eye gaze for practical pointing does not appear to be very promising."* Zhai et al. address this challenge with their MAGIC techniques, which use

a conventional input device within the small area of the eye-gaze to accomplish pointing. We wanted to extend this approach of using gaze in conjunction with conventional input devices and determine if it is possible to devise other practical gaze-based interaction techniques by focusing on interaction design.

For our research we chose to investigate how gaze-based interaction can be made *simple*, *accurate,* and *fast enough* to not only allow disabled users to use it for standard computing applications, but also make the threshold of use low enough that able-bodied users will actually prefer to use gaze-based interaction to traditional input techniques.

2.3 History of Eye Tracking

The history of eye tracking can be traced as far back as the late 19th century and early 20th century [115]. Javal used direct visual observation to track eye movements in 1879. Ohm used mechanical techniques to track eye movements by attaching a pencil at the end of a long lever which was positioned on the cornea such that each time the eye moved the pencil would make a mark. The first recorded effort for eye tracking using a reflected beam of light was done by Dodge and Cline in 1901. Marx and Trendelenburg used a mirror attached to the eye to view the reflected beam of light. Judd, McAllister and Steel used motion picture photography for eye tracking as far back as 1905. They inserted a white speck into the eye which was then tracked in the motion picture recording of the eye.

Buswell [26] used eye tracking studies to examine how people look at pictures. Yarbus [115] in his pioneering work in the fifties used suction caps attached to the eye to measure eye movements. Yarbus shows several different designs of suction caps in his book and his work laid the foundation for the research in the field of eye movements.

Figure 3 shows a scleral coil contact lens which was inserted in the eye of the subject. The scleral contact lens contains an induction coil embedded in the periphery of the lens. The subject's head is kept stationary inside a magnetic cage. The changes in the magnetic field are then used to measure the subject's eye movements.

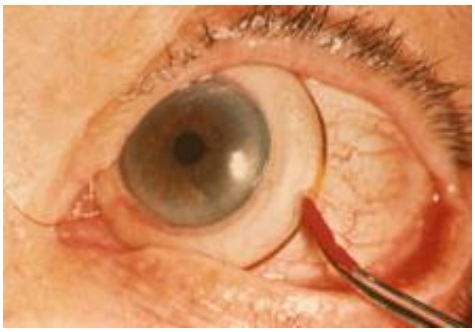


Figure 3. A scleral coil contact lens being inserted into a subject's eye.

Figure 4. Electro-oculography (EOG) approach for eye tracking measures the potential difference between eye muscles.

Figure 4 shows a picture of a subject whose eyes are being tracked using electro-oculography (EOG) which measures the potential difference between muscles of the eye.

The approaches to eye tracking have evolved significantly over the years. Fortunately, eye trackers today have become less invasive that their predecessors. Corneal reflection eye tracking was first introduced by the Dual Purkinje Eye Tracker developed at the Stanford Research Institute. This eye tracker used the reflection of light sources on the cornea as a frame of reference for the movement of the pupil. Figure 5 shows an image of a subject using the SRI eye tracker. It should be noted that this unit required the subject's head to be held stationary.

Head mounted eye trackers have been developed to fix the frame of reference for the eyes relative to the motion of the head (Figure 6). Some head mounted eye

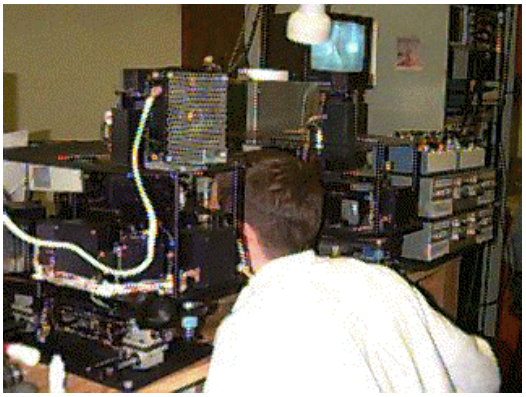


Figure 5. SRI Dual Purkinje Eye Tracker uses corneal reflections to track eye movements.

Figure 6. A head mounted eye tracker which fixes the position of the camera relative to the motion of the head.

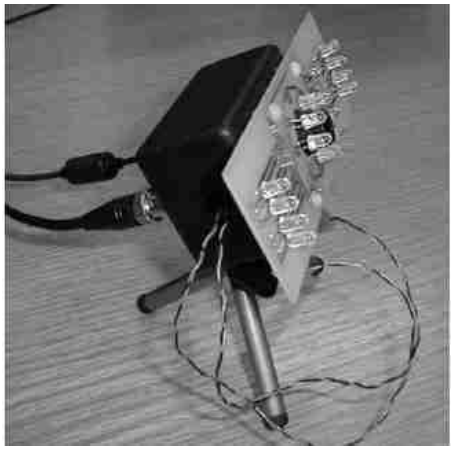


Figure 7. IBM BlueEyes Project prototype eye tracker which uses infra-red illumination.

Figure 8. The Tobii 1750 eye tracker.

trackers provide higher accuracy and frame rate than remote eye trackers since they are able to get a close up image of the eye by virtue of using the head mounted camera.

The BlueEyes project [4] at IBM Almaden developed remote video based eye trackers which used infra-red illumination as shown in Figure 7. Several commercial systems [67, 107, 108] have now been developed which use a similar approach for eye tracking and provide non-encumbering, remote, video-based eye tracking (Figure 8).

2.4 State of the Art in Eye Tracking

The state of the art systems for desktop eye tracking use remote video based eye tracking as described above. Unlike their historical counterparts, these eye trackers allow for some range of free head movement, do not require the user to use a chin-rest or bite bar or to be tethered to the eye tracker in any way. These systems work by measuring the motion of the center of the pupil relative to the position of one or more *glints* or reflection of infra-red light sources on the cornea. These systems provide an accuracy of about 0.5˚ - 1˚ of visual angle. While some systems

boast frame rates as high as 1000 Hz, most commercially available systems provide a frame rate of about 50 Hz.

For our research we use a Tobii 1750 eye tracker shown in Figure 8. This unit costs approximately $30,000, however, based on current technology and economic trends it is conceivable to have a similar unit incorporated into everyday computing devices.

2.5 Challenges for Gaze Input

The eyes are fast, require no training and eye gaze provides context for our actions [36, 52, 53, 116]. Therefore, using eye gaze as a form of input is a logical choice. However, using gaze input has proven to be challenging for three major reasons.

2.5.1 Eye Movements are Noisy

As noted by Yarbus [115], eye movements are inherently noisy. The two main forms of eye movements are *fixations* and *saccades*. Fixations occur when a subject is looking at a point. A saccade is a ballistic movement of the eye when the gaze moves from one point to another. Yarbus, in his pioneering work in the 1960's, discovered that eye movements are a combination of fixations and saccades even when the

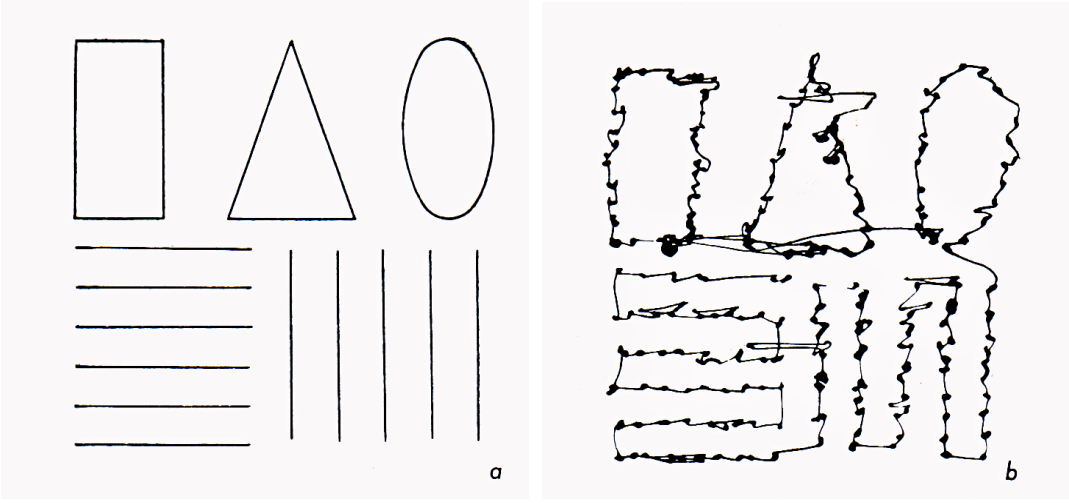


Figure 9. Trace of eye movements when subjects are asked to follow the lines of the figures as smoothly as possible. Source: Yarbus, 1967.

subjects are asked to follow the outlines of geometrical figures as smoothly as possible (Figure 9).

Yarbus, also points out that while fixations may appear to be dots in Figure 9, in reality, the eyes are not stable even during fixations due to drifts, tremors and involuntary micro-saccades (Figure 10).

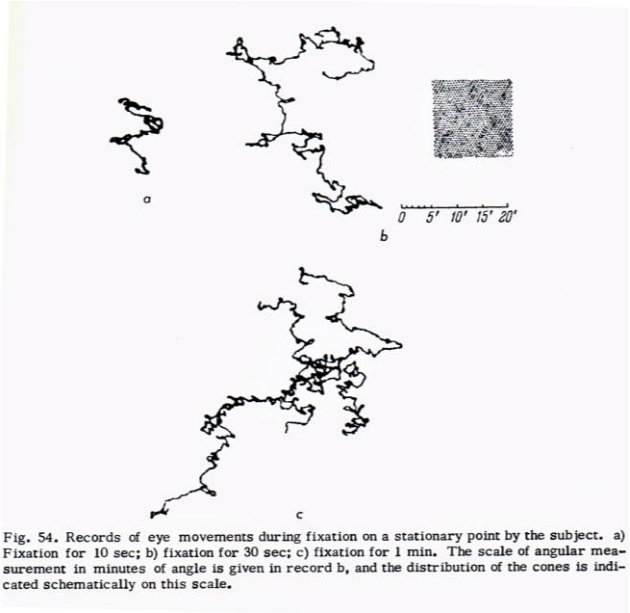


Fig. 54. Records of eye movements during fixation on a stationary point by the subject. a) Fixation for 10 sec; b) fixation for 30 sec; c) fixation for 1 min. The scale of angular measurement in minutes of angle is given in record b, and the distribution of the cones is indicated schematically on this scale.

Figure 10. Fixation jitter due to drifts, tremors and involuntary micro-saccades, Source: Yarbus, 1967.

2.5.2 Eye Tracker Accuracy

Modern day eye trackers, especially remote video based eye trackers, claim to be accurate to about 0.5˚ - 1˚ of visual angle. This corresponds to a spread of about 16-33 pixels on a 1280x1024, 96 dpi screen viewed at a normal viewing distance of about 50 cm [13, 107]. In practice this implies that the confidence interval for a point target can have a spread of a circle of up to 66 pixels in diameter (Figure 11), since if the user is looking at a point (1x1 pixel) target, the reading from the eye tracker can be off by up to 33 pixels in any direction. In addition, current eye trackers require calibration (though some require only a one-time calibration). The accuracy of the eye-tracking data usually deteriorates due to a drift effect caused by changes in eye characteristics over time [105]. Users' eyes may become drier after viewing information on a screen several minutes. This can change the shape and the reflective characteristics of the eyes. Users' posture also

Figure 11. Confidence interval of eye tracker accuracy. Inner circle is 0.5˚. Outer circle is 1.0˚.

changes over time as they begin to slouch or lean after some minutes of sitting. This results in the position/angle of their head changing. The accuracy of an eye tracker is higher in the center of the field of view of the camera. Consequently, the tracking is most accurate for targets at the center of the screen and decreases for targets that are located at the periphery of the screen [20]. While most eye trackers claim to work with eye glasses, we have observed a noticeable deterioration in tracking ability when the lenses are extra thick or reflective.

Current eye trackers are capable of generating data at 50Hz to 1000Hz depending upon the device and the application. However, eye trackers also introduce latency since they need computing cycles to processing data from the camera and compute the current position of the user's eye gaze. The Tobii eye tracker used in our research has a maximum latency of 35 ms.

2.5.3 The Midas Touch Problem

Mouse and keyboard actions are deliberate acts which do not require disambiguation. The eyes, however, are a perceptual organ meant for looking and are an always-on device [53]. It is therefore necessary to distinguish between visual search/scanning eye movements and eye movements for performing actions such as pointing or selection. This effect is commonly referred to as the "Midas Touch" problem [52].

Even if the noise from eye movements could be compensated for and if the eye trackers were perfectly accurate, the Midas Touch problem would still be a concern. This challenge for gaze as a form of input necessitates good interaction design to minimize false activations and to disambiguate the user's intention from his or her attention.

2.6 Summary

This chapter discussed the motivation for using gaze as a form of input, provided a historical background of eye tracking and introduced the current state of the art in eye tracking. While using gaze as a form of input is appealing, the challenges of interpreting noisy eye movements, eye tracker accuracy issues and the

Midas Touch problem must be addressed. In the following chapters of this dissertation we present several gaze-based interaction techniques for everyday computing tasks.

3 Pointing and Selection

We began our research by observing how able-bodied users use the mouse for pointing and selection in everyday computing tasks. While there are large individual differences in how people interact with the computer, nearly everyone used the mouse rather than the keyboard to select links while web browsing. Other tasks for which people used the mouse included launching applications either from the desktop or the start menu, navigating through folders, minimizing, maximizing and closing applications, moving windows, positioning the cursor when editing text, opening context-sensitive menus and hovering over buttons/regions to activate tooltips.

The basic mouse operations being performed to accomplish the above actions are the well-known single-click, double-click, right-click, mouse-over, and click-and-drag. Ideally a gaze-based pointing technique should support all of the above fundamental operations.

It is important to note that our aim is not to replace or beat the mouse. Our intent is to design an effective gaze-based pointing technique which can be a viable alternative for users who choose not to use a mouse depending on their abilities, tasks, or preferences. Such a technique need not necessarily outperform the mouse but must perform well enough to merit consideration (such as other alternatives like the trackball, touchpad, or trackpoint).

Portions of this chapter were originally published by the author, Andreas Paepcke and Terry Winograd in [61] and by the author and Terry Winograd in [63].

3.1 Related Work

Considerable prior research [13, 23, 38, 40, 52, 66, 74, 96, 114, 118] has been done to implement gaze-based pointing techniques. However, a practical technique for pointing and selection is still an open problem. The commonly accepted approach to using gaze-based pointing and selection relies on the use of large targets in custom applications [52, 67, 106]. Other approaches have used speech [74], keyboard [23] and mouse [118] for doing target refinement; used zoomed views [13, 66] or leveraged semantic information [96] about the location of potential targets to improve gaze-based pointing. We discuss each of these in more detail below. It should be noted that we chose not to leverage semantic information in our work since we wanted to have a general purpose pointing technique that does not rely on any additional information from the application or the operating system.

Jacob [52] introduces gaze-based interaction techniques for *object selection*, *continuous attribute display*, *moving an object*, *eye-controlled scrolling text*, *menu commands* and *listener window*. This work laid the foundation for eye-based interaction techniques. It introduced key-based and dwell-based activation, gaze-based hot-spots, and gaze-based context-awareness for the first time. Issues of eye tracker accuracy were overcome by having sufficiently large targets in custom applications.

Zhai et al. [118] presented the first gaze-enhanced pointing technique that used gaze as an augmented input. In MAGIC pointing, the cursor is automatically warped to the vicinity of the region in which the user is looking. The

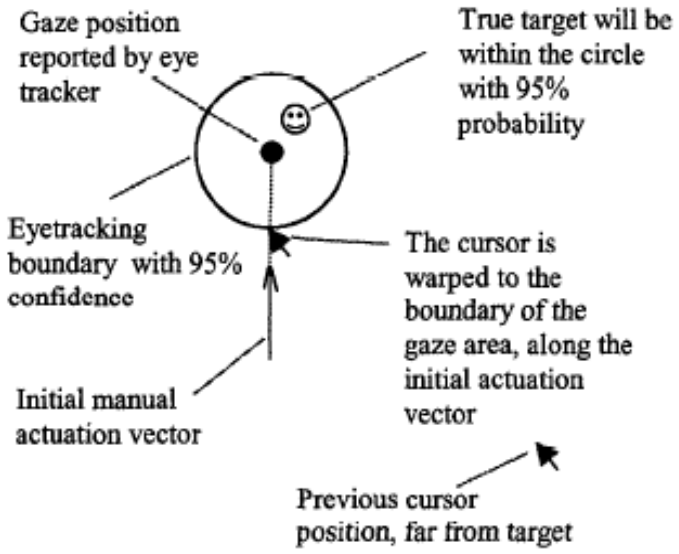


Figure 12. Zhai et al.'s illustration of the MAGIC pointing technique.

MAGIC approach leverages Fitts' Law [39] by reducing the distance that the cursor needs to travel. Though MAGIC uses gaze as an augmented input, pointing is still accomplished using the mouse.

Salvucci and Anderson [96] also use gaze as an augmented input in their work and emphasize that all normal input device functionality is maintained. Their system incorporates a probabilistic model of user behavior to overcome the issues of eye tracker accuracy and to assist in determining user intent. Furthermore, Salvucci and Anderson prefer the use of key based activation as opposed to dwell-based activation. The probabilistic model relies on the use of semantic information provided by the underlying operating system or application about click target locations and hence is not conducive to general use on commercially available operating systems and applications.

Yamato et al. [114] also propose an augmented approach, in which gaze is used to position the cursor, but selection is still performed using the mouse button. Their approach used automatic and manual adjustment modes for target refinement. However, the paper claims that manual adjustment with the mouse was the only viable approach, rendering their technique similar to MAGIC, with no additional advantages.

Lankford [66] introduced a dwell-based technique for pointing and selection. The target provides visual feedback when the user's gaze is directed at it. The user has the ability to abort activation by looking away before the dwell period expires. Lankford also uses zooming to overcome eye tracker accuracy measures. The approach requires one dwell to activate the zoom (which always appears in the center of the screen) and an additional dwell to select the target region and bring up a palette with different mouse action options. A third dwell on the desired action is required to perform the action. This approach does implement all the standard mouse actions and while it is closest to our technique (described below), the number of discrete steps required to achieve a single selection and the delays due to dwell-based activation make it unappealing to users for whom a traditional pointing device is a viable alternative. By contrast, our approach innovates on the interaction techniques to make the interaction fluid and simple for all users.

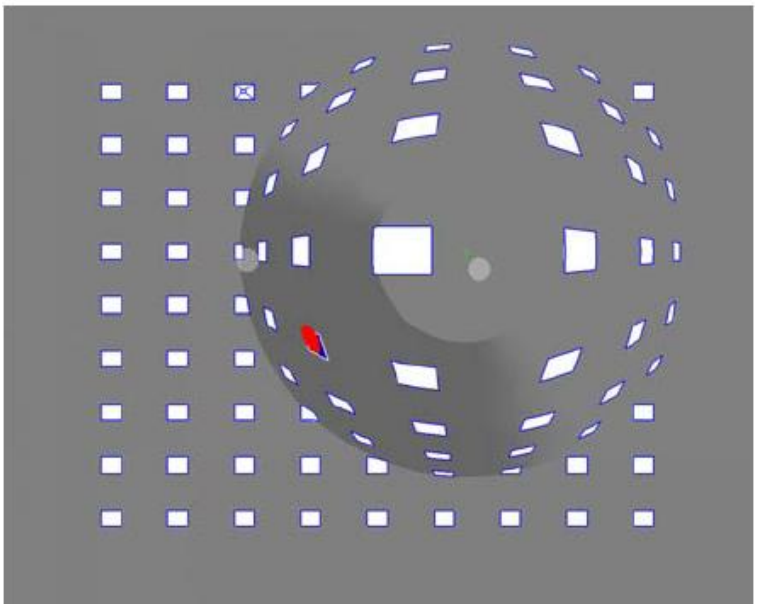


Figure 13. Ashmore et al.'s implementation of a fish eye lens for gaze-based pointing.

Follow-on work to MAGIC at IBM by Beymer, Farrell and Zhai [38] proposes a technique that addresses the other dimension of Fitts' Law, namely target size. In this approach the region surrounding the target is expanded based on the user's gaze point to make it easier to acquire with the mouse. In another system by Farrell and Zhai [23], semantic information is used to predictively select the most likely target with error-correction and refinement done using cursor keys.

Ashmore and Duchowski et al. [13] present an approach using a fish-eye lens to magnify the region the user is looking at to facilitate gaze-based target selection by making the target bigger. They compare approaches in which the fish-eye lens is either non-existent, slaved to the eye movements, or dynamically appearing after a fixation. However, as stated in their paper, the visual distortion introduced by a fish-eye view is not only confusing to users but also creates an apparent motion of objects within the lens' field of view in a direction opposite to that of the lens' motion.

Fono and Vertegaal [40] also use eye input with key activation. They show that key activation was preferred by users over automatic activation. Finally, Miniotas et al. [74] present a speech-augmented eye-gaze interaction technique in

Figure 14. Using EyePoint for a progressive refinement of target using look-press-look-release action. The user first looks at the desired target. Pressing and holding down a hotkey brings up a magnified view of the region the user was looking in. The user then looks again at the target in the magnified view and releases the hotkey to perform the mouse action.

which target refinement after dwell based activation is performed by the user verbally announcing the color of the correct target. This again requires semantic information and creates an unnatural interaction by requiring the user to correct selection errors using an additional modality.

3.2 EyePoint

Our system, EyePoint, uses a two-step progressive refinement process that is fluidly stitched together in a look-press-look-release action (Figure 14). This two-step approach compensates for the accuracy limitations of current state-of-the-art eye trackers, enabling users to achieve accurate pointing and selection without having to rely on a mouse.

EyePoint requires a one-time calibration. In our case, the calibration is performed using the APIs provided in the Software Development Kit for the Tobii 1750 Eye Tracker [107]. The calibration is saved for each user and re-calibration is only required in case there are extreme variations in lighting conditions or the user's position in front of the eye tracker.

To use EyePoint, the user looks at the desired target on the screen and presses a hotkey for the desired action — single-click, double-click, right-click, mouse-over, or start click-and-drag. EyePoint displays a magnified view of the region the user was looking at. The user looks at the target again in the magnified view and releases the hotkey. This results in the appropriate action being performed on the target (Figure 14).

Figure 15. Focus points - a grid of orange dots overlaid on the magnified view helps users focus their gaze.

To abort an action, the user can look anywhere outside of the zoomed region and release the hotkey, or press the *Esc* key on the keyboard.

The region around the user's initial gaze point is presented in the magnified view with a grid of orange dots overlaid (Figure 15). These orange dots are called *focus points* and aid in focusing the user's gaze at a point within the target. This mechanism helps with more fine-grained selections. Further detail on focus points is provided in the following section.

Single-click, double-click and right-click actions are performed when the user releases the key. Click and drag, however, is a two-step interaction. The user first selects the starting point for the click and drag with one hotkey and then the destination with another hotkey. While this does not provide the same interactive feedback as click-and-drag with a mouse, we preferred this approach over slaving movement to the user's eye-gaze, based on the design principles discussed below.

3.2.1 Design Principles

We agreed with Zhai [25] that overloading the visual channel for a motor control task is undesirable. We therefore resolved to determine if there was an interaction technique for using eye gaze in practical pointing *without* overloading the visual channel for motor control.

Another basic realization was that larger targets are easier to acquire using eye gaze. Therefore, to use eye gaze for pointing it would be ideal if all the targets were large enough to not be affected by the accuracy limitations of eye trackers and the jitter inherent in eye gaze tracking. A similar rationale was adopted in [38].

As recognized in prior work [13, 40, 66, 73, 117] for gaze-based pointing zooming and magnification help to increase accuracy in pointing and selection. We sought ways in which zooming and magnification could be used in a unobtrusive way and unlike [13], would not cause any visual distortion of their context.

As previously stated, our goal was to devise an interaction technique that would be universally applicable – for both users for whom the mouse is a viable alternative and for those for whom it is not.

We concluded that it is important to a) avoid slaving any of the interaction directly to eye movements (i.e. not overload the visual channel for pointing), b) use zooming/ magnification in order to overcome eye tracker accuracy issues c) use a fixation detection and smoothing algorithm in order to reduce tracking jitter and d) provide a fluid activation mechanism that is fast enough to make it appealing for able-bodied users and simple enough for disabled users.

3.2.2 EyePoint Implementation

With EyePoint, the eye tracker constantly tracks the user's eye- movements[1]. A modified version of Salvucci's Dispersion Threshold Identification fixation detection algorithm [97] is used to determine the location of the current fixation.

[1] If the eye tracker were fast enough, it would be possible to begin tracking only when the hotkey is pressed. This would also alleviate any concerns about long-term exposure to infra-red from the eye tracker.

When the user presses and holds one of four action-specific hotkeys on the keyboard, the system uses the key press as a trigger to perform a screen capture in a *confidence interval* around the user's current eye-gaze. The default settings use a confidence interval of 120 pixels square (60 pixels in all four directions from the estimated gaze point). The system then applies a *magnification factor* (default 4x) to the captured region of the screen. The resulting image is shown to the user at a location centered at the previously estimated gaze point, but offset when close to screen boundaries to keep the magnified view fully visible on the screen.

EyePoint uses a secondary gaze point in the magnified view to refine the location of the target. When the user looks at the desired target in the magnified view and releases the hotkey, the user's gaze position is recorded. Since the view has been magnified, the resulting gaze position is more accurate by a factor equal to the magnification. A transform is applied to determine the location of the desired target in screen coordinates. The cursor is then moved to this location and the action corresponding to the hotkey (single-click, double-click, right-click etc.) is executed.

EyePoint therefore overcomes the accuracy problem of eye trackers by using magnification and a secondary gaze fixation. The secondary gaze-fixation is achieved by using a fluid look-press-look-release action. As explained by Buxton [27], the two steps refinement in EyePoint would be considered a compound task. The "glue," in Buxton's words, that ties the steps together is the tension of holding the hotkey down, which gives constant feedback to the user that we are in a temporary state, or mode. Explicit activation by the hotkey means that it does not suffer from the Midas Touch problem. Additionally, EyePoint does not overload the visual channel as the eyes are only used for looking at the target.

The user must perform a secondary visual search to refocus on the target in the magnified view necessitating one or more saccades in order to locate the target in the magnified view. To facilitate the secondary visual search we added animation to the magnified view such that it appears to emerge from the initially estimated gaze point.

We tested our initial design by conducting pilot studies, which showed that the gaze data from the fixation on the target in the magnified view was noisy for

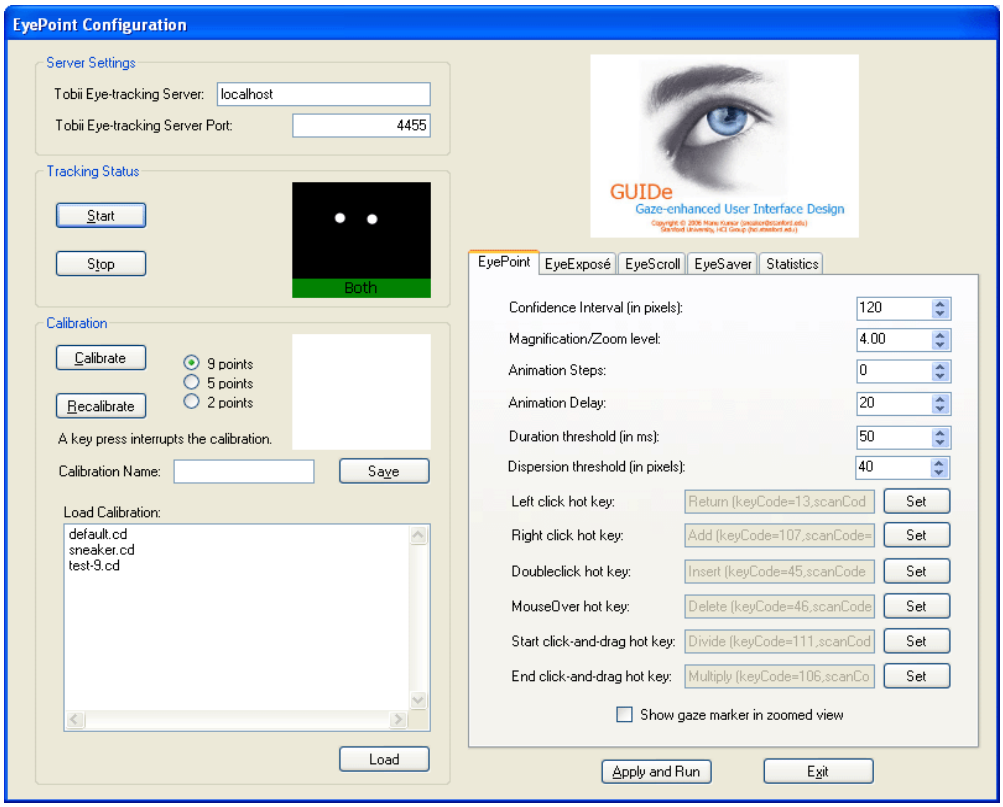


Figure 16. EyePoint configuration screen.

some users. We found that this occurred when the user was looking at the target as a whole (a gestalt view) rather than focusing at a point within the target. Focusing at a point reduced the jitter and improved the accuracy of the system. This led to the introduction of *focus points* in the design – a grid pattern of dots overlaid on the magnified view. Focus points assist the user in making more fine grained selections. Focus points are most helpful in selecting small targets; users are of course free to ignore them (Figure 15). We discuss a study of focus points in Section 9.3.

Some users in the pilot study wondered whether it would be useful to give feedback on what the system thought they were looking at. While this went strongly against our primary design principle of not slaving any visual feedback to eye movements, we implemented a *Gaze Marker* to show the current gaze point as a blue dot in the magnified view. When the same users tried the system with the gaze marker turned on, they quickly concluded that it was distracting. The time to acquire targets increased, since they were now trying to get the gaze marker in precisely the

right position before releasing the hotkey (which is unnecessary since the magnification allows some room for error). As a result, we turned off the gaze marker by default, but decided to test it further in our evaluation.

The default keys for EyePoint are those on the numeric keypad of an extended keyboard (Figure 14 Press) since they are not frequently used, are on the right hand side of the keyboard (close to the typical location for a mouse), and provide larger keys. The ideal placement for EyePoint hotkeys would allow the user's hands to always remain in the home position on the keyboard, perhaps by having dedicated buttons directly below the spacebar. Eye Point allows users to customize several options such as the selection of hotkeys, settings for the confidence interval, the magnification factor, the number of animation steps and the animation delay. The EyePoint configuration screen is shown in Figure 16.

3.2.3 Disabled & Able-bodied Users

EyePoint is designed to address the needs of disabled users and able-bodied users like. In the context of this dissertation, we use the term disabled used to refer to users who are unable to make use of a traditional pointing device, such as a mouse. The hotkey-based triggering mechanism in EyePoint makes it possible for able-bodied users to keep their hands on the keyboard to perform most pointing and selection operations. For laptop users we have considered using gestures on a touchpad where touching different parts of the touchpad would activate different mouse actions.

For disabled users, the EyePoint hotkeys could be mapped to alternative triggering devices such as foot-pedals, speech, gestures or even mouth-tube triggers (breathe in to activate, breathe out to release). We hypothesize that these will be more effective than dwell-based activation, but have not studied these alternatives. Dwell-based activation is also possible in cases where the user does not have the ability to use any alternative devices. In this case we would propose an approach similar to [66], but with off-screen targets to first select the action/mode, followed by dwell based activation (with audio feedback [71]) of the magnified view.

3.3 Evaluation

We conducted three user studies with 20 able-bodied subjects. Subjects were graduate students and professionals and were therefore experienced computer users with an average of 15 years of experience with the mouse. Our subject pool had 13 males and 7 females with an average age of 28 years. Fourteen subjects did not require any vision correction, 4 subjects used contact lenses and 2 wore eyeglasses. None of the subjects were colorblind. Sixteen subjects reported that they were touch-typists. None of the subjects had prior experience using an eye tracker.

We conducted a quantitative evaluation to measure performance and a qualitative evaluation to measure users' subjective opinion. The quantitative task compared the speed and accuracy of three variations of EyePoint with that of a standard mouse. The three variations of EyePoint were: a) EyePoint with Focus Points b) EyePoint with Gaze Marker and c) EyePoint without Focus Points or Gaze Marker. Our qualitative evaluation included the user's subjective feedback on using gaze-based pointing. Consistent with Norman's views in Emotional Design [81], we believe that speed and accuracy must meet certain thresholds. Once that threshold is met, user preference may be dictated by other factors such as the subjective experience or alternative utility of the technique.

It should be noted that while Ware et al. [110] show that gaze-based pointing conforms to Fitts' Law, the opinions of researchers in the eye tracking community are mixed. Additionally, since EyePoint requires a secondary gaze fixation on the target in the magnified view we chose to create tasks designed specifically for measuring the speed and accuracy of EyePoint as opposed to a traditional Fitts' Law task.

3.3.1 Quantitative Evaluation

We tested speed and accuracy using three independent experiments: a) a web browsing task b) a pointing task and c) a mixed typing and pointing task. The orders of both the tasks and the techniques were varied to counterbalance and minimize any learning effects. Subjects were first calibrated on the eye tracker and then underwent a 5-10 minute training phase in which they were taught how to use

Figure 17. EyePoint real-world web-surfing task. The music link in the navigation column on the left has been highlighted in orange.

EyePoint. Subjects practiced by clicking on links in a web browser and also performed 60 clicks in the EyePoint training application (Figure 18). Studies lasted a total of 1 hour and included one additional task reported in a separate paper [60]. The *spacebar* key was used as the trigger key for all three EyePoint variations. Animation of the magnified view was disabled as it introduces an additional delay (user configurable, but generally about 60-100ms).

3.3.1.1 Web Study

For the first pointing and selection task we asked users to navigate through a series of web pages. The pages were taken from popular websites such as Yahoo, Google, MSN, Amazon, etc. To normalize effects of time for visual search and distance from the target, we disabled all links on the page and highlighted exactly one link on each page with a conspicuous orange highlight (Figure 17).

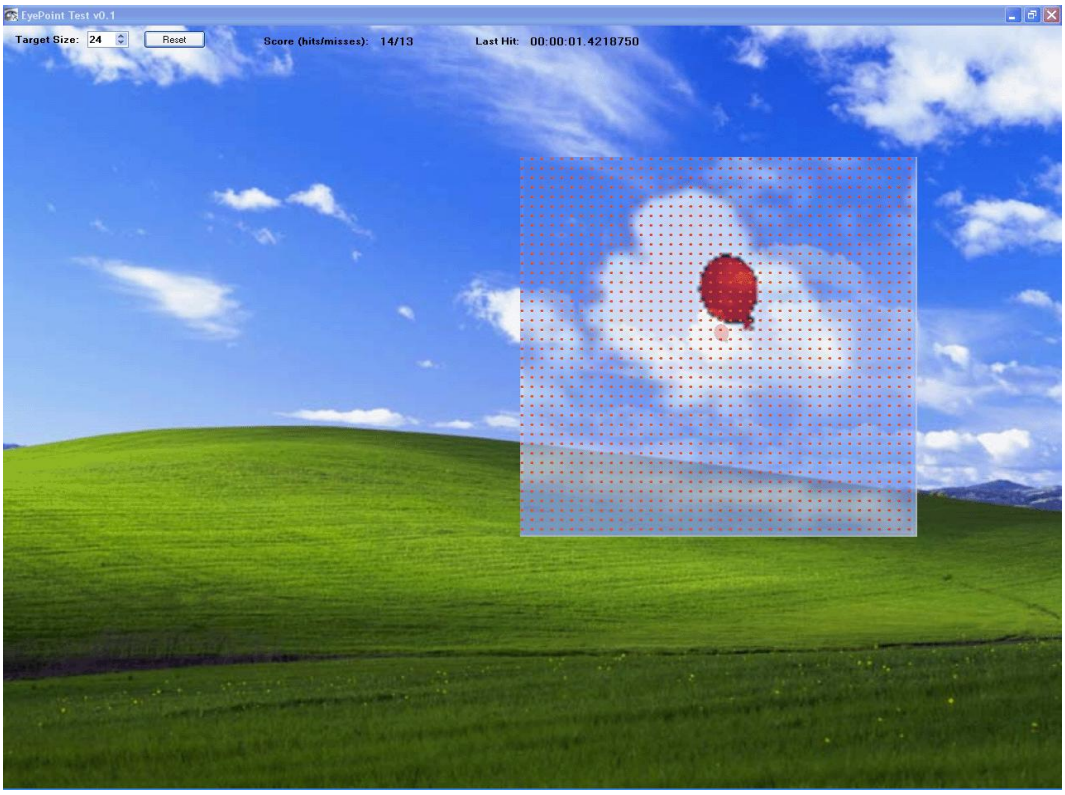


Figure 18. EyePoint training/test application (used for Balloon Study). This screenshot shows the magnified view with focus points.

Users were instructed to ignore the content of the page and simply select the highlighted link. Each time they selected the link, a new web page appeared with another highlighted link. The amount of time between presentation of a page and the click was measured. A misplaced click was recorded as an error. Trials were repeated in case of an error. Each subject was shown 30 web pages. The task was repeated with the same set of pages for all four pointing techniques, with ordering counterbalanced.

3.3.1.2 Balloon Study

To test raw pointing speed, we built a custom application that displayed a red balloon on the screen. The user's task was to select the balloon. Each time the balloon was selected, it moved to a new location (Figure 18). If the user clicked, but missed the target, this was recorded as an error and the trial was repeated. Users were instructed to click on the balloon as quickly as they could. The application gathered

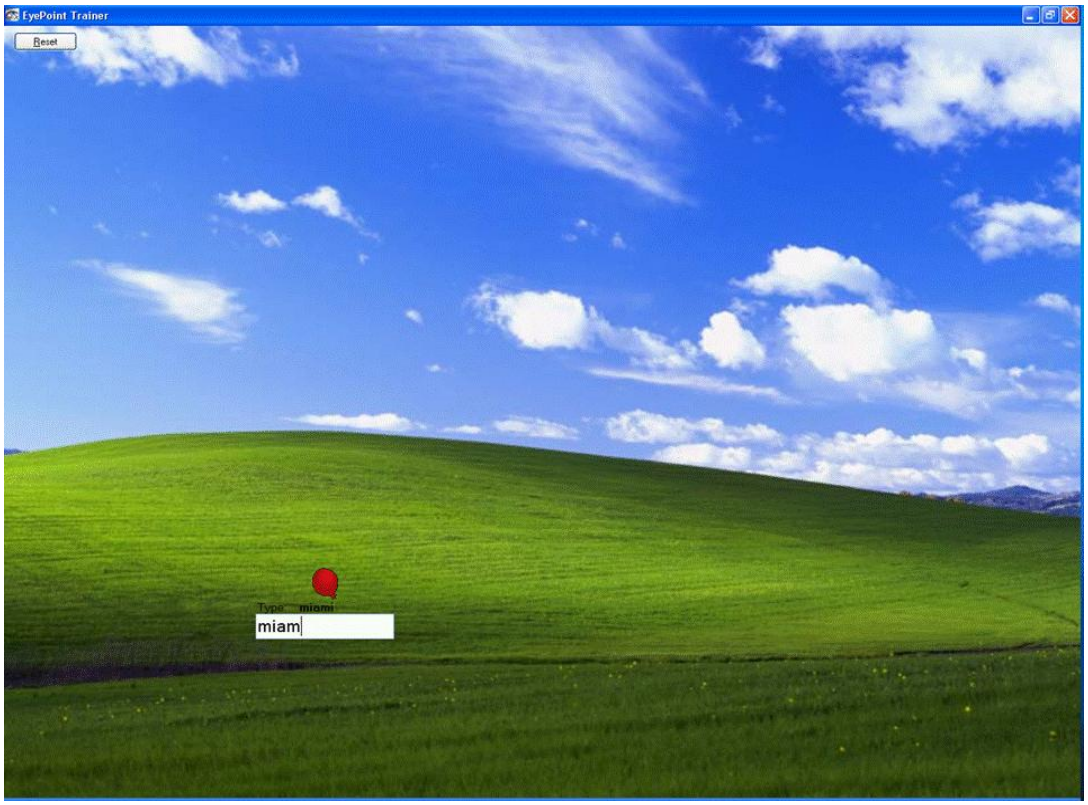


Figure 19. Mixed task study for pointing and typing. When the user clicks on the red balloon, a textbox appears below it. The user must type in the word shown above the textbox.

timing data on how long users took to perform the click. The size of the balloon was varied among 22 pixels (the default size of a toolbar button), 30 pixels and 40 pixels. The resulting study is a 4 by 3 within-subjects study (4 techniques, 3 sizes).

3.3.1.3 Mixed Study

The third task was a mixed typing and pointing task. The goal of this task was to force subjects to move their hands between the keyboard and the mouse. In this study, subjects first clicked on the target (a red balloon of constant size) and then typed a word in the text box which appeared after they clicked (Figure 19). We measured the amount of time from the click to the first key pressed on the keyboard and the time from the last character typed to clicking on the next balloon. Subjects did not have to press *Enter* (unlike [34]). As soon as they had typed the correct word, the system would show the next balloon. The amount of time to correctly type the

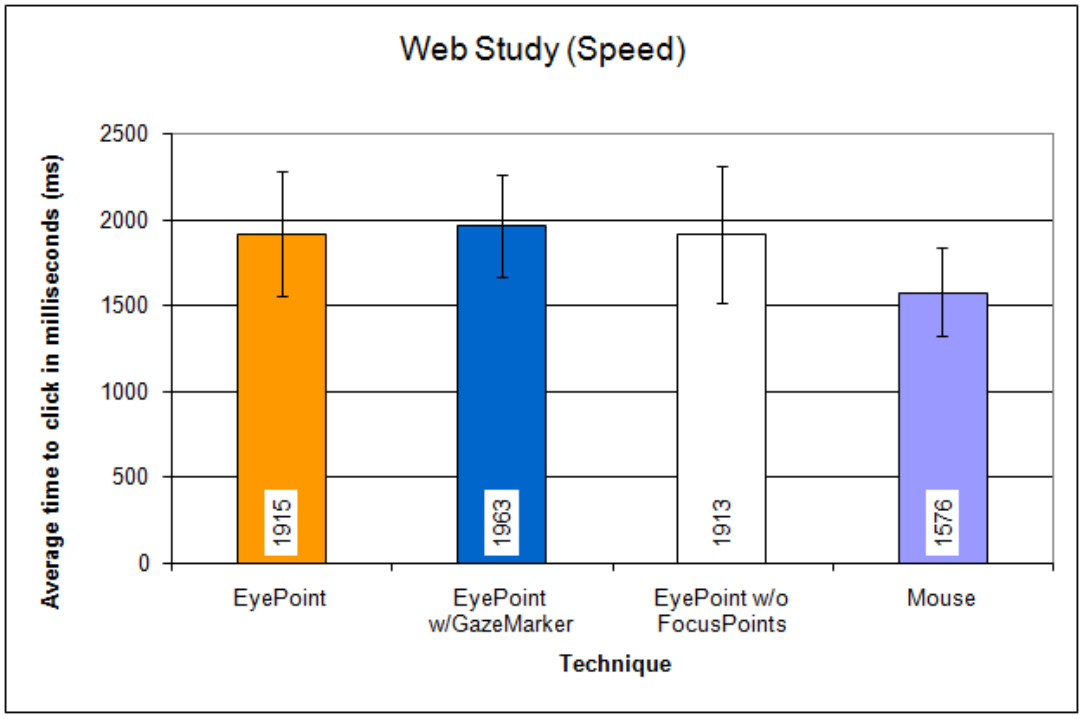


Figure 20. EyePoint Web Study speed results.

word shown was not considered, because we were only interested in the subject's ability to point and not how well they could type. If the subject clicked but did not hit the balloon, this was recorded as an error and the trial was repeated.

The sum of the two measured times is the round-trip time to move the hands from the keyboard to the mouse, click on a target and then return back to the keyboard. The Mixed Study compared the mouse with basic EyePoint, i.e. without a gaze marker but with focus points.

3.3.2 Qualitative Evaluation

For the qualitative evaluation to measure users' subjective opinion, users were asked to fill out a questionnaire at the end of each study and to provide their comments and opinions on the interaction techniques. They were asked to rank gaze-based pointing and the mouse for speed, accuracy, ease of use, and preference. In addition, subjects were also asked about which of the EyePoint variations (with focus points, with gaze marker, or without focus points) they preferred.

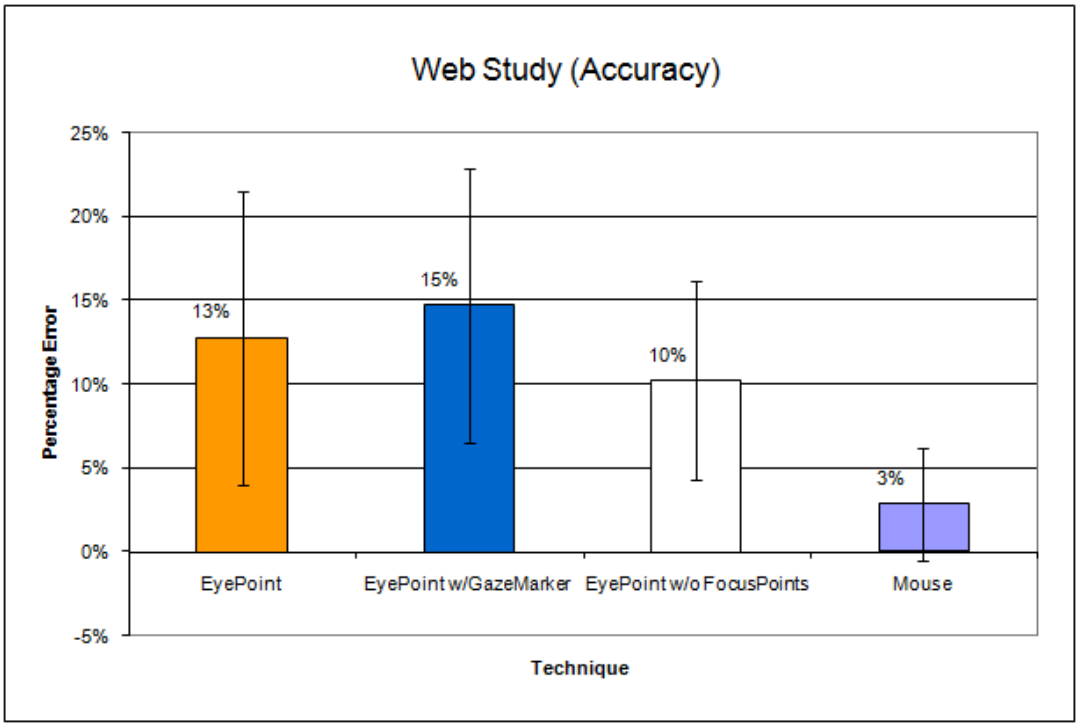


Figure 21. EyePoint Web Study accuracy results.

3.3.3 Web Study Results

Figure 20 shows the average time to click in the Web Study. A repeated measures ANOVA for technique showed that the differences are significant ($F(3,57)=11.9$, $p<.01$). Contrast analyses showed a significant difference for each eye-based technique when compared to the mouse. Differences for the gaze marker condition were also significant. However, there was no significant difference between the focus points and no focus point conditions. The average time to click with the mouse was 1576 milliseconds and 1915 milliseconds with EyePoint.

Figure 21 shows the accuracy results for the Web Study. A repeated measures ANOVA analysis showed that the differences in error rate are significant ($F(3,57)=14.9$, $p<.01$). Contrast analyses showed a significant difference in error rate between each eye-based technique and the mouse. Differences among the three eye-based variations were not significant. The mouse had an average error rate of 3%, while the EyePoint error rate was 13%. The no focus points condition had an average error rate of 10%.

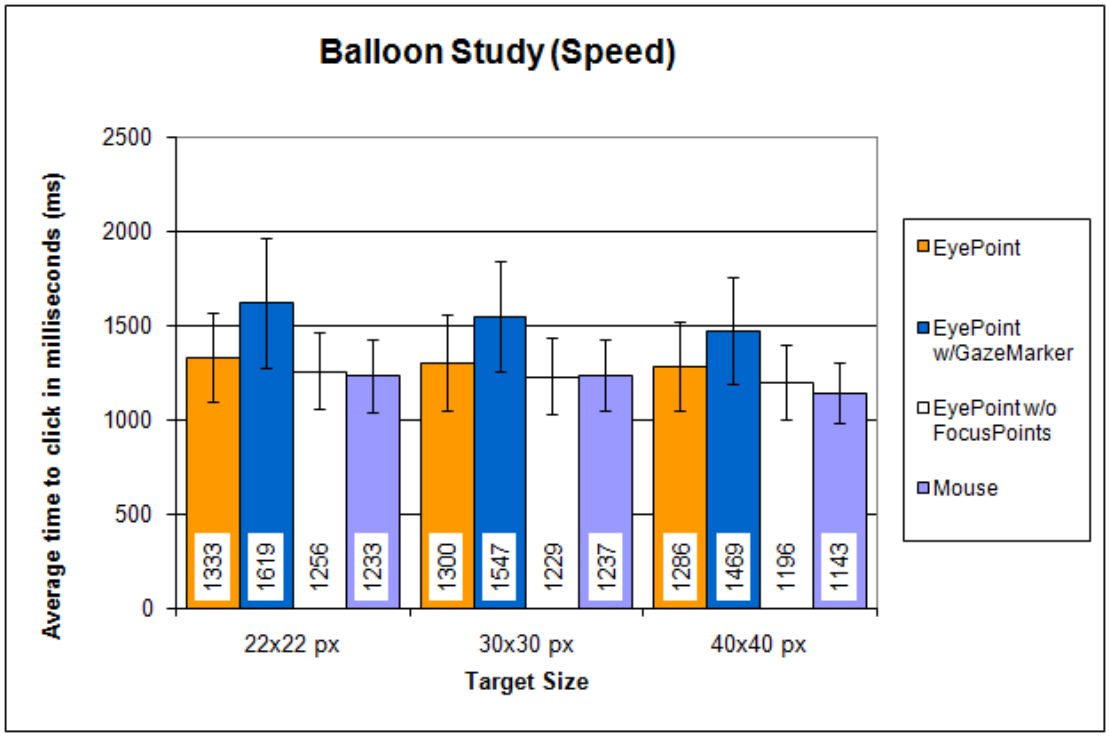


Figure 22. Balloon Study speed results.

Survey results for the Web Study showed that subjects' opinions were evenly split on which technique was faster (EyePoint or mouse) and which was easier to use. Although all subjects felt that the mouse was more accurate, three quarters of the subjects said they would choose to use EyePoint for this task over the mouse since they felt it was faster, easier or just cooler. A majority of the subjects preferred having focus points and felt that the focus points gave them something to "hold" on to. We discuss the reasons users gave for the subjective opinions in Section 3.4.

3.3.4 Balloon Study Results

Figure 22 shows the average time to click in the Balloon Study. EyePoint performs on average about 8.3% (100ms) slower than the mouse. A repeated measures ANOVA for size and technique showed a significant effect for size ($F(2,38) = 26.8$; $p < .01$), and for technique ($F(3, 57) = 14.8$; $p < .01$). We found no interaction effect between size and technique. Contrast analyses showed that significant differences existed for all pairs of sizes. For technique, contrasts showed a significant

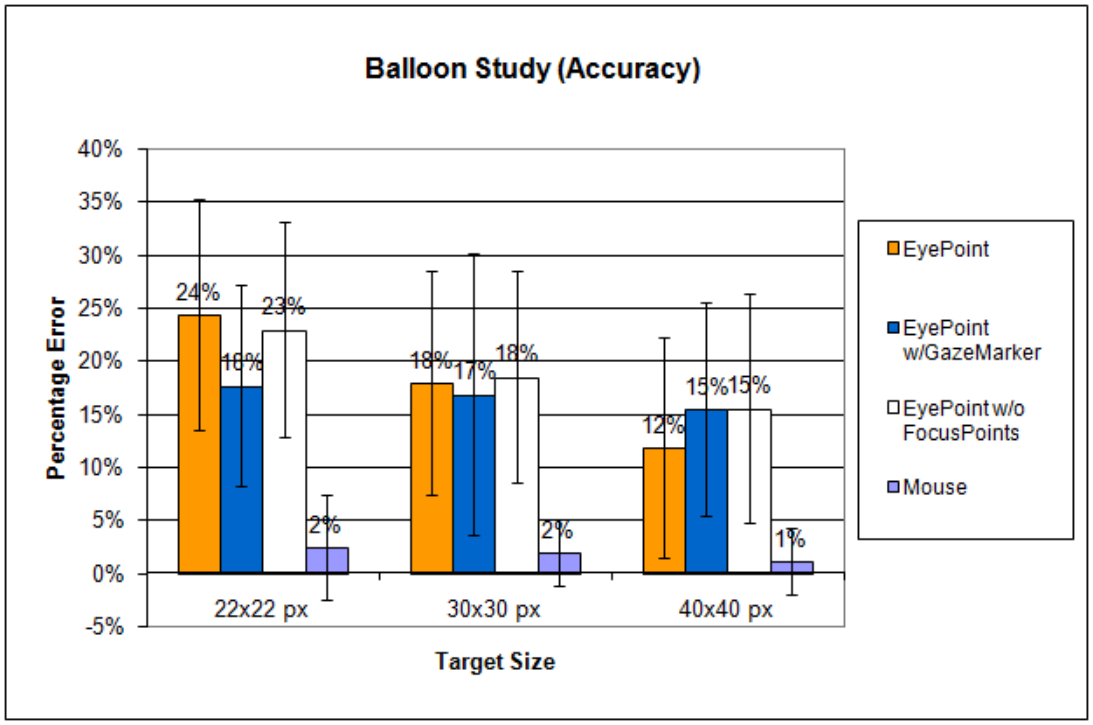


Figure 23. Balloon Study accuracy results.

difference between all pairs of techniques except EyePoint with no focus points vs. mouse.

Figure 23 shows the error rates for the Balloon Study. In accordance with Fitts' Law, the size of the target did have an appreciable impact on the error rates. Contrast analyses showed that the differences in error rates among the gaze-based techniques were not significant. The differences between each of the gaze-based techniques and the mouse were significant. It should be noted that the error rates for gaze-based pointing techniques were considerably higher than in the web study. We will discuss these results in the next section.

Survey results for the Balloon Study showed that subjects found the mouse to be faster and more accurate. However, the gaze-based techniques were still perceived to be easier to use, and three quarters of the subjects again said they would prefer to use the gaze-based technique for this task. Subjects felt that moving the mouse was fatiguing over time and that it was easier to click using the gaze-based methods despite the speed disadvantage. We discuss the reasons users gave for the subjective opinions in Section 3.4.

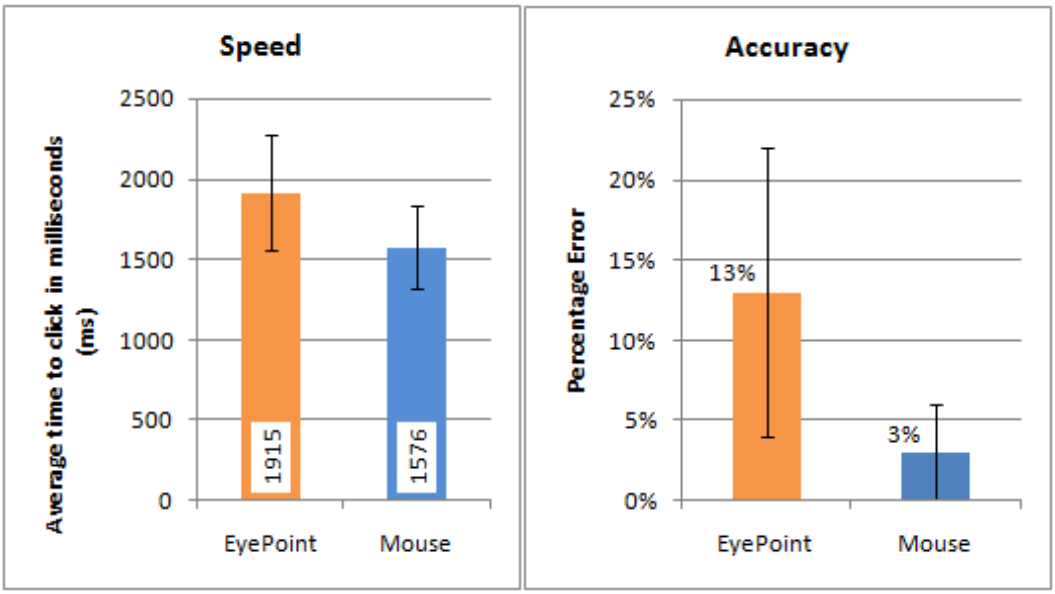


Figure 24. Mixed Study performance/error results.

3.3.5 Mixed Study Results

Figure 24 shows the total round trip time to point to a target and return to the keyboard for the Mixed Study. EyePoint is faster than the mouse in this task. A paired sample two-tailed *t*-Test showed that the results are statistically significant with $p < .05$.

Figure 24 also shows the accuracy results from the Mixed Study. It should be noted that while the gaze-based technique had a lower time to point and return to the keyboard, it had much lower accuracy than the mouse. A paired sample two-tailed *t*-Test showed that the error results are statistically significant with $p < .01$.

Survey results for the Mixed Study showed a strong preference (>90%) for EyePoint on the speed, ease of use and user preference dimensions — primarily because users didn't have to move their hands off the keyboard. The mouse was preferred only on the accuracy dimension.

3.4 Discussion

The above results present an incomplete picture without a deeper analysis. If we isolate the actions the user must perform to point and click on a target with the mouse, the total time would be:

$$T_{mouse} = t_{acquire\ target} + t_{acquire\ mouse} + t_{acquire\ cursor} + t_{move\ mouse} + t_{click\ mouse}$$

where $t_{acquire\ target}$ is the amount of time it takes the user to conduct a visual search for the target, $t_{acquire\ mouse}$ is the amount of time to move the hand from the keyboard to the mouse (if the hands are not already on the mouse), $t_{acquire\ cursor}$ is the amount of time to locate the cursor on the screen, $t_{move\ mouse}$ is the amount of time to move the mouse and $t_{click\ mouse}$ is the amount of time to click the mouse button.

By contrast, the total time for selection using EyePoint would be:

$$T_{eyepoint} = t_{acquire\ target} + t_{acquire\ hotkey} + t_{press\ hotkey} + t_{reacquire\ target} + t_{release\ hotkey}$$

where $t_{acquire\ target}$ is the again the amount of time it takes the user to conduct a visual search for the target, $t_{acquire\ hotkey}$ is the amount of time to position the hand on the hotkey, $t_{press\ hotkey}$ is the amount of time to press the hotkey, $t_{reacquire\ target}$ is the amount of time it takes the user to conduct the secondary visual search and find the target in the magnified view of EyePoint, and $t_{release\ key}$ is the amount of time to release the hotkey.

It can be reasonably expected that the time to acquire the target, i.e. perform a visual search for the target is the same in both cases. The time to acquire the mouse vs. the hotkey would depend on Fitts' Law [28, 34]. In our studies, we found that having a large hotkey such as the space bar, reduced the acquisition time for the hotkey. The key performance difference between using the mouse and using the eye arises from the second visual search to re-acquire the target in the magnified view. We observed that subjects were able to parallelize tasks when using the mouse. For instance, they would already have their hand on the mouse and begin moving it even before they had performed the visual search. This may be the result of years of practice with using the mouse. Due to the concurrent nature of the sub-tasks for pointing with the mouse, the amount of time it takes the user to move the mouse and the amount of time it takes the user to perform a secondary visual search when using

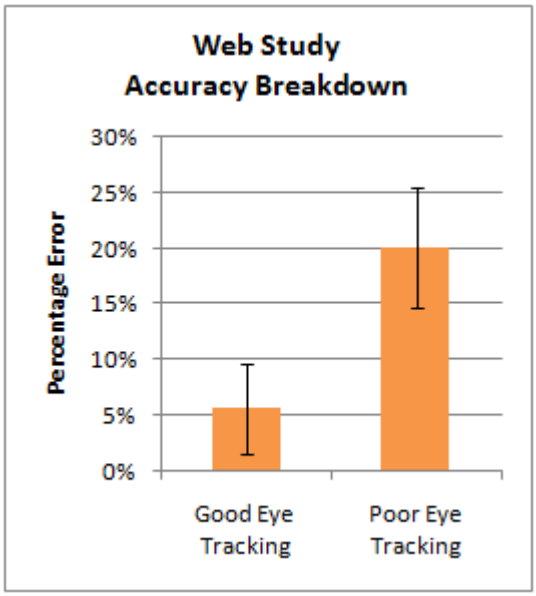


Figure 25. Breakdown of error rates from the web study into two groups: users for whom the eye tracker worked well and users for whom the eye tracker didn't work as well.

gaze are similar (assuming the time to click the mouse and release the key are similar).

Based on the empirical results and the model proposed above, we find that the pointing speed of EyePoint is similar to the performance of the mouse and can actually be faster than the mouse for mixed pointing and typing tasks.

The analysis of error rates is a little more complex. While the results shown in the previous section suggest that the error rates when using gaze-based pointing are considerably higher than those when using the mouse, the graphs do not tell the complete story. A deeper analysis of the error data showed that the error rates varied significantly across subjects. The eye-gaze tracker works better for some subjects than others. The accuracy of the eye-tracking depends not only on the individual, but on the quality of the calibration and the posture of the subject over the course of the experiments.

If we partition the error data from the Web Study into subjects for whom the eye tracker worked well and subjects for whom the eye tracker didn't work as well, the average error rate for the first group is 6% while that for the second group is 20% (Figure 25).

In the case of the balloon studies, we observed that since the task required subjects to click on the balloons in rapid succession, some subjects would press the EyePoint hotkey prematurely, in anticipation of the next balloon, before they even actually looked at it. This resulted in a significantly higher error rate. In practice, we can reasonably expect that subjects will look at the target before activating the hotkey.

The implementation of EyePoint uses a fixation detection algorithm that expects the subject's gaze to be within a certain region for at least 25-50 ms before it updates the current gaze coordinate. This resulted in timing issues in the balloon studies. Subjects would see the balloon in their peripheral vision and press the hotkey before their foveal vision fixated on the target. To reduce such errors, we propose measuring the initial fixation during a window of time that extends slightly beyond the hotkey activation time, thereby giving the subject the ability to focus on the target before the gaze-point is determined. We present details of this technique in Chapter 9.

Our observation of the subjects while they performed the study also revealed other interesting details. One subject, for instance, laughed and smiled a lot, which caused the subject's eyes to squint and resulted in a loss of eye-tracking accuracy (sometimes no data at all). Our pilot studies included a subject with astigmatism and weighted contact lenses which reduced the accuracy of the eye tracker, possibly due to the differential movement of the weighted contact lenses. For subjects with glasses we found that large frames with work better than narrow frames because the rim of the frame doesn't occlude the view of the camera. Similarly, this lenses work better than thick lenses since thick lenses introduce a distortion of the eyes as seen by the camera in the eye tracker.

Experimental effects also contributed to an increased error rate. In particular, since we instructed subjects to complete the task as fast as possible, they optimized their behavior for speed and thereby the accuracy of the system was compromised. We verified this effect in following work which is discussed in Chapter 9. While we randomized trials in order to compensate for learning effects, we did observe other effects which affected the results. For instance, if the subject is first introduced to the EyePoint with no focus points condition and then later to the EyePoint condition, by the time they begin the test for the EyePoint condition, they already feel confident about the technique and therefore tend to try to speed up, thereby increasing the error rate. In addition, for real-life usage one might expect an improvement over time as users adapt to using gaze-based pointing. This would also reduce the cognitive load of pressing the right hotkey for the desired action.

Survey results showed that subjects strongly preferred EyePoint over using the mouse even though the mouse was more accurate and faster. This is contrary to our empirical findings. When we asked subjects why this was so, they reported that they felt that EyePoint was more natural since they were already looking at the target when they wanted to point. Some subjects stated that they felt it was a more "lightweight" pointing technique than the mouse. It also allowed them to keep their hands on the keyboard and was therefore much faster for mixed tasks that involved typing and pointing. They also felt that EyePoint reduced the risk of repetitive stress injury from using the mouse.

In the qualitative evaluation subjects also reported that they found that the gaze-based techniques required more "focus" and more "concentration" and therefore found the studies to be fatiguing over time. It should be noted that each subject participated in the study for one hour during which they had to click on approximately 500 targets with their eyes and about 100 targets with the mouse. In standard use, we do not expect users to be engaging is such intense usage, which should alleviate any fatigue issues. However, the fatigue, if any, caused by a gaze-based pointing technique would need to be analyzed in more depth in a long term study with normal use patterns.

3.5 Summary

EyePoint presents a practical and innovative interaction technique that combines the use of gaze- and key-based activation into a single look-press-look release action. This transforms a two-step refinement process into a single fluid action and prevents overloading the visual channel while still using gaze-based target refinement. EyePoint makes gaze-based pointing equally compelling for use by both disabled and able-bodied users.

4 Scrolling

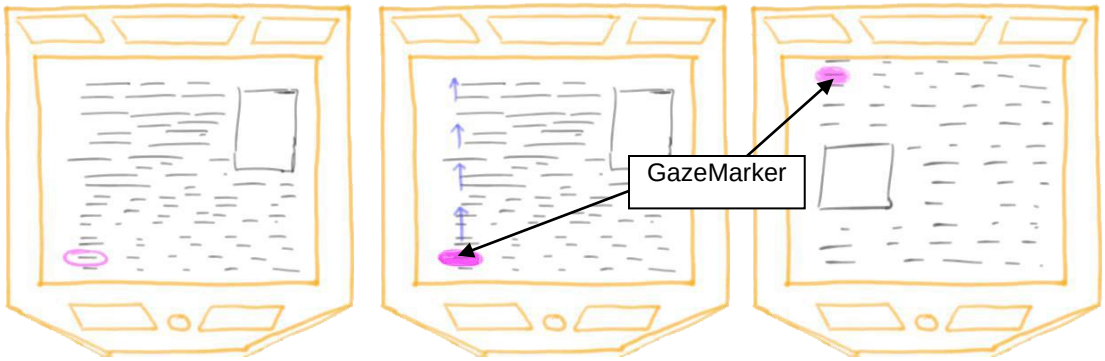
Scrolling is an essential part of our everyday computing experience. It is essential for viewing information on electronic displays, which provide a limited viewport to a virtually unlimited amount of information. Contemporary scrolling techniques rely on the explicit initiation of scrolling by the user. Considerable prior work [30, 46, 65, 109, 119] has been done in evaluating various techniques and devices for scrolling.

The act of scrolling is tightly coupled with the user's ability to absorb information via the visual channel, i.e. the user initiates a scrolling action to inform the system that he/she is now ready for additional information to be brought into view. We therefore posit that gaze information can be an invaluable source of contextual information making it a natural choice for enhancing scrolling techniques.

By understanding the characteristics of reading patterns and how users consume visual information [24, 87, 89] it is possible to devise new techniques for scrolling, which can either use gaze-information to automatically control the onset and the speed of scrolling or use gaze information passively to augment manual scrolling techniques.

Individual differences in reading and scanning patterns may result in no one technique being suitable for all users. We therefore explore a range of techniques in this chapter that may satisfy different user preferences. We designed and implemented techniques for both manual and automatic scrolling on a Tobii 1750

Portions of this chapter were originally published by the author, Andreas Paepcke and Terry Winograd in [64], and by the author and Terry Winograd in [63] and [62] (submitted to UIST 2007).



A. The user's gaze position right before pressing the Page Down Key.

B. When the user presses the Page Down Key, the region below the user's eye gaze is highlighted with a GazeMarker and scrolled to the top of the viewport.

C. The motion of the GazeMarker directs the user's gaze up to the top of the page keeping it positioned where the user was reading. The GazeMarker slowly fades away over a couple of seconds.

Figure 26. The Gaze-enhanced Page Up / Page Down approach addresses the limitations of current Page Up and Page Down Techniques by Positioning the region under the user's gaze at the bottom or top of the page respectively.

[107] eye tracker. We also introduce the use of off-screen gaze-actuated buttons or hotspots that allow users to explicitly control document navigation.

4.1 Manual Scrolling

Manual scrolling techniques such as the use of the Page Down key can be improved by using gaze information as an augmented input for the scrolling action. In this section we first identify a common problem with the use of the Page Down action and propose a gaze-enhanced solution to this problem.

4.1.1 The Page Up / Page Down Problem

The implementation of Page Up and Page Down on contemporary systems is based on the expectation that the user will press the page down key when he or she is looking at the last line on the page. However, observing users revealed that users often initiate scrolling in anticipation of getting towards the end of the content in the viewport. This results in users pressing page down before reaching the last line of the text. Consequently, the text the user was looking at scrolls out of view off the top of the viewport. This necessitates a fine-tuning of the scrolling movement to bring

the text back into view. In addition, most users tend to lose track of where they were reading once the page scrolls and must reacquire their position in the text.

4.1.2 Gaze-enhanced Page Up / Page Down

We propose a new approach for a gaze-enhanced page-down which uses a *GazeMarker* to always keep user's eyes on the text they were reading even through page transitions. In this approach, the user's eye gaze on the screen is tracked. When the user presses the page down key, the region where the user was looking immediately before pressing the page down key is highlighted. We call this highlight a "GazeMarker". The page is then scrolled such that the highlighted region becomes the topmost text shown in the viewport (Figure 26). Since the highlight appears immediately before the page scrolls and then moves up in the viewport, the user's gaze naturally follows the highlight. This ensures that the user's gaze is kept on the text he or she was reading and minimizes the need to reacquire the text after scrolling. The GazeMarker slowly fades away within a few seconds.

This technique ensures that the content the user is looking at is brought to the top of the page. By implication, the amount of the page that is scrolled is also controlled by the position of the user's gaze when the Page Down key is pressed. In addition the scrolling motion of the page is controlled so that the GazeMarker is animated up towards the top of the page (as opposed to a discrete jump) in order to smoothly carry the user's eyes to the new reading location.

4.2 Automatic Scrolling

The design of any automatic scrolling techniques must overcome two main issues: a) the Midas Touch problem, b) controlling the speed at which the content is scrolled. We address each of these problems below.

4.2.1 Explicit Activation/Deactivation

PC keyboards include a vestigial Scroll Lock key, which the vast majority of users have never used. The historical function of the Scroll Lock key was to modify the behavior of the arrow keys. When the scroll lock mode was on, the arrow keys

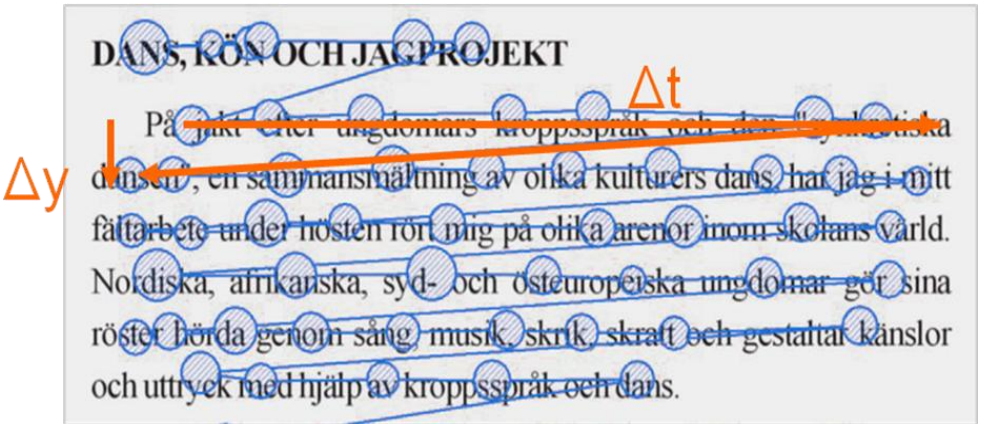


Figure 27. Estimation of reading speed. Vertical pixels viewed per second = Δy/Δt (base image of gaze pattern while reading taken from wikipedia.org).

would scroll the contents of a text window instead of moving the cursor. The Scroll Lock key is a defunct feature in most modern programs and operating systems.

To overcome the Midas Touch problem we chose to use explicit activation of the automatic scrolling techniques by putting the Scroll Lock key back into use. The user toggles the automatic scrolling on and off by pressing the Scroll Lock key on the keyboard.

4.2.2 Estimation of Reading Speed

For several of the techniques presented in this chapter, it is useful to be able to measure the user's vertical reading speed. Previous work [24, 89] has shown that the typical eye movements (fixations and saccades) for a subject reading text conforms to Figure 27. Beymer et al. [24] present an estimate of reading speed based on forward-reads. For our use – to control scrolling — it is more interesting to measure the speed at which the user is viewing vertical pixels. This can be estimated by measuring the amount of time for the horizontal sweep of the user's eye gaze (Δt) and the delta in the number of vertical pixels during that time (Δy). The delta in the vertical pixels divided by the amount of time for the horizontal sweep (Δy/Δt) provides an instantaneous measure of "reading speed" (Figure 27). A smoothing algorithm is applied to the instantaneous reading speed to account for variations in column sizes and the presence of images on the screen. The resulting smoothed

reading speed provides a best guess estimate of the rate at which the user is viewing information on the screen.

We present three scrolling techniques that start and stop scrolling automatically, depending upon the user's gaze position. The techniques differ in the details of whether the content is scrolled smoothly or discretely. The automatic scrolling techniques presented in this chapter, scroll text only in one direction. This was a conscious design choice to overcome the Midas Touch problem. Scrolling backwards or navigating to a particular section of the document can be achieved either by using manual methods or by using off-screen navigation buttons.

4.2.3 Eye-in-the-middle

The eye-in-the middle technique for automatic scrolling measures the user's reading speed while dynamically adjusting the rate of the scrolling to keep the user's gaze in the middle third of the screen (Figure 28). This technique relies on accelerating or decelerating the scrolling rates to match the user's instantaneous reading speed. It is best suited for reading text-only content since the user's scanning patterns for images included with the text may vary. This technique requires that the user read text while it is scrolling smoothly, similar to a teleprompter.

4.2.4 Smooth scrolling with gaze-repositioning

This automatic scrolling approach relies on using multiple invisible threshold lines on the screen (Figure 29). When the user's gaze falls below a *start threshold*, the document begins to scroll slowly. The scrolling speed is set to be slightly faster than the user's reading speed so as to gradually move the user's gaze position towards the top of the screen. When the user's gaze reaches a *stop threshold*, scrolling is stopped (text is stationary) and the user can continue reading down the page normally. If the user's gaze falls below a *faster threshold*, the system begins to scroll the test more rapidly. The assumption here is that either the scrolling speed is too slow or the user is scanning and therefore would prefer that the content scroll faster. Once the user's gaze rises above the start threshold, the scrolling speed is reduced to the normal

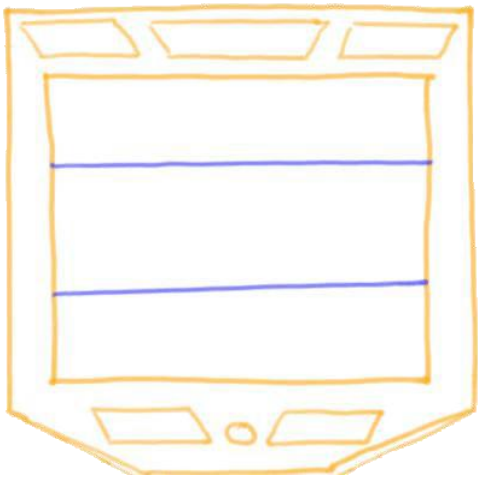


Figure 28. The eye-in-the-middle automatic scrolling technique adjusts the scrolling speed to match the user's reading speed and tries to keep the user's eyes in the middle third of the screen.

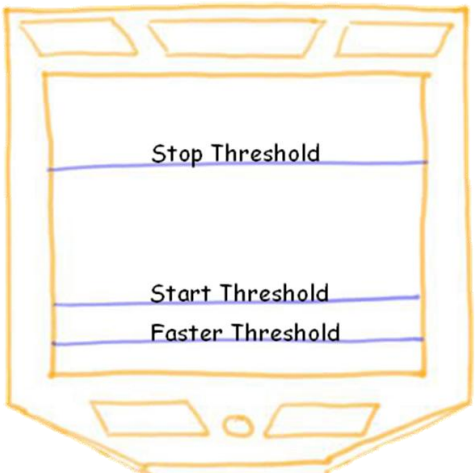


Figure 29. The smooth scrolling with gaze-repositioning technique allows for reading and scanning of content. Scrolling starts and stops depending on the position of the user's gaze with respect to invisible threshold lines on the screen.

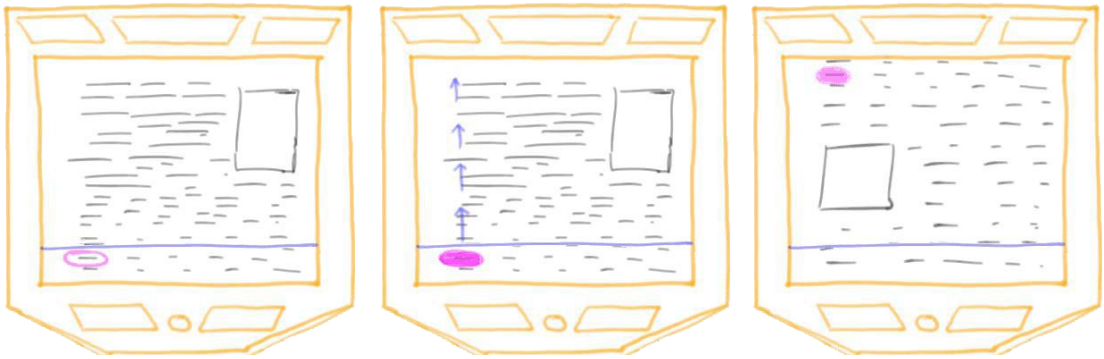
scrolling speed. The scrolling speed can be adjusted based on each individual's reading speed.

In our implementation, the position of the threshold lines was determined based on user feedback. In particular, placing the stop threshold line higher on the screen resulted in subjects in our pilot study worrying that the text would "run away" before they would have the chance to finish reading it. We therefore lowered the stop threshold to one-third the height of the screen so that scrolling would stop before the users became anxious. In addition, whenever scrolling is started or stopped, it is done by slowly increasing or decreasing the scrolling rate respectively. This is done to make the state transitions from continuous and fluid.

This approach allows for both reading and scanning, however, in this approach while the user is reading, sometimes the text is moving and other times the text is stationary.

4.2.5 Discrete scrolling with gaze-repositioning

The discrete scrolling with gaze-repositioning approach leverages the gaze-enhanced Page Up / Page Down technique for manual scrolling and extends it by



A. The user's gaze position when the eye gaze drops below the scrolling threshold.

B. If the user's gaze stay below the threshold for the duration of a micro-dwell (~150-200ms) the system issues a Page Down command, which results in the GazeMarker being drawn.

C. The motion of the GazeMarker directs the user's gaze up to the top of the page keeping it positioned where the user was reading. The GazeMarker slowly fades away over a couple of seconds.

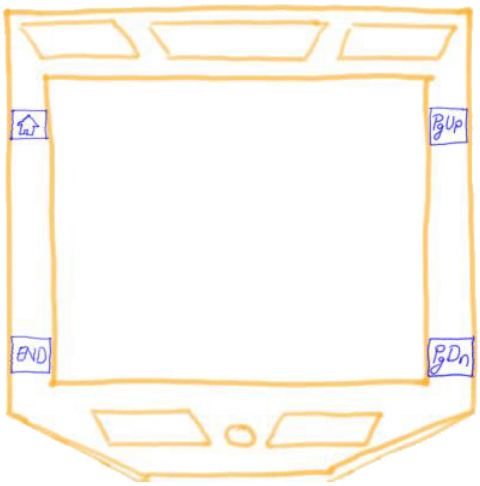
Figure 30. The discrete scrolling with gaze-repositioning leverages the gaze-enhanced Page Up / Page down and triggers a Page Down event when the users gaze falls below a threshold line for a specified duration.

adding an invisible threshold line towards the bottom of the screen. When the user's eyes fall below the threshold the system issues a page down command which results in the GazeMarker being drawn and the page being scrolled (Figure 30). The user's gaze must stay below the threshold for a micro-dwell duration (~150-200ms) before the event triggers. This minimizes the number of false activations from just looking around at the page and disambiguates scanning the screen from reaching the end of the content on the screen while reading. The scrolling motion happens smoothly to keep the user's eyes on the GazeMarker, but fast enough for the scrolling to appear as if it occurred a page at a time.

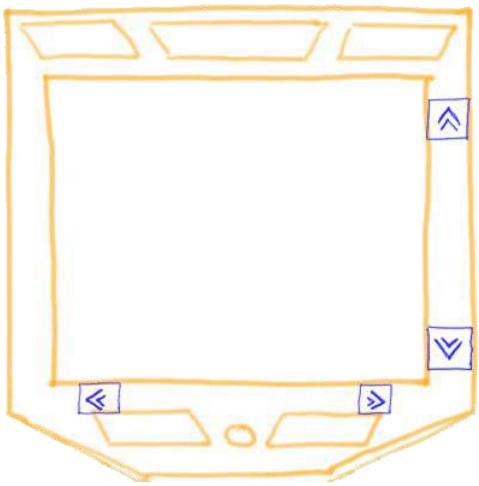
This approach ensures that users read only when the content is stationary (in contrast to the previous automatic scrolling approaches).

4.3 Off-Screen Gaze-Actuated Buttons

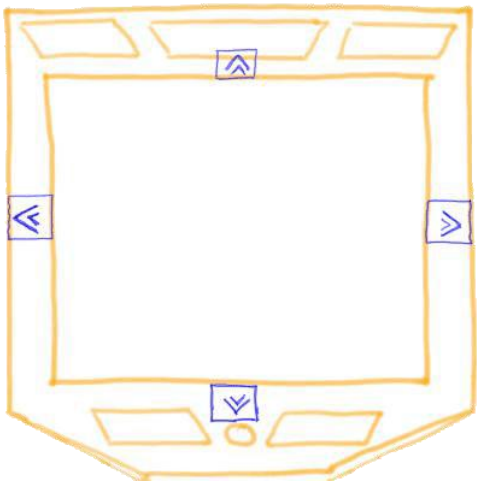
The Tobii eye-tracker provides sufficient field of view and resolution to be able to clearly identify when the user is looking beyond the edges of the screen at the bezel. This provides ample room to create gaze-based hotspots for navigation



A. Home, End, Page Up and Page Down buttons activated by dwell (400-500 ms)



B. Scrolling buttons activated by a micro-dwell (150-200ms) provide continuous input while the user is looking at them.



C. Scrolling buttons located in the center to be aligned with the user's gaze direction.

D. 8-way panning regions activated by looking slightly off-screen for a micro-dwell duration (150-200ms)

Figure 31. Off-screen gaze-actuated buttons/hotspots for document navigation and control. Buttons which trigger discrete events (Home, Page Down etc.) use a dwell-based activation. Hotspots that have a more continuous action (scroll up etc.) use a micro-dwell based activation.

controls. We implemented several variations of off-screen gaze-actuated buttons for document navigation as seen in Figure 31.

Figure 31A shows the use of off-screen targets for document navigation commands such as Home, End, Page Up and Page down. Figure 31B and Figure 31C show two alternative placements of scroll bar buttons. Figure 31D shows the

placement of hotspots for an eight-way panning approach. We used this approach to implement a prototype of a *gaze-controlled virtual screen* where the total available screen real-estate exceeds the visible portion of the screen (See .Chapter 8).

4.3.1 Dwell vs. Micro-Dwell based activation

Document navigation requires either a discrete one time activation (such as Home, End, Page Up and Page Down buttons), or a more continuous or repetitive action (such as the cursor keys or the controls on a scroll bar). To accommodate the different forms of these actions we implement two different activation techniques. The first, dwell-based activation, triggers only once, when the user has been staring at the target for at least 400-500 ms. For actions that require continuous input, we chose to use a micro-dwell based activation when the user has been staring at the target for at least 150-200 ms. The dwell based activation triggers the event just once. The micro-dwell based activation repeats the command or action till the user stops looking at the associated hot-spot.

4.4 Evaluation

We conducted informal user studies to gauge user reaction to the gaze-enhanced scrolling techniques described above. Feedback from the user studies was used to help refine the techniques and motivated key design changes (such as the introduction of micro-dwell). Detailed comparative quantitative evaluation of the each of the scrolling techniques was not performed since any such evaluation would be plagued by differences in subjects' reading style and speed. In addition, users may prefer one approach over another depending upon their subjective preferences.

4.4.1 Gaze-enhanced Page Up / Page Down

Informal user studies with 10 users indicated that subjects unanimously preferred the gaze-enhanced Page Up/Page Down technique over the normal Page Up / Page Down. Subjects reported that the system eliminated the need to reposition the text after pressing page down, consistently highlighted the region that they were looking at and kept their eyes on the content even after it scrolled.

4.4.2 Smooth-scrolling with Gaze-Repositioning

To evaluate the smooth scrolling with gaze-repositioning technique we conducted a two part study with 10 subjects (6 male, 4 female). The average age of the subjects was 22 years. None of the subjects wore eye-glasses, though two did use contact lenses. None of the subjects were colorblind. English was the first language for all but two of the subjects. On average, subjects reported that they did two-thirds of all reading on a computer. The scroll-wheel was the most-favored technique for scrolling documents when reading online, followed by scroll bar, spacebar, page up / page down or arrow keys.

In the first part of the study, subjects were told that they would be trying a new gaze-based automatic scrolling technique to read a web page. For this part of the study, subjects were given no explanation on how the system worked. To ensure that subjects read each word of the document, we requested them to read aloud. We did not test for comprehension of the reading material since we were only interested in the subjects being able to view the information on the screen. Once subjects had finished reading the page, they were asked to respond to questions on a 7-point Likert scale.

In the second part of the study, we explained the technique's behavior to the subjects and showed them the approximate location of the invisible threshold lines (Figure 2). Subjects were allowed to practice and become familiar with the approach and then asked to read one more web page. At the conclusion of this part subjects again responded to the same set of questions as before.

Figure 32 summarizes the results from the study showing the subjects' responses in each of the two conditions (without explanation and with explanation).

Subjects' feeling that *scrolling started when they expected it to*, and that *they were in control* show increases in the with-explanation condition. For all other questions regarding comfort, fatigue and user preference there was no significant change in the subjects' responses across the two conditions. Subjects' response on the reading speed was mostly neutral, suggesting that they felt the scrolling speed was reasonable. While the differences in the results for reading speed in the two conditions are not significant, results do show that subjects were more comfortable

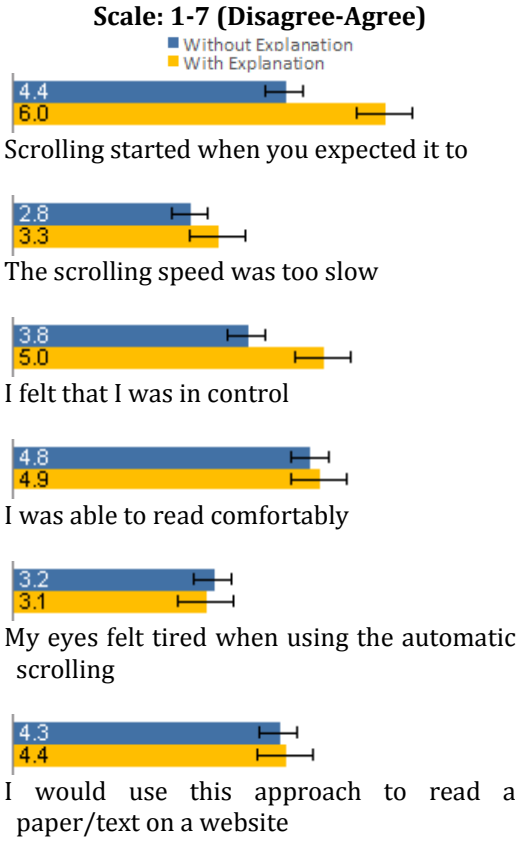


Figure 32. Subjective evaluation results for *Smooth scrolling with gaze-repositioning* in two conditions (with and without explanation of how the system works). Error bars show Standard Error.

(more neutral) in the with-explanation condition since they were more familiar with the operation and less worried about the content running off the screen.

Several subjects commented that they found reading text while it was scrolling to be disconcerting at first, but then became more comfortable with it once they realized that the text would not scroll off the screen and would stop in time for them to read. It is conceivable that, like a teleprompter, subjects may be comfortable with reading moving text with practice.

4.4.3 Discrete scrolling with Gaze-repositioning

In informal user studies with 10 subjects, users indicated that they preferred the discrete scrolling with gaze repositioning approach over other automatic scrolling techniques since it required them to read only when the text was stationary.

4.5 Summary

We presented several techniques for gaze-enhanced scrolling which include augmenting existing manual scrolling techniques with gaze input and variations of automatic gaze-based scrolling techniques. We also introduced the use of off-screen gaze-actuated buttons or hotspots.

Gaze enhanced scrolling has the potential to radically reduce the number of scrolling actions users need to perform in order to surf the web or consume other information displayed in electronic form. With the inclusion of cameras into current

display devices [2] and the impending reduction in cost of eye-tracking technology (Chapter 10), gaze-based scrolling techniques will increase in importance and provide users with a natural alternative to current approaches.

5 Application Switching

Application switching is an integral part of our daily computing experience. Users are increasingly engaged in multiple tasks on their computers. This translates into a larger number of open windows on the desktop. On average, users have 8 or more windows open 78.1% of the time [49]. While there has been extensive research in the area of window managers and task management [9, 35, 44, 90, 91, 100], few of these innovations have been adopted by commercially available desktop interfaces. Clicking on the iconic representation of the application in the taskbar/dock or using Alt-Tab/Cmd-Tab have been the de facto standard for application switching for several years. Probably the most notable advance has been the introduction of the Exposé [1] feature in Apple's Mac OS X operating system.

Exposé allows the user to press a key (default *F9*) on the keyboard to instantly see all open windows in a single view (Figure 33). The windows are tiled, scaled down and neatly arranged so that every open application is visible on the screen. To switch to an application the user moves the mouse over the application and then clicks to bring that application to the foreground. Every open application window is restored to its original size and the window clicked upon becomes the active window.

Windows Vista includes new application switching features. The taskbar in Windows Vista displays live thumbnail views of open applications when the user hovers the mouse on the taskbar. Alt-Tab functionality has been updated with

Portions of this chapter were originally published by the author, Andreas Paepcke and Terry Winograd in [60] and by the author and Terry Winograd in [63].

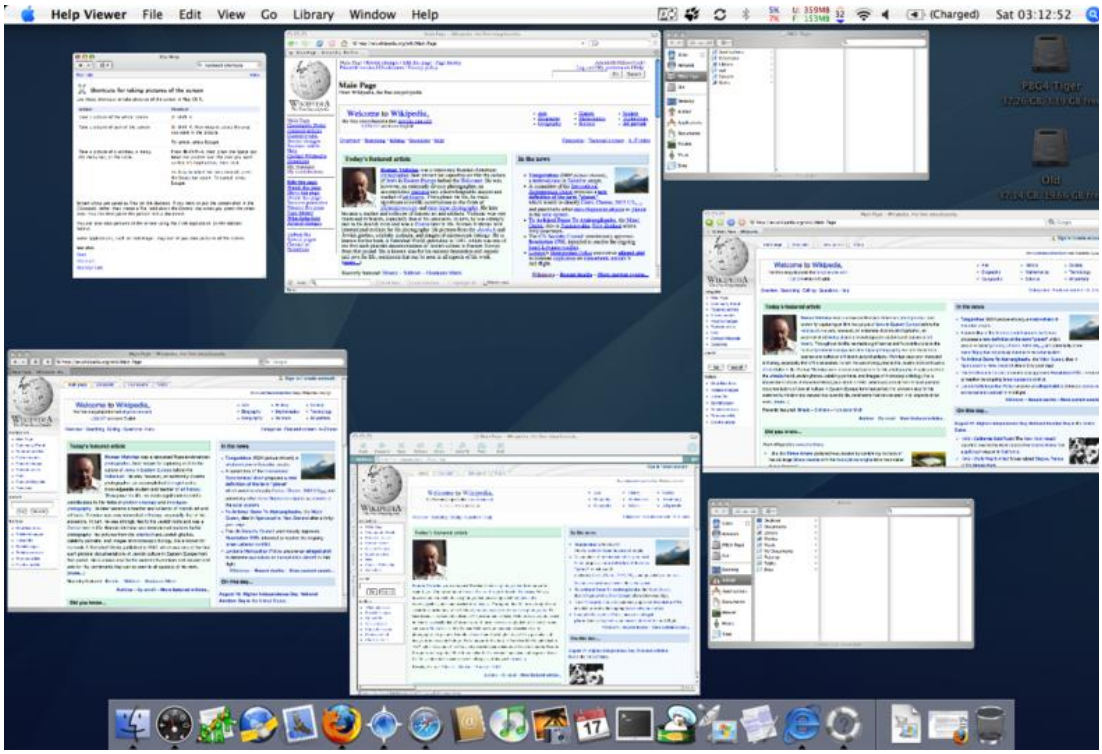


Figure 33. Exposé view of open applications (image from wikipedia.org).

Windows Flip and Flip3D [8]. Flip allows users to view live thumbnails of the applications as they press Alt-Tab. Flip3D shows a stacked 3-D visualization of the applications with live previews and allows users to cycle through applications with the scroll wheel or the keyboard.

In this chapter we introduce and evaluate a technique that uses eye gaze for the selection of the desired window in conjunction with Exposé-like visualization of the open application windows.

5.1 Background and Related Work

Application switching has been necessary ever since computers were able to multi-task. In the days of command-line UNIX this was achieved with the commands *bg*, *fg* and *jobs*. With the advent and ubiquity of graphical interfaces and the desktop metaphor, application switching has become commonplace.

The techniques for application switching can be categorized into three approaches: *Temporal*, *Spatial* and *Hybrid*. Temporal approaches sort windows based

on their time of last access, and therefore the order in which the windows are shown to the user changes depending on which application was last used. Spatial approaches may use an initial ordering based on when the application was launched or where it is located on the screen. The relative order of applications in the application switching view does not change unless there is a change in the number of open applications or the spatial location of an application. Hybrid approaches use a combination of spatial and temporal characteristics of the open application windows.

Alt-Tab is a temporal approach. It organizes applications in the order in which they were last used. Users are able to cycle through the list of applications by sequentially stepping through the list until they arrive at the application they desire. Such techniques make best use of the user's temporal memory and make switching among a limited number of tasks very efficient.

The organization of window buttons on the Taskbar or in the dock follows the spatial approach. The user can access any open application directly by clicking on a button/iconic representation of the application. The location of the iconic representation of the application on the Taskbar is fixed and therefore this approach takes advantage of the user's spatial memory.

Exposé uses a spatial layout to arrange the open application windows in a visual representation. It also uses heuristics to keep the current application in the center of the visualization and to arrange windows based on their relative spatial position [54]. While the location of the windows in the Exposé view may change, it is relative to the spatial locations of the open applications.

Hybrid approaches, which use a temporal ordering but allow for random access (as opposed to the sequential access of Alt-Tab) are becoming more popular. The Windows XP PowerToy TaskSwitch [5] shows a thumbnail of the current application and allows users to either cycle through the open applications by repeatedly pressing Alt-Tab or to use the mouse to click on the icon for the desired application. This functionality is also embodied in the implementation of Flip and Flip3D in Windows Vista.

In EyeWindows [40], Fono and Vertegaal explore two window management techniques for *non-overlapping* windows which use the elastic windowing algorithm

Figure 34. Fono et al.'s EyeWindows technique for switching between non-overlapping windows using eye gaze. When the user looks at a particular window, it is restored to its full dimension while all other windows are distorted using an elastic windowing algorithm.

to spatially lay out application windows. The EyeWindows approach suffers from two major drawbacks. First, the technique is limited to use with non-overlapping windows (all techniques described previously allow overlapping windows). Secondly, switching between applications in EyeWindows requires windows to be zoomed in and out, which can be visually distracting for the user.

Several research systems [9, 35, 44, 90, 91, 100] have been proposed with novel window management and task switching techniques. Our gaze-based selection technique can complement the techniques in other research systems. For this chapter we focus on task/window switching techniques in commercially available and commonly used operating systems.

5.2 Design Rationale

We hypothesized that it would be preferable to switch between applications simply by looking at the application the user wants to switch to – a concept similar to EyeWindows. Exposé in Mac OS X provides a well established and highly usable technique for switching between applications. Unfortunately, the research literature is lacking a scientific evaluation of different application switching techniques (Alt-Tab/Cmd-Tab vs. Taskbar/Dock vs. Exposé vs. Flip/Flip3D). Anecdotal evidence, however, suggests that the Exposé approach is preferred by users for random access

Figure 35. Using EyeExposé – Pressing and holding the EyeExposé hotkey tiles all open applications on the screen. The user simply looks at the desired target application and releases the hotkey to switch applications.

to open applications, while the Alt-Tab/Flip approach is preferred for access to the last used application.

To use Exposé, users press a hotkey (*F9*) and then use the mouse to point at and click on the desired application. Using this approach requires both the keyboard and the mouse, whereas with the Alt-Tab approach, the user can switch applications using only the keyboard. Exposé does allow users to activate application switching by moving the mouse to a designated hotspot (one corner of the screen) and then clicking on the desired application. This still requires users to move their hands from the keyboard to the pointing device.

As discussed in Section 2.5, the accuracy of eye trackers is insufficient to be able to point to small targets. In Chapter 3, we presented a technique that overcomes this limitation of eye trackers by using a secondary gaze fixation in a magnified view. By contrast, for the purpose of application switching, the size of the tiled windows in Exposé is usually large enough for eye-tracking accuracy to not be an issue. Therefore, direct selection of the target window using gaze is possible.

5.3 EyeExposé

Our system, EyeExposé, combines a full-screen two-dimensional thumbnail view of the open applications with gaze-based selection. EyeExposé has been implemented on Microsoft Windows using a Tobii 1750 eye gaze tracker for the gaze-based selection.

Figure 35 show how EyeExposé works. To switch to a different application, the user presses and holds down a hotkey. EyeExposé responds by showing a scaled view of all the applications that are currently open on the desktop. The user simply looks at the desired target application and releases the hotkey.

Whether the user relies on eye gaze or the mouse, the visual search task to find the desired application in the tiled view is a required prerequisite step. By using eye gaze with an explicit action (the release of the hotkey) we can leverage the user's natural visual search to point to the desired selection. If we analyze the actions needed by the user to select a target window using the mouse, the total time would be:

$$T_{mouse} = t_{activation} + t_{visual\ search} + t_{acquire\ mouse} + t_{acquire\ cursor} + t_{move\ mouse} + t_{click\ mouse}$$

where $t_{activation}$ is the time for the user to press the hotkey or move the mouse to a corner of the screen to activate application switching; $t_{visual\ search}$ is the amount of time it takes the user to locate the target on the screen; $t_{acquire\ mouse}$ is the amount of time it takes the user to move the hands from the keyboard to the mouse; $t_{acquire\ cursor}$ is the amount of time to locate the cursor on the screen and $t_{move\ mouse}$ and $t_{click\ mouse}$ are the times to move and click the mouse button respectively.

We assume here that the visual search only needs to happen once since short term spatial memory enables the user to remember where the mouse needs to be moved. By contrast, the total time for selection using EyeExposé should be:

$$T_{eyeexposé} = t_{activation} + t_{visual\ search} + t_{release}$$

where $t_{release}$ is the time to release the hotkey. We expect $t_{release}$ to be considerably lower than ($t_{acquire\ mouse} + t_{acquire\ cursor} + t_{move\ mouse} + t_{click\ mouse}$). Gaze-based application switching can therefore result in time savings by eliminating several of the cognitive and motor steps and replacing them with the single action of releasing the hotkey/trigger.

However, efficiency is not the only measure of the success of a particular interaction. The affect generated by that interaction and the subjective user experience is a key measure of the success and factor for adoption [81]. We hypothesized that users would like using EyeExposé since it provides a very simple

and natural way of switching between applications. Therefore, we also chose to evaluate the user's subjective experience when using the gaze-based application switching.

5.4 Evaluation

To evaluate EyeExposé, we conducted a user study with 20 subjects. Subjects were mostly graduate students and professionals and as such were experienced computer users who used various ways of switching applications. When asked which application switching technique they used the most, subjects reported that 46% used Alt-Tab, 38% used the Taskbar, 13% used Exposé and 4% used some kind of Virtual Desktop. Our subject pool had 13 males and 7 females with an average age of 28 years. 14 subjects did not require any vision correction. Four subjects wore contact lenses and 2 wore eyeglasses. None of the subjects were colorblind. Subjects had an average of 15 years of experience using the mouse.

5.4.1 Quantitative Evaluation

We tested speed and accuracy for 4 different application switching techniques — the Taskbar, Alt-Tab, an Exposé-clone with mouse based selection and EyeExposé in a 4 by 3 within-subjects experiment (4 techniques, 3 number of windows). For each application switching technique (Taskbar, Alt-Tab, Exposé and EyeExposé) we conducted trials with 4, 8 and 12 open windows to account for the number of windows being below, at and above average [49]. The order of the trials for each combination of technique and number of windows was varied to counterbalance and minimize learning effects.

Our original experiment design used real application windows such as Word, Excel and PowerPoint as the target windows. We believed that subjects would be easily able to recognize real application windows. However, our pilot studies revealed that subjects found it difficult to recognize applications in a testing environment, which was not based on their own work context. We therefore chose to use colored windows to reduce the cognitive load and the search time for subjects to identify the right target window.

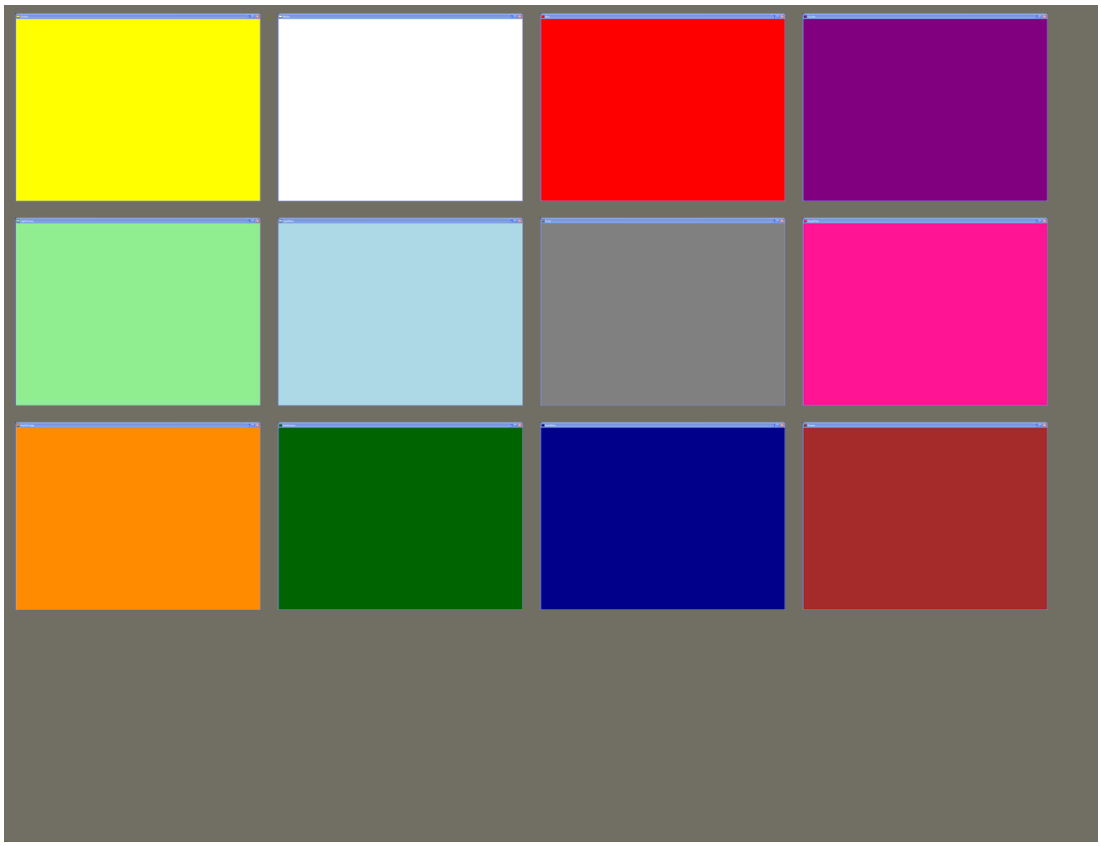


Figure 36. Exposé/EyeExposé view of 12 open windows, each window being a distinct color (yellow, white, red, purple, light green, light blue, grey, pink, orange, dark green, dark blue and brown).

Each window was a unique color and the name of the window matched the color of the window. Colors were carefully chosen to maximize recognition of the color by name (Figure 36). We verified that subjects were able to easily identify windows by the name of the color in our pilot studies. The window icon that appeared on the Taskbar and in the Alt-Tab view matched the color of the window. Maintaining the color consistency on window icons and names ensured that the Taskbar and Alt-Tab techniques also benefited from the use of colors. The final design used a unique icon and color for each window and was therefore biased in favor of the Taskbar and Alt-Tab since there were no applications with the same icon repeated – a condition that happens often in normal use. In addition, the use of colored windows does not provide any additional contextual information that may otherwise be available in a real-use scenario. This also biases against Exposé and

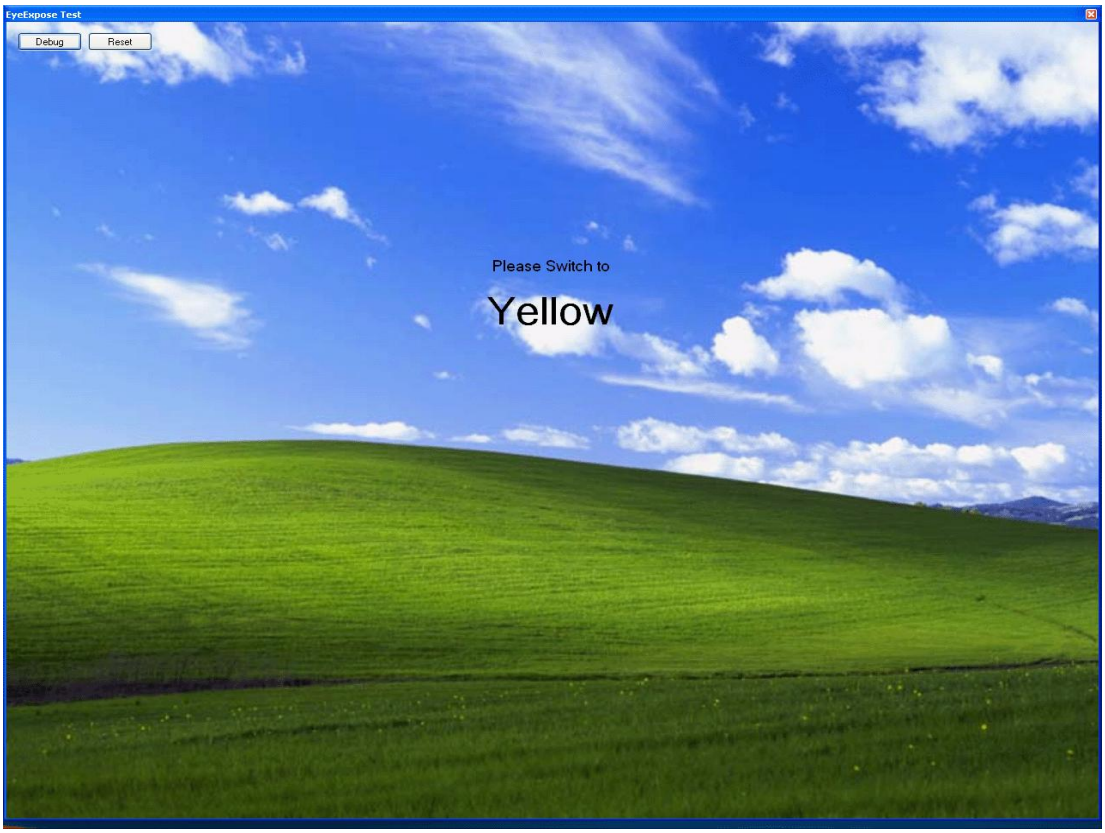


Figure 37. Instructions for which window to switch to next were shown on a second monitor.

EyeExposé, since those techniques have the ability to display more context than just color when the visualization is activated.

We used our implementation of an Exposé clone to perform the tests in a Windows environment and to instrument the code to capture timing data. Our implementation differs from the Mac OS X implementation in that EyeExposé uses a simpler layout algorithm, ordering windows heuristically based on the height, width or area of the window. EyeExposé does not optimize window placement based on the spatial location of windows (not a variable in the study since all the applications were full-screen). The eye-based selection and mouse-based selection both used the same underlying code and layout algorithm and therefore the only difference in the setup was the selection technique used.

In the Exposé and EyeExposé conditions the placement of windows was randomized for half of the subjects: each time the user activated the view, the order

Figure 38. Taskbar in each of the 4, 18 and 12 window conditions.

of the windows changed. For the other half of the subjects, the order of the windows remained the same as in previous trials.

In the Taskbar condition, users had to click on the application button on the Taskbar and then click on a randomly placed "Next" button. This was done to force users to move the mouse away from the Taskbar before the subsequent trial. For all other techniques, users were prompted with the name of the next target window as soon as they completed the current trial. The number of windows on the Taskbar never exceeded a threshold that would cause it to add a second line with a scroll button (Figure 38).

The experiment used a Tobii 1750 (17" LCD) eye gaze tracker as the primary display. The screen resolution was set to 1280x1024 pixels. The test environment presented a window on a second monitor placed to the right of the primary screen, which displayed the instructions for the user (Figure 37).

We recorded the amount of time it took a user to select the target window,

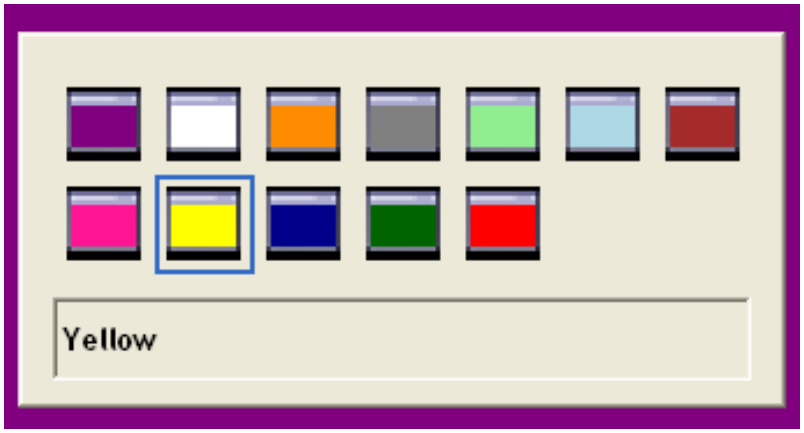


Figure 39. Alt-Tab view of 12 open application windows.

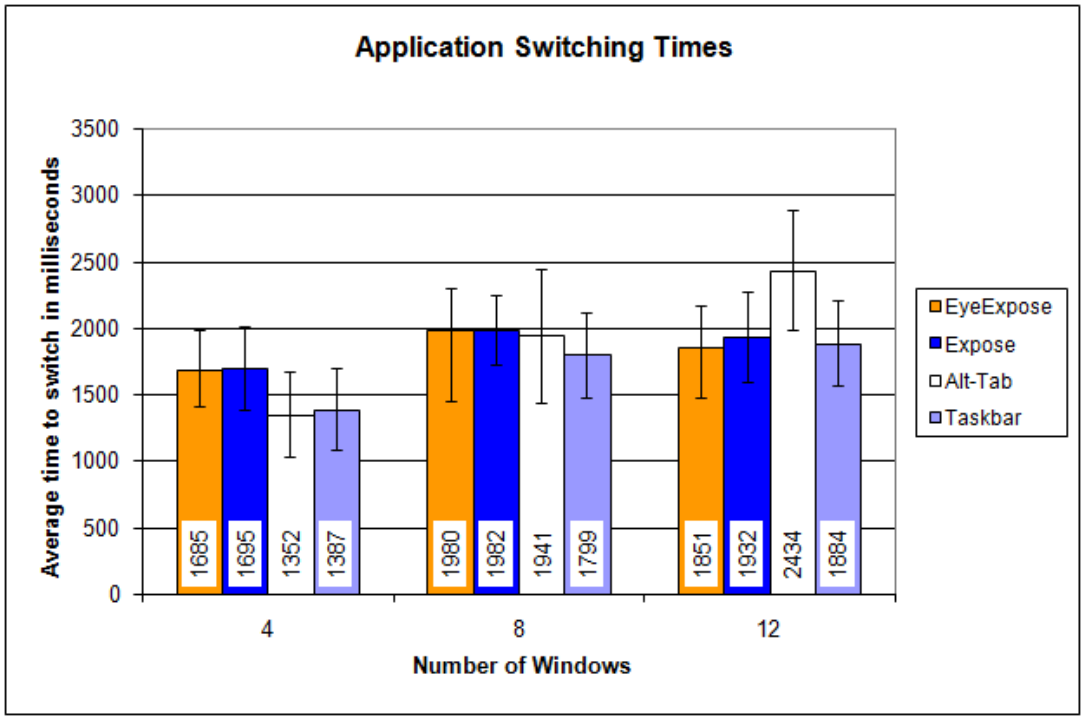


Figure 40. Quantitative evaluation results – time to switch between applications.

starting from the time the instruction appeared on the screen. If the user switched to an incorrect window, we recorded an error. In each of the 12 conditions (technique x number of windows), users were asked to switch windows until they had completed 20 successful trials.

5.4.2 Qualitative Evaluation

At the end of the study, subjects completed a questionnaire in which they ranked each of the four techniques on dimensions of perceived speed, accuracy, ease of use, and preference.

5.5 Results

Figure 40 shows the time to switch between applications in the quantitative evaluation. A repeated measures ANOVA for number of windows and technique showed a significant effect for number of windows ($F(2,38)=55.07$, $p < .01$), for technique ($F(1.9,36.9)=5.29$, $p < .01$, Greenhouse-Geisser corrected) and interactions between number of windows and technique ($F(6, 114) = 22.22$, $p < .01$). Contrast

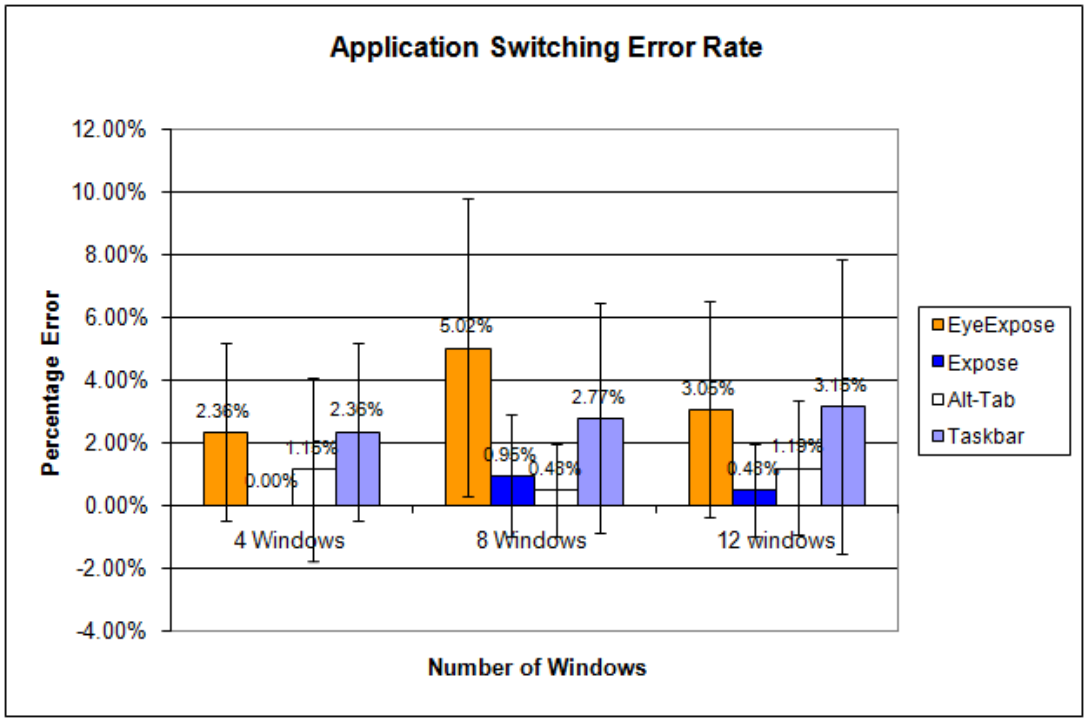


Figure 41. Quantitative study results - error rate.

analyses showed no significant difference between the Exposé and EyeExposé techniques. For the 4-window condition, as we expected, Alt-Tab was faster than Exposé and EyeExposé. For 8 windows, switching times for all four techniques were about the same, with the Taskbar showing a slight (but significant) advantage over Exposé. For the 12 window condition, EyeExposé had the lowest switching time (significant compared to Alt-Tab only).

Figure 41 shows the error rates from the study. It should be noted that the maximum error was less than 5%, or 1 error in over 20 trials. Several subjects performed the trials with no errors at all. The error rate distributions were highly non-normal. We therefore performed a Friedman's (non-parametric) ANOVA to compare participants' error rates. Results are shown in Table 1. The first row shows the ANOVA results. The second row shows the result of Bonferroni corrected pair-wise comparisons between the conditions. Only the listed condition pairs exhibited significant differences in error rates. As expected, the error rates for Exposé were the smallest since it provides large, easily recognizable targets, which are clicked on with a mouse.

	Number of Windows		
	4	8	12
Friedman ANOVA	$X^2(3)=13.1$; $p<.01$	$X^2(3)=18.0$; $p<.01$	$X^2(3)=11.7$; $p<.01$
Significant Differences (pairs)	EyeExposé Exposé Exposé Taskbar	EyeExposé Alt-Tab EyeExposé Exposé	EyeExposé Exposé Exposé Taskbar

Table 1. Result of Friedman's ANOVA on errors.

Figure 42 shows a summary of the results from the qualitative evaluation where subjects ranked the four techniques for speed, accuracy, ease of use and preference. EyeExposé was the subjects' choice for speed, ease of use, and the technique they would prefer to use most if they had all four approaches available. Exposé was the subjects' choice for accuracy.

5.6 Discussion

The results from our user survey show that users have a strong preference for EyeExposé for switching between applications. Subjects reported that they felt

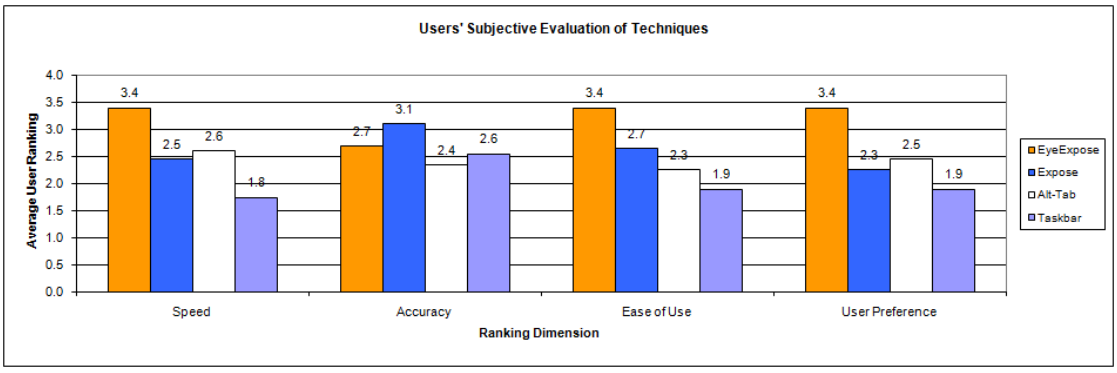


Figure 42. Qualitative evaluation results - survey ranking data.

that EyeExposé was “natural,” “faster” and “less annoying” when compared to other approaches. They also reported that they liked not having to move their hands off the keyboard to use the mouse when compared to using Exposé.

5.6.1 Performance Results

We expected the performance of EyeExposé to show a clear advantage over using the mouse in the Exposé condition. However, the results did not show a strong advantage in the time to switch and EyeExposé had a higher or comparable error rate to most other techniques.

We suspect that there are two reasons for this. First, the experimental design was such that users could keep one hand on the keyboard and the other hand on the mouse in the Exposé condition. Therefore, the cost of acquiring the mouse was zero. Furthermore, since the users already knew the location of the cursor on the screen from previous trials, the cost of acquiring the cursor was also negligible. In real-world usage, users may undertake other actions and may not remember the location of the cursor. The time to acquire the cursor in these cases would not be negligible.

As discussed in Section 3.4, we observed during the study that users could successfully parallelize some of the tasks required for pointing with the mouse. We noticed that users moved the mouse concurrently with visual search on the screen. This may be a result of the years of practice users have had with using the mouse as their primary pointing device. Therefore, the theoretical model for the time to switch we proposed earlier, which assumes a sequential ordering of the tasks is flawed due to the concurrent nature of some of the intermediate steps.

Card et al. [28] measured the device switching time from the keyboard to the pointing device to be around 360 ms. In real world use, users will incur this additional cost of acquiring the pointing device when using the Exposé technique. EyeExposé would then have a clear advantage over mouse-based selection.

Our implementation of the Exposé and EyeExposé technique took longer to show the visualization than the Alt-Tab condition or the Taskbar (always visible) due to the sluggishness of painting the screen in Windows. In the ideal scenario, the

application switching technique would be integrated into the operating system and be optimized for drawing performance.

5.6.2 Accuracy Results

The error results exhibit high variance because most subjects were able to complete the task in a given condition with zero errors. The low number of errors suggests that the performance and the user preference may dominate as factors in the decision choice for which technique users choose to use.

A closer analysis of the errors in task switching suggests that Alt-Tab is prone to errors where the user overshoots or undershoots the target window. This is because Alt-Tab's temporal window ordering strategy reorders the display of open applications each time the user selects a new application. Only 4 subjects (20%) used the *Shift* key in order to cycle backwards when using Alt-Tab.

For the Taskbar, errors usually stemmed from clicking on a neighboring window button or missing the Taskbar. This was especially true in the case of the 12 window condition where the size of the target decreased. This reaffirms the advantage of an Exposé like approach which provides large targets by using the transitional whole screen view as opposed to a permanently visible dedicated region of the screen. It should be noted that the number of windows was always low enough to show all the windows on the Taskbar without having to click the scroll button on the Taskbar.

Most errors in the EyeExposé condition occurred due to subjects picking the incorrect color (the brown color was initially confused with red by some users). The error counts therefore include both perceptual error and motor error. One reason the error rates for EyeExposé were higher than those of Exposé is because when using the mouse for selection, subjects have to first find the target and then move the mouse; the latter action of moving the mouse the target provides a second opportunity for subjects to correct any perceptual error before they click (i.e. if they were going to select the wrong color). By contrast in the case of EyeExposé, subjects can release the hotkey as soon as they think they are looking at the target. Additionally, as noted in Section 3.4, the eye tracker requires user to focus on the

target to provide an accurate reading. If the user had not focused on the target window and was only relying on peripheral vision, the data from the eye tracker will not be accurate enough to make a selection. We propose the introduction of focus points (see Section 9.3) on the EyeExposé visualization which will provide users with a visual anchor during gaze-based selection.

The Alt-Tab and Taskbar did not have as many errors since the name of the window (color) is readily visible in those techniques. In the Alt-Tab condition subjects would often notice that they had picked the incorrect target before releasing the Alt key and would therefore be able to correct the error immediately. Correcting errors in all other techniques requires subjects to repeat the trial. In the case of EyeExposé timing was another issue. We observed that users looked away at the side monitor in anticipation of instructions for the next target before they released the trigger key.

5.6.3 Subjective Results

Subjects' perception that EyeExposé was faster than other approaches (in contrast to empirical data) is an interesting result. We hypothesize that users may interpret the lower cognitive load of a technique like EyeExposé as speed in their mind. Survey results also indicate that user preference for EyeExposé was strong. While some of this could be attributed to demand effects, in our discussion with subjects they expressed that EyeExposé "feels like the right way," was "natural," and "simple to use."

5.7 Summary

We found that using a combination of keyboard or other trigger to activate the Exposé-like visualization of open applications and then using eye gaze for selection was an effective technique for switching between applications quickly and naturally. Our studies showed that users strongly preferred EyeExposé as the application switching technique of choice.

6 Password Entry

Text passwords remain the dominant means of authentication in today's systems because of their simplicity, legacy deployment and ease of revocation. Unfortunately, common approaches to entering passwords by way of keyboard, mouse, touch screen or any traditional input device, are frequently vulnerable to attacks such as shoulder surfing (i.e. an attacker directly observes the user during password entry), keyboard acoustics [14, 22, 120], and screen electromagnetic emanations [55].

Current approaches to reducing shoulder surfing typically also reduce the usability of the systems; often requiring users to use security tokens [93], interact with systems that do not provide direct feedback [92, 113] or they require additional steps to prevent an observer from easily disambiguating the input to determine the password/PIN [6, 41, 92, 103, 111, 113]. Previous gaze-based authentication methods [47, 48, 69] do not support traditional password schemes.

We present EyePassword, an alternative approach to password entry that retains the ease of use of traditional passwords, while mitigating shoulder-surfing and acoustics attacks. EyePassword utilizes gaze-based typing, a technique originally developed for disabled users as an alternative to normal keyboard and mouse input. Gaze-based password entry makes gleaning password information difficult for the unaided observer while retaining simplicity and ease of use for the user. As expected, a number of design choices affect the security and usability of our system. We discuss

Portions of this chapter were originally published by the author, Tal Garfinkel, Dan Boneh and Terry Winograd in [59].

these in Section 6.4 along with the choices we made in the design of EyePassword. We implemented EyePassword using the Tobii 1750 [107] eye tracker and conducted user studies to evaluate the speed, accuracy and user acceptance. Our results demonstrate that gaze-based password entry requires marginal additional time over using a keyboard, error rates are similar to those of using a keyboard and users indicated that they would prefer to use the gaze-based approach when entering their password in a public place.

6.1 Background and Related Work

Shoulder-surfing is an attack on password authentication that has traditionally been hard to defeat. It can be done remotely using binoculars and cameras, using keyboard acoustics [120], or electromagnetic emanations from displays [55]. Access to the user's password simply by observing the user while he or she is entering a password undermines all the effort put in to encrypting passwords and protocols for authenticating the user securely. To some extent, the human actions when inputting the password are the weakest link in the chain.

Biometric methods, which identify individuals based on physiological or behavioral characteristics, have the advantage that they are harder to replicate and therefore are not susceptible to the risks of shoulder surfing. However, biometric techniques suffer from the drawback that biometric characteristics are non-secret and non-revocable. While it is easy for a user to change a password, it is a considerably less convenient and presumably more painful procedure for the user to change a fingerprint or retinal scan.

Physical token based approaches such as the RSA SecurID token [93] overcome shoulder-surfing, but such devices require users to carry a physical access token, which is prone to being lost or stolen.

In general, approaches to overcoming shoulder surfing rely on "increasing the noise" for the observer so that it becomes difficult for the observer to disambiguate the user's actions/input. Roth et al. [92] present an approach for PIN entry which uses the philosophy of increasing the noise for the observer. In their approach, the PIN digits are displayed in two distinct sets colored black and white. For each digit

the user must make a series of binary choices as to which set (black or white) the PIN digit appears in. The correct PIN digit is identified by intersecting the user's set choices. The approach requires users to make multiple binary selections in order to correctly input each digit of the PIN.

Wiedenbeck et al. [113] introduce a shoulder-surfing-resistant graphical password scheme. The user selects a number of icons as his or her pass icons. When logging in, the user is presented with a random assortment of icons. The user must find the pass icons previously identified, create a mental image of the convex hull formed by these icons and then click inside this convex hull. The scheme again relies on multiple challenge response passes in order to successfully authenticate the user. This approach requires the user to learn a new approach and also increases the length of the authentication process.

PassFaces [6] relies on the user recognizing faces and pointing to recognized faces as responses to a series of challenges. Hoanca et al. [48] extend PassFaces using eye gaze for selecting the face from within the grid. Weinshall [111] introduces an approach that uses a set of machine generated pictures as the user's password. The user must memorize the pictures. When presented with the login screen, the user must mentally trace a path which includes the password pictures and answer a multiple choice question. A series of challenge-response sets result in authentication. Since only the user knows which path was traced, a human or software observer (spy-ware) would be unable to determine the correct password. However, as the author states, "the benefit is obtained at the cost of a relatively long login time of a few minutes." The approach has been shown to be insecure against an eavesdropping adversary in [41].

Tan et al. [103] propose a spy-resistant keyboard, which uses a level of indirection to prevent the observer from guessing the password. Their approach adds sufficient ambiguity for the observer to be unable to determine the user's choice without remembering the layout of the entire keyboard. However, to enter the password, users must use an unfamiliar keyboard layout and complex interaction technique.

While there are other approaches to prevent shoulder surfing [47], it is sufficient to note that all the approaches have the common theme of increasing the noise/ambiguity for the observer. Usually this is achieved by increasing the number of interactions the user must do to successfully log in.

Maeder et al. [69] present a gaze-based user authentication scheme in which a user is presented with an image and must dwell upon previously specified points of interest on the image in a predetermined order in order to log in. The authors do not present an analysis of the ease with which a malicious user may guess the order of the points of interest on the image. In addition, this scheme doesn't support the use of traditional passwords.

Other approaches to overcoming shoulder-surfing include the use of tactile passwords [99] or more invasive techniques such as brain computer interfaces [104].

6.2 Motivation for Eye Tracking

Devices such as Apple's MacBook laptops include a built-in iSight camera [2] and hardware trends indicate that even higher resolution cameras will be embedded in standard display devices in the future. Using such a camera for eye tracking would only require the addition of inexpensive IR illumination and image processing software.

Figure 43. On screen keyboard layout for ATM PIN entry.

ATMs are equipped with security cameras and the user stands directly in front of the machine. Since ATM pins typically use only numbers, which need fewer distinct regions on the screen, the quality of the eye tracking required for tracking gaze on an ATM keypad does not need to be as high as the current state-of-the-art eye trackers. Current generation eye trackers

require a one-time calibration for each user. We envision a system where the calibration for each user can be stored on the system. Inserting the ATM card identifies the user and the stored calibration can be automatically loaded.

Gaze-based password entry has the advantage of retaining the simplicity of using a traditional password scheme. Users do not need to learn a new way of entering their password as commonly required in the techniques described in the previous section. At the same time, gaze-based password entry makes detecting the user's password by shoulder surfing a considerably harder task, thereby increasing the security of the password at the weakest link in the chain – the point of entry. Gaze-based password entry can therefore provide a pragmatic approach achieving a balance between usability and security.

6.3 Threat Model

We model a shoulder surfer as an adversary who observes the user's keyboard and screen. Moreover, the adversary can listen to any sound emanating from the system. Our goal is to build an easy to use password-entry system secure against such adversaries. We assume the adversary can observe the user's head motion, but cannot directly look into the user's pupils. A shoulder surfer looking at the user's eyes during password entry will surely arouse suspicion. We note that a video recording of both the computer screen and the user's eyes during password entry could in theory defeat our system. The purpose of our system is to propose a pragmatic interaction which eliminates the vast majority of the shoulder-surfing attacks. It would indeed be difficult for a shoulder surfer to record both the screen activity and a high resolution image of the user's eyes and be able to cross-reference the two streams to determine the user's password.

6.4 Design Choices

The basic procedure for gaze-based password entry is similar to normal password entry, except that in place of typing a key or touching the screen, the user looks at each desired character or trigger region in sequence (same as eye typing). The approach can therefore be used both with character-based passwords by using

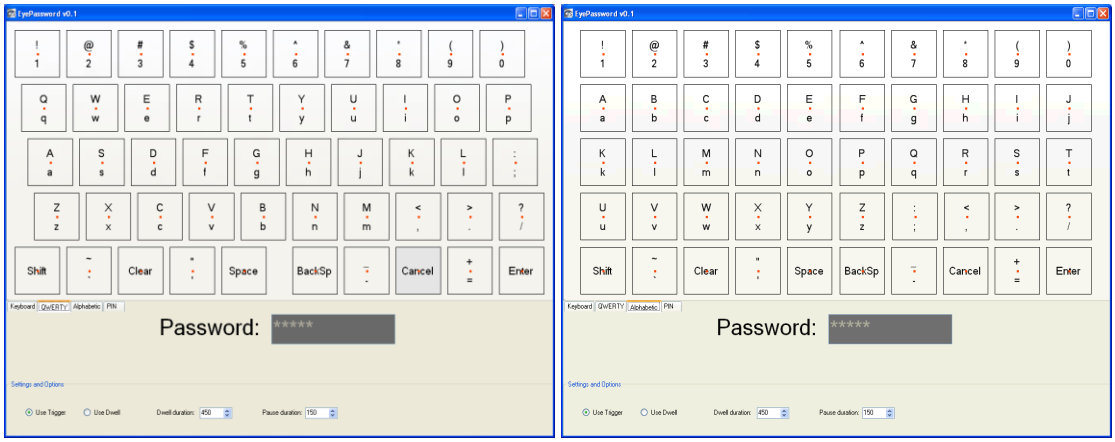


Figure 44. On-screen keyboard layout for gaze-based password entry showing QWERTY, Alphabetic layouts.

an on-screen keyboard and with graphical password schemes as surveyed in [102]. A variety of considerations are important for ensuring usability and security.

6.4.1 Target Size

The size of the targets on the on-screen keyboard should be chosen to minimize false activations. The key factor in determining the size of the targets is not the resolution of the display, but the accuracy of the eye tracker. Since the accuracy is defined in terms of degrees of visual angle, the target size is determined by calculating the spread of the angle measured in pixels on the screen at a normal viewing distance.

The vertical and horizontal spread of the 1 degree of visual angle on the screen (1280x1024 pixels at 96 dpi) at a normal viewing distance of 50 cm is 33 pixels. This implies that when looking at a single pixel sized point, the output from the eye-tracker can have an uncertainty radius of 33 pixels, or a spread of 66 pixels. The size of the targets should be sufficiently greater than 66 pixels to prevent false activations. We chose a target size of 84 pixels with a 12 pixel inter-target spacing to minimize the chances of false activations when using gaze-based selection.

While it is certainly possible to use gaze-based password entry with eye movements alone and no corresponding head movements, we observed that subjects may move their head when looking at different parts of the screen. Though the head movements are subtle they have the potential to reveal information about what the

user may have been looking at. For example, the attacker may deduce that the user is looking at the upper right quadrant. Clearly, the smaller and more tightly spaced the keys in the on-screen keyboard, the less information the attacker obtains from these weak observations. This suggests a general design principle: the on-screen keyboard should display the smallest possible keys that support low input error rates.

6.4.2 Keyboard Layout

Since muscle memory from typing does not translate to on-screen keyboard layouts, the user's visual memory for the spatial location of the keys becomes a more dominant factor in the design of on-screen keyboards. The trade-off here is between usability and security — it is possible to design random keyboard layouts that change after every login attempt. These would require considerably more visual search by the user when entering the passwords and therefore be a detriment to the user experience, but would provide increased security. For this reason, we chose not to use randomized layouts in our implementation.

6.4.3 Trigger Mechanism

There are two methods for activating character selection. In the first method, dwell-based [70], the users fix their gaze for a moment. The second method is multi-modal — the user looks at a character and then presses a dedicated trigger key. Using a dedicated trigger key has the potential to reveal timing information between consecutive character selections, which can enable an adversary to mount a dictionary attack on the user's password [101]. The dwell-based method hides this timing information. Furthermore, our user studies show that dwell-based methods have lower error rates than the multi-modal methods.

6.4.4 Feedback

Contrary to gaze-based typing techniques [71], gaze-based password entry techniques should not provide any identifying visual feedback to the user (i.e. the key the user looked at should not be highlighted). However, it is still necessary to provide the user with appropriate feedback that a key press has indeed been registered. This

can be done by sounding an audio beep or flashing the background of the screen to signal the activation. Additional visual feedback may be incorporated in the form of a password field that shows one additional asterisk for each character of the password as it is registered. To reduce the amount of timing information leaked by the feedback mechanism, the system can output a feedback event only in multiples of 100 ms. In either case, the feedback will leak information regarding the *length* of the password.

6.4.5 Shifted Characters

Limits on screen space may prevent all valid password characters (e.g., both lower and upper case) from being displayed in an on-screen layout. Our implementation shows both the standard character and the shifted character in the same target. To type a shifted character, the user activates the shift key once, which causes the following character to be shifted. This approach reveals no additional information to the observer. An alternative approach would be to show only the standard character on-screen and change the display to show the shifted characters once the user activates the shift mode. However, this approach would leak additional information to the observer about the user's password.

6.5 Implementation

We implemented EyePassword on Windows using a Tobii 1750 eye tracker [107] set to a resolution of 1280x1024 pixels at 96 dpi. Figures 1 shows the EyePassword on-screen keyboards using a QWERTY, alphabetic and ATM pin keypad layout respectively. As discussed earlier, to reduce false activations we chose the size of each target to be 84 pixels square. Furthermore, the keys are separated by a 12 pixel margin which further decreases the instances of false activations. We also show a bright red dot at the center of each of the on-screen buttons. These "focus points" (Figure 45) help users to focus their gaze at a point in the center of the target thereby improving the accuracy of the tracking data [61].

It should be noted that our on-screen layout does not conform exactly to a standard keyboard layout. A standard QWERTY layout has a maximum of 14 keys in

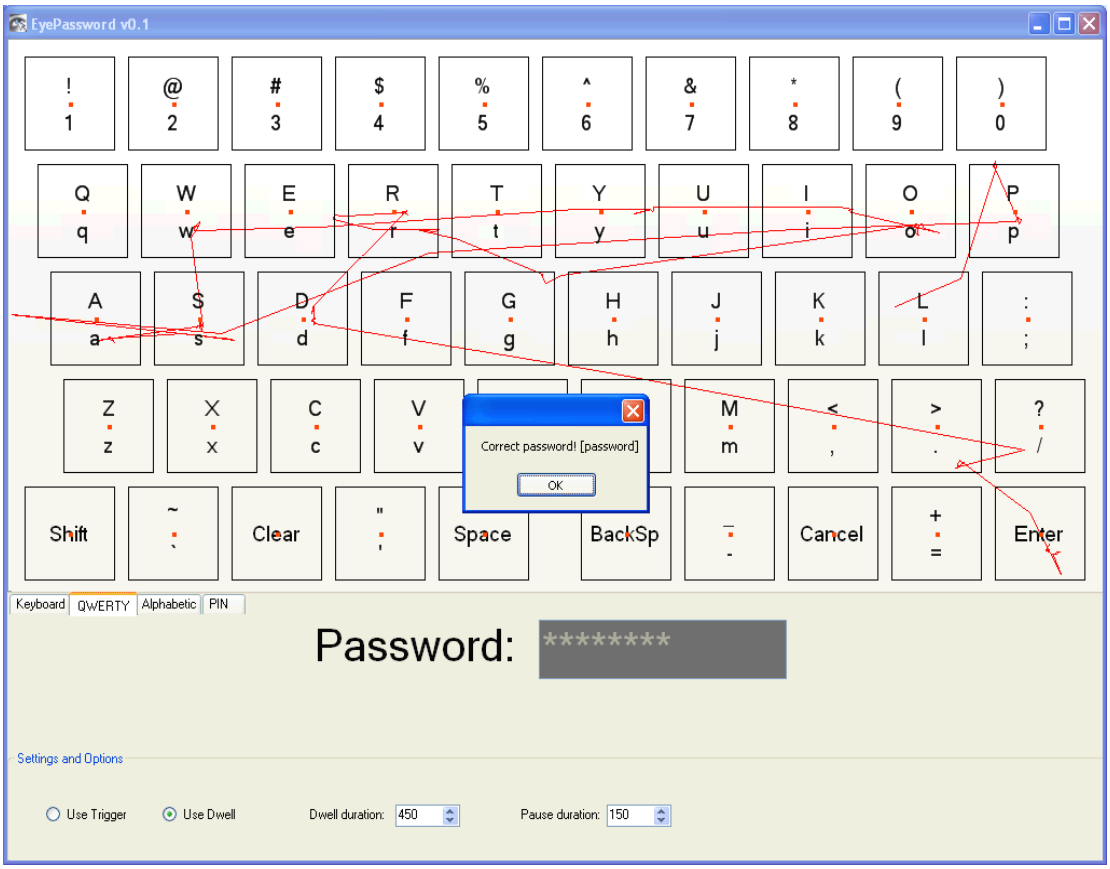


Figure 45. Gaze-pattern when the user enters "password" as the password. Each key has a bright red dot at the center of it. This focus point allows the user to focus their gaze at the center of the target thereby increasing the accuracy of eye tracking data.

a row. At a width of 84 pixels it would be possible to fit all 14 keys and maintain a QWERTY layout if we used all of the horizontal screen real-estate on the eye-tracker (1280x1024 resolution). We chose to implement a more compact layout which occupies less screen real-estate, keeping the regular layout for the alphabetical and number keys

Previous research [70-72] has shown that the ideal duration for activation by dwell is on the order of 400-500 ms. Consequently, we chose 450 ms for our implementation, with an inter-dwell pause of 150 ms. An audio beep provides users with feedback when a dwell-based activation is registered.

Our implementation shows both the standard characters and the shifted characters on-screen and provides no visual feedback for the activation of the shift key.

Gaze data from the eye tracker is noisy due to errors in tracking and also due to the physiology of the eye. We therefore implemented a saccade[2] detection and fixation smoothing algorithm [56] (discussed in Section 9.1) to provide more reliable data for detecting fixations.

6.6 Evaluation

To evaluate EyePassword, we conducted user studies with 18 subjects, 9 males and 9 females with an average age of 21. Thirteen subjects did not require any vision correction; 5 subjects used contact lenses[3]. Twelve subjects reported that they were touch-typists. Subjects had an average of 12 years of experience using a keyboard and mouse.

We compared the password entry speed and error rates of three approaches: a standard keyboard for entering a password: (Keyboard) to provide a baseline; using EyePassword with dwell-based activation (Gaze+Dwell); and using EyePassword with trigger-based activation (Gaze+Trigger). In addition, we evaluated two different on-screen layouts for the dwell case: QWERTY layout and alphabetic layout.

6.6.1 Method

We implemented a test harness to capture timing and error data for users entering passwords in a controlled environment. To minimize memory effects, the users were shown the password in a dialog box immediately before they were asked to enter it. Each subject was first trained on the four test conditions: Keyboard, Gaze+Trigger (QWERTY layout), Gaze+Dwell (QWERTY layout) and Gaze+Dwell (Alphabetic layout). Subjects were trained on using each of the techniques on a practice set of four passwords which exercised the use of letters, numbers, upper-case and lower-case characters and symbols. Once subjects were comfortable with

[2] A saccade is a ballistic movement of the eye used to reposition the visual focus to a new location in the visual environment.

[3] The eye tracker does work with eye-glasses provided the glasses do not occlude/impair the camera's view of the eye. We have had subjects with eye-glasses in previous studies.

each approach, they repeated the trials with the real password data set of ten passwords shown below. Passwords were chosen to be representative of common passwords with a length of 8-9 characters and included a combination of lowercase, uppercase, numbers and symbols.

Training set: password, number1, capitalA, $symbol

Real set: computer, security, apple314, sillycat, Garfield, password, $dollar$, GoogleMap, dinnertime, Chinatown.

The order of the techniques was varied for each subject in order to counterbalance across subjects and to minimize learning effects. We measured the amount of time it took the user to enter each password. If the password was entered incorrectly, this was recorded as an error and the trial was repeated. Upon completion of the study, subjects were asked fill out a questionnaire to provide their subjective opinions on the techniques used.

6.6.2 Results

Figure 46 shows the average time to enter the password in each of the four conditions. Figure 47 shows the percentage error in each condition.

A repeated measures analysis of variance (ANOVA) of the password entry time shows that the results are significant ($F(1.44,24.54)=117.8$, $p<.01$, Greenhouse-Geisser corrected). Contrast analyses between the four techniques showed that the differences between the keyboard and all the gaze-based techniques are significant. While the average typing time for the trigger-based approach was higher than the dwell-based approach, this result was not significant — some users were faster using dwell, others using the trigger. The differences between the QWERTY layout and the alphabetic layout were significant indicating that users found the QWERTY layout faster.

The error rates on Gaze+Dwell (QWERTY) and Gaze+Dwell (Alpha) were similar to those on a keyboard. The trigger-based approach had a significantly higher error rate.

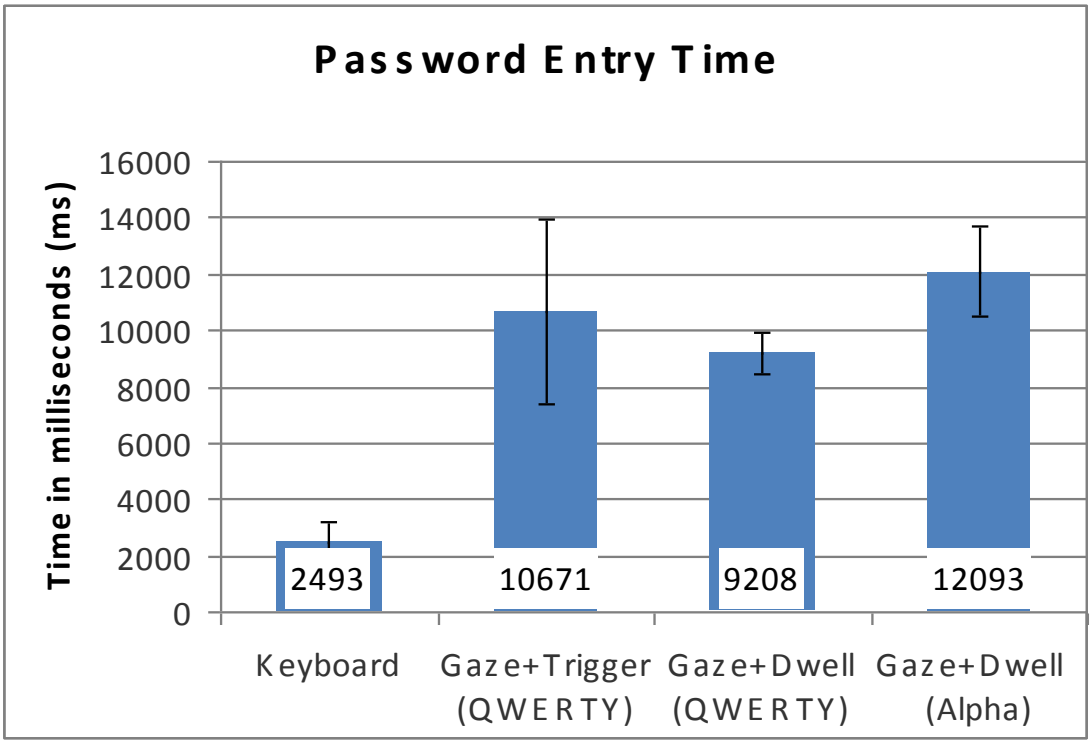


Figure 46. Average time for password entry across all users in each of the 4 conditions. Differences between Gaze+Dwell and Gaze+Trigger are not significant. Differences between QWERTY and alpha layouts are significant.

Our survey results showed that subjects unanimously preferred using the QWERTY layout over the alphabetic layout. Subjects did not indicate that the time to enter the password using the gaze-based approaches was a concern. The subjective results for the trigger mechanism (dwell-based or trigger-based) were counter to the results from our objective evaluation – a majority (63%) of subjects felt that the trigger approach was faster and more accurate than using dwell. Subjects overwhelmingly (83%) indicated that they would prefer to use a gaze-based approach over using a traditional keyboard when entering their password in a public place.

6.7 Discussion

While the speed difference between using dwell or trigger was not significant, the results do show that the error rates with the trigger approach are significantly higher (15% compared to 3-4%). We speculate this is because it is difficult for

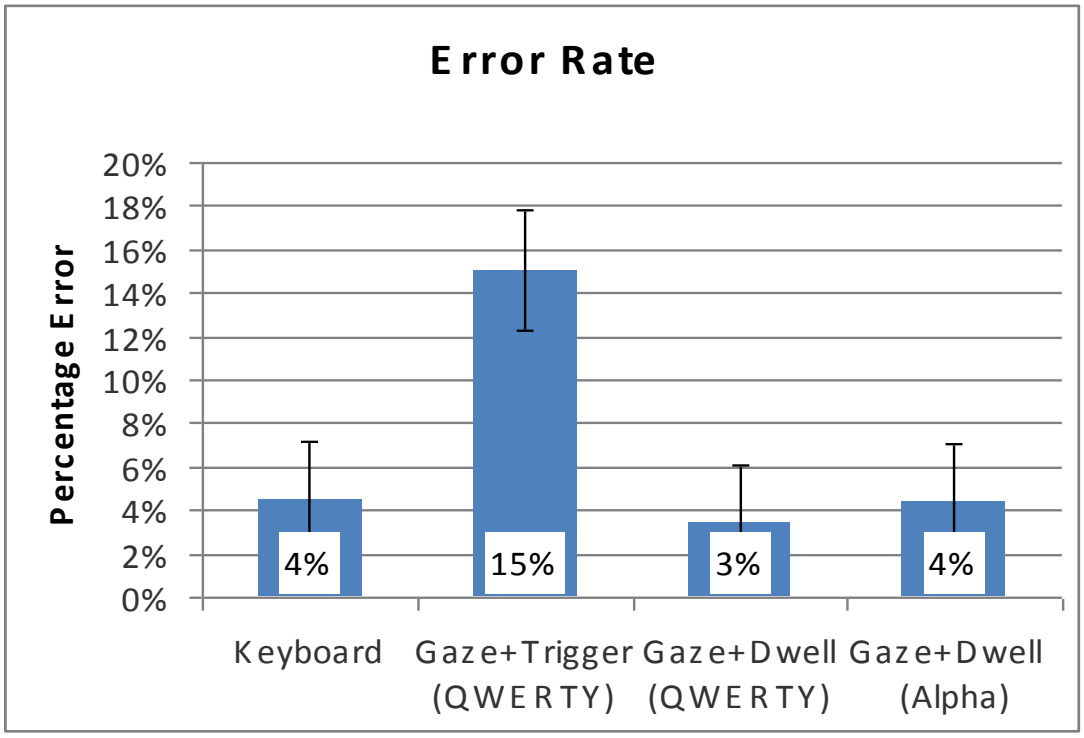


Figure 47. Percentage error in password entry across all users in each of the four conditions. Error rates in the Gaze+Dwell conditions were similar to those of the keyboard. Gaze+trigger error rates were considerably higher presumably due to eye-hand coordination.

humans to time their eye gaze and hand to coordinate perfectly (This was studied further and is reported in Section 9.2). Most errors in the trigger condition occurred because either the subjects had not yet focused on the target or had already moved their eyes off the target by the time they pressed the trigger. While we suspect that this behavior can probably be corrected for algorithmically (see Section 9.2), under the current implementation the dwell based implementation is more robust.

The results also showed that the QWERTY layout outperformed the alphabetic keyboard layout. This indicates that the visual search time for finding characters on a QWERTY layout is lower than the visual search time for an alphabetic layout, possibly due to the fact that people have extensive training on the QWERTY layout. This is consistent with the findings in Norman [82].

The study for entering passwords using a keyboard did not account for the increase in speed seen as a result of subjects developing muscle memory over time

by entering their password repeatedly. We expect that similar to the muscle memory for typing passwords, learning effects for visual search on the on-screen layout will speed up password entry over time as subjects develop muscle memory in their eyes to enter their password.

When compared to password-entry time with the keyboard, the gaze-based approaches are about five times slower. However, it should be noted that even at an average of a 10 second entry time, the gaze-based password entry is several times faster than alternative techniques to prevent shoulder surfing [47, 48, 92, 103, 111, 113].

6.8 Future Work

We can strengthen a password by extracting a few additional bits of entropy from the gaze path that the user follows while entering the password. In theory, the user will follow a similar path, with similar dwell times, every time. A different user, however, may use completely different dwell times. As a result, stealing the user's password is insufficient for logging in and the attacker must also mimic the user's spatiotemporal gaze path. A similar technique was previously used successfully to enhance the entropy of passwords entered on a keyboard [75].

While our results showed that the trigger-based mechanism had considerably higher error rates due to eye-hand coordination, it is conceivable that this can be accounted for algorithmically by examining the historical gaze pattern and correlating it with trigger presses.

6.9 Summary

Passwords possess many useful properties as well as widespread legacy deployment. Consequently we can expect their use for the foreseeable future. Unfortunately, today's standard methods for password input are subject to a variety of attacks based on observation, from casual eavesdropping (shoulder surfing), to more exotic methods. We have presented an alternative approach to password entry, based on gaze, which deters or prevents a wide range of these attacks. User studies have demonstrated that this approach requires additional entry time, has accuracy

similar to traditional keyboard input, and provides an experience preferred by a majority of users.

7 Zooming

Zooming user interfaces have been a popular topic of research [18, 19, 79]. Zooming interfaces have the potential to provide an overview of the data or information being visualized while at the same time to provide additional detail upon demand by the user. The characteristic interaction of zooming interfaces requires the user to pick the region of interest that should be zoomed in to. Typically this is provided by the mouse or some form of pointing device. In this chapter we investigate the possibility of using eye gaze to provide the contextual information for zooming interfaces.

7.1 Background and Related Work

Fono and Vertegaal [40] use both the concept of zooming and gaze-based interaction in their work on EyeWindows. They propose the use of elastic windows which grow and shrink depending on the focus of the user's eye gaze. In this work, eye-gaze is used as a selection mechanism for switching between windows.

Bedersen introduced the concept of Zooming User Interfaces (ZUIs) with his pioneering work on Pad++[18] and the Zooming Web Browser [19]. Both of these interfaces and other examples of ZUIs all rely on the use to define their region of interest by clicking and performing some kind of a mouse action.

Mapping applications such as Google Maps [3] are perhaps today the most widely used application which makes extensive use of zooming. Mapping services by Google and others all use semantic zooming to display additional detail once the user zooms in to a region of interest. In the case of Google Maps, zooming can be accomplished in a couple of different ways. The most common approach is to click

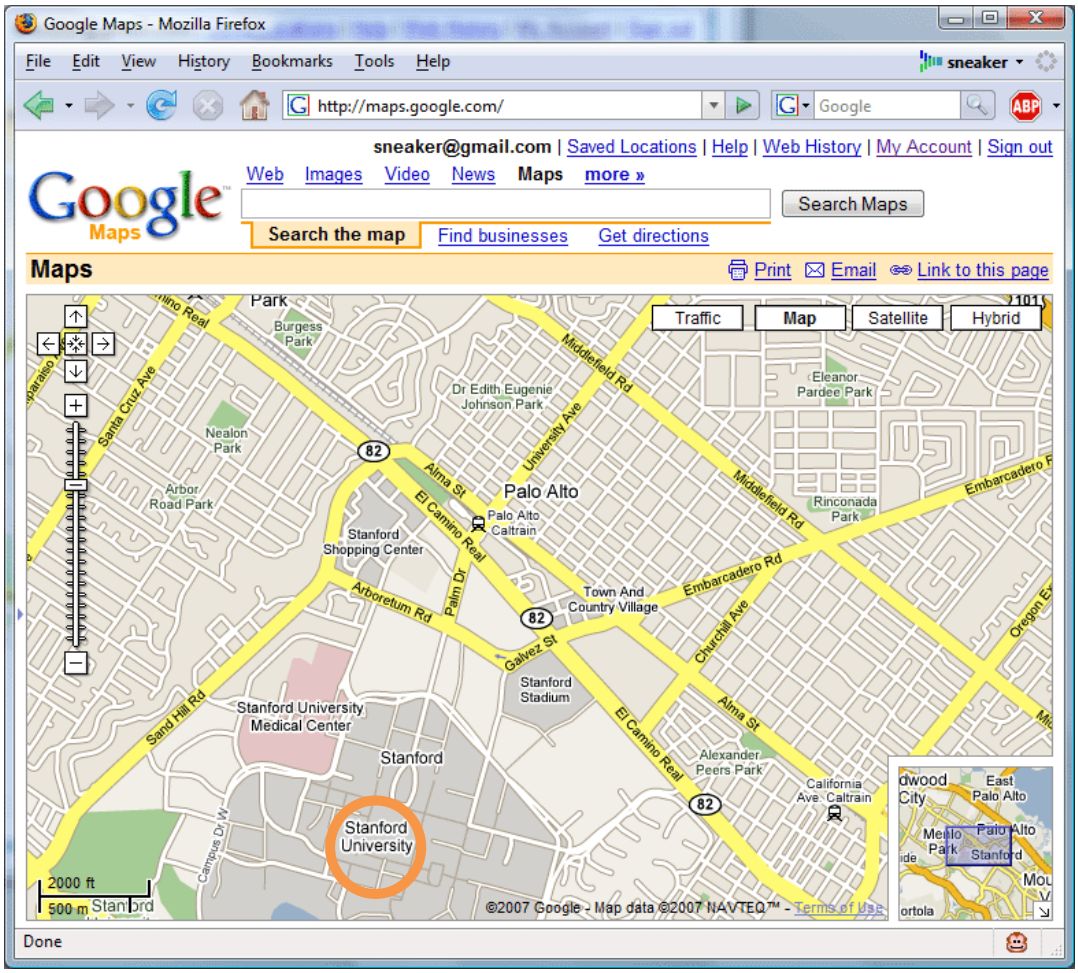


Figure 48. Google Map image before the user clicks on the + button to zoom in one level. The region of interest (annotated by the orange circle) happens to be the Stanford oval.

zooming. The object of gaze-contingent semantic zooming is to allow the user to specify his or her region of interest, simply by looking at it and then activating the zoom action. The zoom action may be activated by using any approach such as pressing a key on the keyboard or using mouse buttons.

7.3 Prototype Implementations

We implemented several prototypes for gaze contingent semantic zooming as described below and conducted pilot studies to test their efficacy.

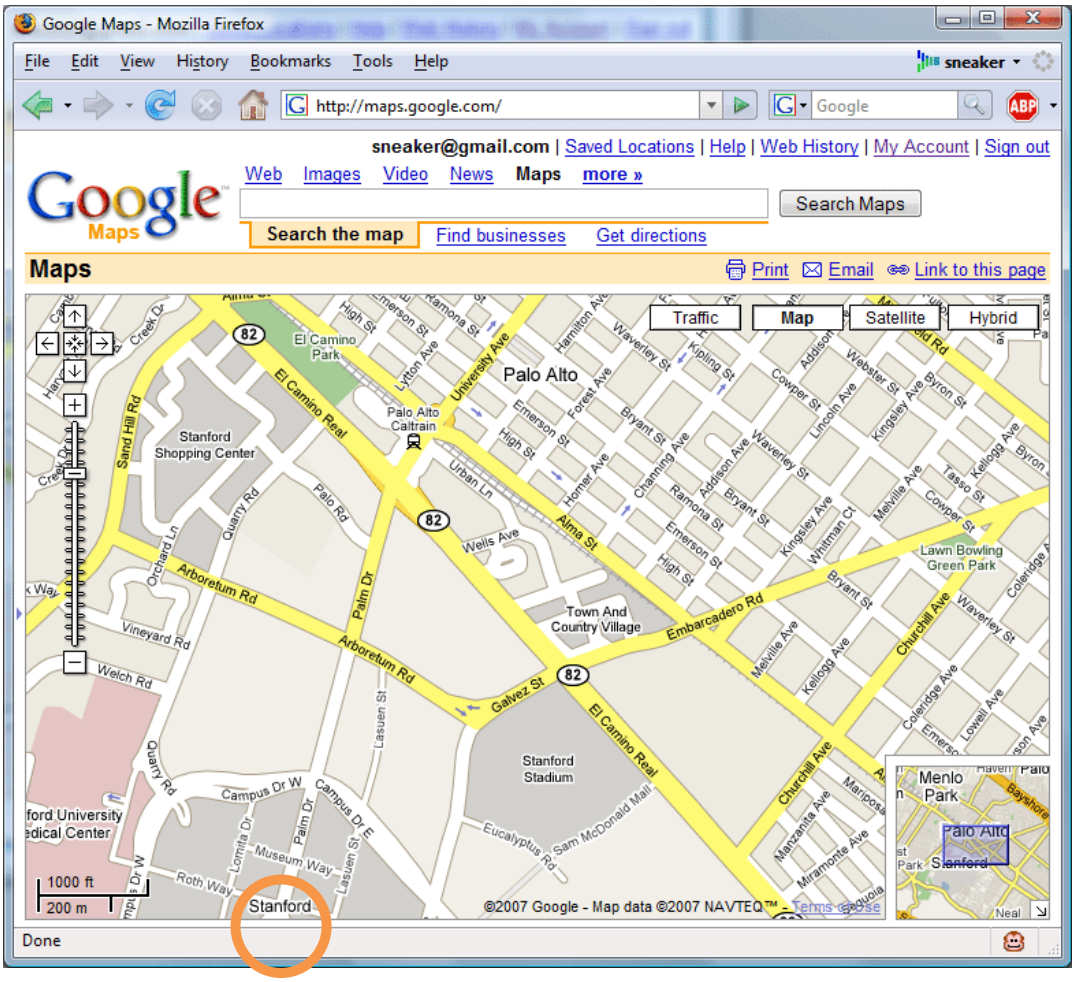


Figure 49. Google Map image immediately after the user clicked the + button to zoom in. Notice that the region of interest (annotated by the orange circle) is already outside the visible region of the map.

7.3.1 Google Maps Prototype

We implemented a prototype which automatically moved an on-screen cursor to the location where the user was looking. The scroll wheel on the mouse was used to initiate zooming. In this prototype, since the mouse location moved to follow the user's eye gaze, we expected that the zooming would then happen based on the user's gaze position, thereby implementing the gaze-contingent zooming described above.

Pilot studies with this prototype revealed that this approach is problematic because the gaze-location returned by the eye tracker is not very accurate.

Therefore, if the user was looking at point *P*, chances are that the eye tracker may think that the user is looking at the point *P+ε*, where ε is the error introduced by the eye tracker. Once the user initiates a zoom action, the map is magnified. Therefore, if the zoom factor is *z,* then the resulting error gets magnified to zε, which can be considerably larger than the original error. In addition, Google Maps uses discrete, non-continuous zooming, which made it difficult to use make small-grained corrections as the eye adjusts to the new location of the region of interest after each zoom step.

It should be noted here that this analysis presents a generalizable problem with using gaze as a source of context for semantic zooming. In particular, zooming based on gaze does not work well, since the error in eye tracking gets magnified with each successive zoom level. This negative result for gaze-contingent semantic zooming is in line with this dissertation's research on EyePoint for pointing and selection (described in Chapter 3). EyePoint introduced a magnified view of the region the user was looking at, thereby increasing the visible size of the target on the screen. The secondary gaze position when the user looked at the target in the magnified view helped to refine the target by a factor equal to the magnification, i.e. we were now closer to the target by the amount of the magnification. In the case of gaze-contingent semantic zooming, the error in tracking gets magnified and there is no simple way to reduce this error, other than by introducing additional steps into the interaction.

The Google Maps prototype illustrated a fundamental problem for gaze-contingent semantic zooming. We considered several other approaches to try to overcome this limitation.

7.3.2 Windows Prototype

One of the issues we encountered with the Google Maps prototype was the discrete nature of the zooming. We felt that a more continuous zooming action, might provide for the possibility of progressive refinement, i.e. the user's gaze-position is sampled multiple times during the zooming which may make it possible for the gaze to adjust and adapt to the error being introduced by zooming.

To overcome the zooming granularity and speed issues, we implemented a Windows application written in C#. However, the speed at which the interface would repaint to do multiple zoom levels made the prototype unusable.

7.3.3 Piccolo Prototype

We therefore implemented a second prototype that used the Piccolo Toolkit [17] for zooming user interfaces. Pilot studies with this prototype showed that while we could now control the granularity of the zooming sufficiently to make small corrections, the speed of the zooming with large canvases was still too slow for the prototype to be usable for further analysis.

7.4 Discussion

While the prototypes here do not demonstrate a clear performance win for gaze-contingent semantic zooming on applications such as Google Maps, there are several ZUIs [18, 19, 51] that use discrete, well-defined targets that are fairly large in size. It is very plausible to use a gaze-plus-trigger activation for these applications. For instance in the case of the Zooming Web Browser [19], the user can simply look at the link he or she wishes to zoom into and press a key. Similarly in the case of TimeQuilt [51] or Denim[79], selecting which collection to zoom into can easily be specified using gaze. However, we do not consider these applications of gaze to be examples of gaze-contingent semantic zooming since they can equally well be classified as examples of gaze-based pointing – like the approaches we presented in Chapter 2 and Chapter 3.

Although these explorations are preliminary, our prototyping did reveal a fundamental problem with using gaze as part of zooming interfaces. Zooming interfaces tend to magnify the error in the accuracy of the eye-tracker and therefore using gaze to provide content for ZUIs does not seem to be a promising approach.

8 Other Applications

Previous chapters presented an in-depth discussion of applications that use gaze as a form of input. Using contextual information gained from the user's gaze enables the design of novel applications and interaction techniques, which can yield useful improvements for everyday computing. We implemented several such applications: a gaze-contingent screen and power saver, a gaze-enhanced utility for coordination across multi-monitor screens, a gaze-controlled virtual screen and a prototype for showing a deictic reference in a remote collaboration environment. These applications were implemented on the Tobii 1750 eye tracker. While formal usability analyses of these applications we not performed, pilot studies and personal use have shown that these applications have utility for users. We also present the concept of the no-nag IM windows and the focus plus context mouse.

8.1 Gaze-contingent screen and power saver

The eye tracker provides gaze-validity data for each eye. When the eye tracker does not find any eyes in the frame, it returns a validity code indicating that no eyes were found. It is therefore trivial to determine if and when a user is looking at the screen.

We implemented *EyeSaver* as a simple application which can activate the screen saver when the user has not been looking at the screen for a specified period of time. This approach is more effective at determining when to activate the screen saver than traditional approaches which rely on periods of keyboard and mouse inactivity. Setting a short delay (10-15 seconds) for activating the screen saver when relying on keyboard and mouse inactivity can yield numerous false positives, since

the user may be reading something on the screen for that duration of time without having typed or moved the mouse. Therefore, screen saver activation delays are typically set to be in minutes rather than seconds when using traditional time out based methods. With a gaze-based approach, the system can reliably determine whether or not the user is looking at the screen before activating the screen saver with a very short delay.

In addition, since the system can also detect when the user begins looking at the screen again, it can automatically deactivate the screen saver as well.

It should be noted that the same approach that is used to activate and deactivate the screen saver can also be used to conserve power by turning off the screen when the user is not looking at it and turning it back on when the user looks at it. This approach may be especially useful for mobile computers which run on battery. This concept has been explored in depth by Dalton et al. in [32].

8.2 Gaze-enhanced Multi-Monitor Coordination

An increasing number of computer users and especially computer professionals now use multiple displays. It is not uncommon to see two or even sometimes three displays on a user's desktop. However, while the increasing screen real estate can lead to productivity gains [31], it also increases the distance that needs to be traversed by the mouse. Multiple monitors have the potential to increase the time required for pointing since users may need to move the mouse across multiple screens. In addition, users often complain that they context switch between different monitors and sometimes will begin typing when they look at the other monitor, but before they have actively switched their application focus to the right window.

We propose a solution to these problems using a gaze-enhanced approach to multi-monitor coordination. In essence, since the system now can be aware of which screen the user is looking at, it can automatically change the focus of the active application depending on where the user is looking. Similarly, the mouse can also be warped in the vicinity of the user's gaze. Benko [21] proposed a Multi-Monitor Mouse solution which uses explicit button based activation to warp the mouse

between the screens in a multi-monitor setup. Our solution extends this approach by leveraging the fact that we can detect which screen the user is looking at. This effectively applies the same concept as in Zhai's MAGIC pointing [118] to a multi-monitor setup where the benefit of having the augmented pointing technique would be greater than that on a single monitor.

The mudibo system proposed by Hutchings [50] overcomes the problem of determining dialog placement on multiple monitor setups by replicating the dialog on all screens. By contrast, a gaze-enhanced multi-monitor setup could position dialogs depending on where the user is looking. In fact, it can also use attention-based notification to place urgent dialogs directly in the user's gaze and place non-urgent dialogs in the periphery of the user's vision.

8.3 Gaze-controlled virtual screens/desktops

As noted in Section 4.3, the eye tracker provides sufficient accuracy and field of view to distinguish when the user is look off the screen at the bezel of the monitor. Using this approach we implemented off-screen gaze-actuated buttons for document navigation. Figure 31D shows how the eye tracker can be instrumented for 8-way panning. We extended this prototype to create a *gaze-controlled virtual screen* — where the available screen real-estate is more than the viewable region of the screen. When the user's gaze falls upon one of the gaze-activated hotspots for the duration of a micro-dwell, the system automatically pans the screen in the appropriate direction. Our prototype was implemented by using VNC to connect to a computer with a higher resolution than the resolution of the eye tracker screen. Informal studies and personal use of this prototype suggests that this technique can be effective when the user only has a small display portal available, but needs to use more screen real-estate.

The gaze-activated hotspots on the bezel of the screen can also be used to summon different virtual desktops into view. In this scenario, each time the user looks off screen at the bezel for the duration of a micro-dwell (150-200 ms) and then back again, the display on the screen is changed to show the content of the virtual desktop that would be in the same spatial direction as the users gaze gesture. This

approach has the potential to allow for an infinite number of virtual desktops; the practical limits would defined by the cognitive load of keeping track of the content and the location of these desktops.

8.4 Deictic Reference in Remote Collaboration

Remote collaboration tools such as WebEx, Live Meeting, and Netspoke provide users with the ability to share their desktop or specific applications with a larger number of viewers on the web. However, when displaying an application or document remotely, it is common for the presenter to be looking at a region of interest on the screen while talking. Unfortunately, this deictic reference is lost in most remote collaboration tools, unless the presenter remembers to actively keep moving the mouse to point to what he or she is looking at. This problem can be addressed easily by tracking the presenter's gaze and highlighting the general area that the presenter is looking at for the viewers of the remote collaboration session.

Duchowski [37] uses gaze as a deictic reference in a virtual environment and has also done work on using gaze for training novices in an aircraft inspection task [94]. Qvarfordt [88] also discusses the use of gaze as a deictic reference for controlling the flow of conversation in a collaborative setting. The suggested approach extends their work to apply it to remote collaboration environments, such as web conferencing, to transfer the visual cues about what the user is looking at in a co-located environment to a distributed collaboration environment.

8.5 No-Nag IM Windows

Instant messaging is being increasingly used by computer users at home and at work. It is not uncommon to be busy working on something and to be interrupted by an instant message window. Even if the user attempts to ignore the window and continue working until a reasonable stopping point, most IM windows will continue to flash in order to gain the users attention. The current solution is to interrupt the task at hand in order to click on the IM window to acknowledge the alert.

Gaze could be leveraged to create a *No-Nag IM window* which can be context aware: as soon as the user has looked at the window once, it stops flashing for some

period of time (it may resume flashing at a later point to remind the user in case the user has not attended to the message for a while). This concept has been suggested by other researchers as well as an example of an attentive user interface.

8.6 Focus Plus Context Mouse

We consider here the case of those applications which require very fine-grained mouse movements, such as image editing or drawing. These applications require the user to perform fine-grained motor control tasks in order to gain the necessary precision with the mouse. We propose a gaze-enhanced version of the mouse cursor, where the control-to-display ratio of the mouse is modified to reduce the acceleration and mouse movement within the user's current gaze point, thereby allowing for more fine-grained control within the current gaze region. This approach allows the user to still move the mouse rapidly across the screen, but slows down the movement of the mouse once it gets within range of target, which is typically where the user is looking.

This approach is similar in theme to the Snap-and-Go work by Baudisch [16] where the user is able to snap to grid by adjusting the control-display ratio of the mouse when close to traditional snapping regions. Our approach can also be considered to be an extension to Zhai's MAGIC pointing [118] where the mouse is allowed to warp or move rapidly in all parts of the screen, except with it is within the user's gaze point, to allow for finer control on the movement of the mouse. Further research would be needed to evaluate if such a technique is useful.

8.7 Summary

The applications described in this chapter present several examples of how gaze can be used to inform applications of the user's attention and intention and help to design novel interactions. The use of gaze as a sensory input expands the potential for such attentive user interfaces. Application designers can design interfaces that blend seamlessly with the user's task flow by developing an interruption model of the user that leverages gaze information, making it possible to design interactions that are less intrusive and decrease cognitive load.

9 Improving Gaze Input

In implementing EyePoint [61], we found that while the speed of a the gaze-based pointing technique was comparable to the mouse, the error rates were significantly higher. Our work on gaze-based password entry showed that dwell-based activation had significantly lower error rates than trigger based activation. These results encouraged us to conduct a series of studies to better understand the source of these errors and to identify ways to improve the accuracy of gaze-plus-trigger activation.

This chapter presents three methods we developed for improving the accuracy and user experience of gaze-based pointing: an algorithm for real-time saccade detection and fixation smoothing, an algorithm for improving eye-hand coordination and the use of focus points. These methods boost the basic performance for using gaze information in interactive applications. In our applications they made the difference between prohibitively high error rates and practical usefulness of gaze-based interaction.

9.1 Saccade Detection and Fixation Smoothing

As discussed in Section 2.5.1, basic eye movements can be broken down into two types: *fixations* and *saccades*. A fixation occurs when the gaze rests steadily on a single point. A saccade is a fast movement of the eye between two fixations. However, even fixations are not stable and the eye jitters during fixations due to drift, tremor and involuntary micro-saccades [115]. This gaze jitter, together with

Portions of this chapter were originally published by the author, Jeff Klingner, Rohan Puranik, Terry Winograd and Andreas Paepcke in [59] (submitted to UIST 2007).

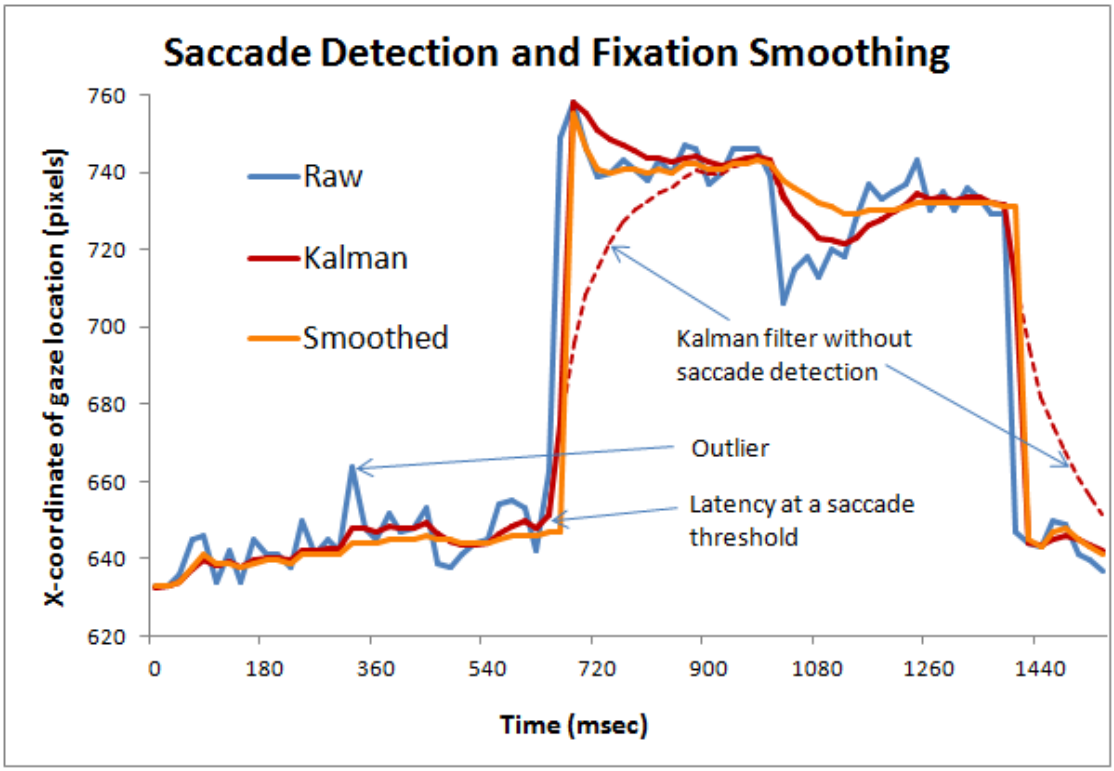


Figure 50. Results of our real-time saccade detection and smoothing algorithm. Note that the one measurement look-ahead prevents outliers in the raw gaze data from being mistaken for saccades, but introduces a 20ms latency at a saccade thresholds.

the limited accuracy of eye trackers, results in a noisy gaze signal.

The prior work on algorithms for identifying fixations and saccades [76, 95, 97, 115] has dealt mainly with post-processing previously captured gaze information. For using gaze information as a form of input, it is necessary to analyze eye-movement data in real time.

To smooth the data from the eye tracker in real-time, it is necessary to determine whether the most recent data point is the beginning of a saccade, a continuation of the current fixation or an outlier relative to the current fixation. We use a gaze movement threshold, in which two gaze points separated by a Euclidean distance of more than a given saccade threshold are labeled as a saccade. We chose a saccade threshold of 40 pixels (slightly more than 1° of visual angle) for our implementation. This is similar to the velocity threshold technique described in [97], with two modifications to make it more robust to noise. First, we measure the displacement of each eye movement relative to the current estimate of the fixation

```
point FilterData(point)
  if current fixation is empty add the point to current fixation and return
  if current fixation has points and potential fixation has points
  calculate the distance of the point from each fixation
  if the new point is closer to the current fixation
    if the distance is less than the saccade threshold
      add the point to the current fixation (max 20 points)
      clear the potential fixation
      return the new current fixation
    else
      clear the potential fixation
      add the point to the potential fixation
      return the current fixation
  else
    if the distance is less than the saccade threshold
      add the point to the potential fixation (max 20 points)
      make potential fixation the new current fixation
      clear the potential fixation
      return the current fixation
    else
      save the potential fixation
      clear the potential fixation
      add the point to the potential fixation
return the saved potential fixation
  else
    if the point is beyond the saccade threshold from the current fixation
      add the point to the potential fixation
      return the current fixation
    else
      add the point to the current fixation
      return the current fixation
End FilterData
```

Figure 51. Pseudocode listing for Saccade Detection and Fixation Smoothing algorithm.

location rather than to the previous measurement. Second, we look ahead one measurement and reject movements over the saccade threshold that immediately

return to the current fixation. This prevents single outliers of the current fixation from being mislabeled as saccades. It should be noted that this look-ahead introduces a one-measurement latency (20ms for the Tobii 1750 eye tracker [107]) at saccade thresholds (Figure 50).

The algorithm (Figure 51) maintains two sets of points: the current fixation window and a potential fixation window. If a point is close to the current fixation (within a saccade threshold), then it is added to the current fixation window. The new current fixation is calculated by a weighted mean which favors more recent points (described below). If the point differs from the current fixation by more than a saccade threshold, then it is added to the potential fixation window and the current fixation is returned. When the next data point is available, if it was closer to the current fixation, then we add it to the current fixation and throw away the potential fixation as an outlier. If the data point is closer to the potential fixation, then we add the point to the potential fixation window and make this the new current fixation.

The fixation point is calculated as a weighted mean (a one-sided triangular filter) of the set of points in the fixation window. The weight assigned to each point is based on its position in the window. For a window with n points (P_0, P_1… P_{n-1}) the mean fixation would be calculated by the formula:

$$P_{fixation} = \frac{1P_0 + 2P_1 + ... + nP_{n-1}}{(1 + 2 + ... + n)}$$

The size of the fixation window (n) is capped to include only data points that occurred within a dwell duration [70, 71] of 400-500ms (20 data points for the Tobii eye tracker). We do this to allow the fixation point to adjust more rapidly to slight drift in the gaze data.

Figure 50 shows the output from the smoothing algorithm for the x-coordinate of the eye-tracking data. We also show a Kalman Filter [112] applied to the entire raw gaze data and a Kalman Filter applied in parts to the fixations only. A Kalman Filter applied over the entirety of the raw gaze data smoothes over saccade intervals. The nature of eye movements, in particular the existence of saccades, necessitates that the smoothing function only be applied to fixations, i.e. within saccade boundaries. Applying the Kalman filter in parts to the fixations yields

comparable results to our one-sided triangular filter discussed above. It is possible that applying a non-linear variant of the Kalman filter [12], or a better process model of eye movements for the Kalman filter may yield better smoothing results. The advantage of our approach is that the algorithm is very simple and most of all it is tailored to account for the different forms of eye movements and also tolerate the noise in eye tracking data.

While there is still room for improvement in the algorithm above by taking into account the directionality of the incoming data points, we found that our saccade detection and smoothing algorithm improved the reliability of the results for applications which rely on the real-time use of eye-tracking data. Simulation results are presented in the .following section.

9.2 Eye-hand Coordination

Our research on using a combination of gaze and keyboard for performing a pointing task [61] showed that error rates were very high. Additional work on using gaze-based password entry [58] showed that the high error rates existed only when the subjects used a combination of gaze plus a keyboard trigger. Using a dwell-based trigger exhibited minimal errors. These observations led us to hypothesize that the errors may be caused by a failure of synchronization between gaze and triggers.

To determine the cause and the number of errors we conducted two user studies with 15 subjects (11 male, 4 female, average age 26 years). In the first study, subjects were presented with a red balloon. Each time they looked at the balloon and pressed the trigger key, the red balloon popped and moved to a new location (Moving Target Study). In the second study, subjects were presented with 20 numbered balloons on the screen and asked to look at each balloon in order and press the trigger key (Stationary Targets Study). Subjects repeated each study twice, once optimizing for speed and trying to perform the task as quickly as possible and the second time optimizing for accuracy and trying to perform the study as accurately as possible. Study order was counter-balanced and trials were repeated in case of an error.

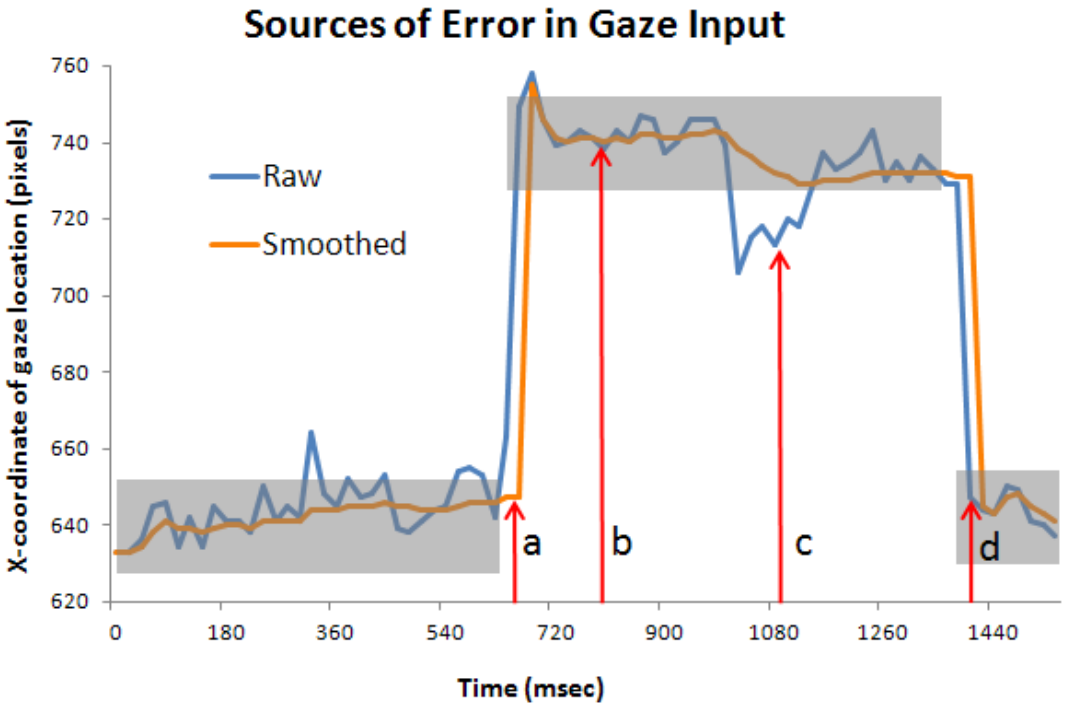


Figure 52. Sources of error in gaze input. Shaded areas show the target region. Example triggers are indicated by red arrows. The triggers shown are all different attempts to click on the upper target region. The trigger points correspond to: a) early trigger error, b) raw hit and smooth hit, c) raw miss and smooth hit, and d) late trigger error.

We performed an in-depth analysis of the data from the two studies by manually plotting a graph showing the current target location, the gaze-position as reported by the eye tracker and the trigger for each error. Errors were coded by two independent coders and classified based on the following error types:

Tracking errors: caused due to the eye tracker accuracy. These include cases in which the gaze data from the eye tracker is biased or when the location of the target closer to the periphery of the screen results in lower accuracy from the eye tracker [20].

Early-Trigger errors: caused because the trigger happened before the user's gaze was in the target area. Early triggers can happen because a) the eye tracker introduces a sensor lag of about 33ms in processing the user's eye gaze, b) the smoothing algorithm introduces an additional latency of 20ms at saccade thresholds c) in some cases (as in the Moving Target study) the users may have only looked at

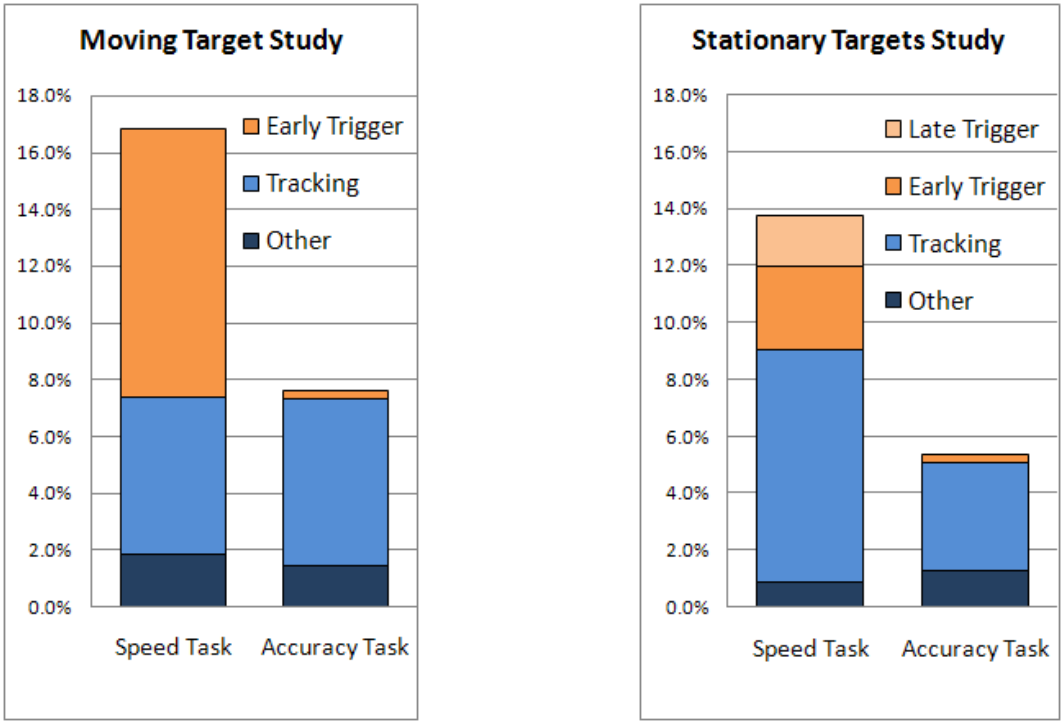


Figure 53. Analysis of errors in the two studies show that a large number of errors in the Speed Task happen due to early triggers and late triggers – errors in synchronization between the gaze and trigger events.

the target in their peripheral vision and pressed the trigger before they actually focused on the target.

Late-Trigger errors: caused because users had already moved their gaze on to the next target before they pressed the trigger. Late triggers can happen only in cases when multiple targets are visible on the screen, as in the Stationary Targets study or in gaze-based typing.

Other errors: these include a) smoothing errors caused when the smoothed data happened to be outside the target boundary, but the raw data point would have, by chance, resulted in a hit, b) human errors where the subject just was not looking at the right thing or the subject looked down at the keyboard before pressing the trigger.

Figure 52 illustrates the different error types. Figure 53 shows how often each type of error occurred in the two studies.

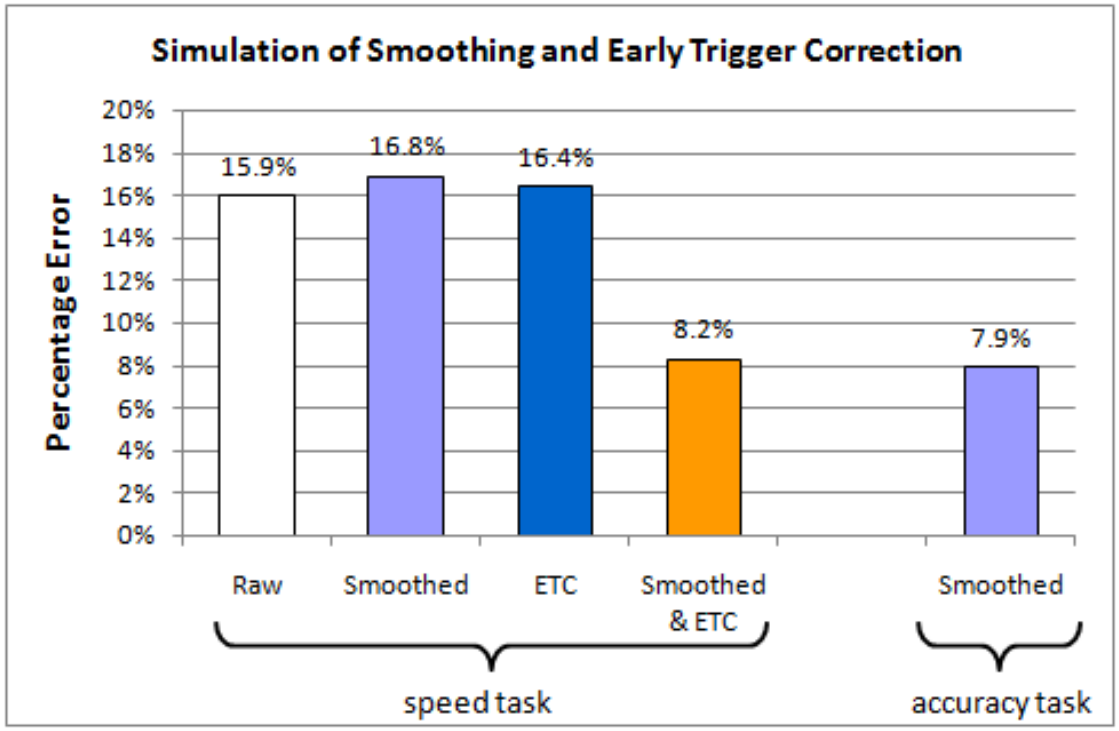


Figure 54. Simulation of smoothing and early trigger correction (ETC) on the speed task for the Moving Target Study shows that the percentage error of the speed task decreases significantly and is comparable to the error rate of the accuracy task.

To improve the accuracy of gaze-based pointing in the case of the speed task, we implemented an Early Trigger correction (ETC) algorithm which delays trigger points by 80ms to account for the systematic bias due to sensor lag, smoothing latency and peripheral vision effects. We simulated this algorithm over the data from the Moving Target study. Figure 54 shows the outcome from the simulated results. It should be noted that applying smoothing and early-trigger correction alone actually increased the error rate, because smoothing introduces a latency that the early trigger would be correcting. The error rate in the speed task when using a combination of smoothing and early trigger correction approaches the error rate of the accuracy task — without compromising the speed of the task.

While our ETC algorithm used a fixed delay, it is conceivable to run a one-time calibration program which measures, which uses known target locations and trigger-based activation to measure the empirical temporal offset for triggers by correlating the gaze position with the known target location at the time of the

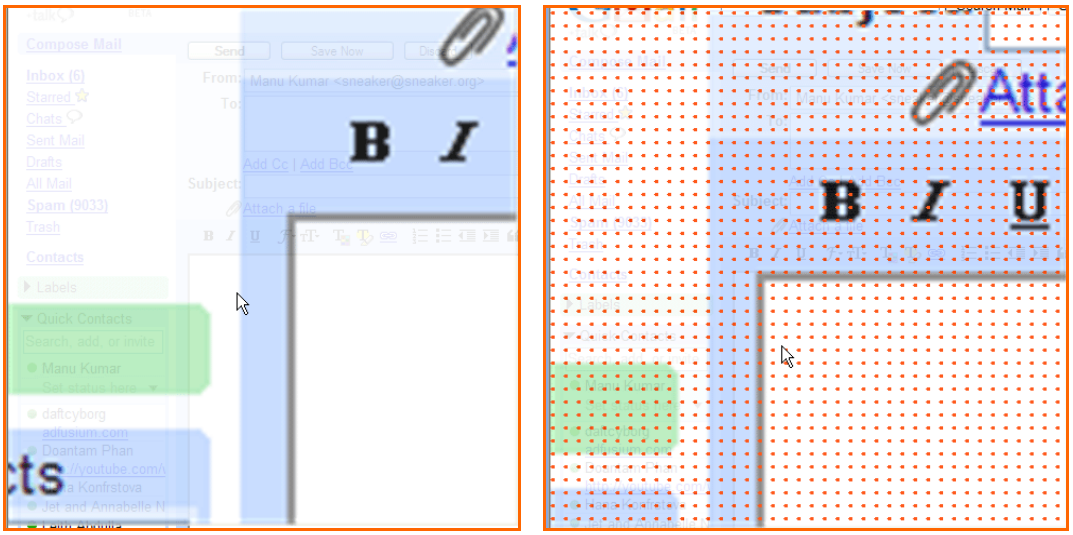


Figure 55. Magnified view for gaze-based pointing technique with and without focus points. Using focus points provides a visual anchor for subjects to focus their gaze on, making it easier for them to click in the text box.

trigger. The ETC algorithm can potentially increase errors in pathological situations if the trigger is delayed beyond the current fixation, however, analysis of the results from the simulations show that its benefits far exceed the costs. Our algorithm represents preliminary work for resolving the problem of eye-hand coordination and this remains an open area for additional research.

While we were able to identify late-trigger errors in the analysis of the data, it is difficult to distinguish a late trigger from an early trigger or even an on-time trigger without using semantic information about the location of the targets. Since our approach has focused on providing generally applicable techniques for gaze-input, which do not rely on application or operating system specific information, we did not attempt to correct for late triggers. We note that the use of semantic information about target locations has the potential to significantly improve the accuracy of gaze-based input by allowing the current fixation to be applied to the closest target.

9.3 Focus Points

In Section 3.2, we introduced the use of *Focus Points*—a grid pattern of dots overlaid on the magnified view that contained the targets (see Figure 55). We

hypothesized that focus points assist the user in making a more fine-grained selection by focusing the user's gaze, thereby improving the accuracy of the eye tracking. However, the studies presented in that chapter showed no conclusive effect of an improvement in tracking accuracy when using focus points.

To test this hypothesis further, we conducted a user study with 17 subjects (11 male, 6 female, average age 22. In the first part of the study, subjects were shown a red balloon and asked to look at the center of the balloon. Once they had looked at the balloon for a dwell duration (450ms) the balloon automatically moved to a new location. In the second part, subjects repeated the study, but with the center point of the balloon clearly marked with a focus point. The order was varied and each subject was shown 40 balloons. The user's raw and smoothed gaze positions were logged for each balloon. At the end of the study users were presented with a 7-point Likert scale questionnaire which asked them which condition was easier and whether they found the focus point at the center of the balloon useful.

We computed the standard deviation of the Euclidean distance of each gaze point from the center point of the target. The results from the study show that within the bounds of the measurable accuracy of the eye tracker (33 pixels in any direction, diameter of spread 66 pixels), the use of focus points did not have a significant impact in concentrating the user's gaze on the center of the target. The questionnaire results however, indicate that subjects found the condition with the focus point easier and found the focus point to be useful when trying to look at the center of the target. These results are consistent with our findings in Section 3.4.

We conclude that while the use of focus points may not measurably improve the accuracy of the raw gaze data from the eye tracker, they do indeed make pointing easier and provide a better user experience. This is illustrated by Figure 55, which shows two views of the magnified view from EyePoint. If the subject intends to click in the text area in the bottom right of the magnified view, the task is easier for the subject in the condition with focus points, since the focus points provide a visual anchor for the subject to focus upon.

9.4 Summary

In this Chapter we revisited and deepened the exploration of some common underlying issues for using gaze as a form of input. The techniques presented above improve the use of gaze input by addressing challenges in how the system interprets gaze data from eye trackers (saccade detection and smoothing), how to match gaze input with an external trigger (eye-hand coordination) and by introducing features that make it easier for the user to look at the desired target (focus points). The above techniques can be applied at an application layer to improve the use or gaze as a form of input and are orthogonal to any improvement in the underlying tracking technology that provides for more accuracy and range of head movement from the eye tracker.

10 Low-cost Eye Tracking

Previous chapters have demonstrated several gaze-enhanced interaction techniques and ways of improving the systems interpretation of gaze data. However, one fundamental issue that needs to be addressed before gaze-based interfaces can be widely used is cost. Eye trackers range in cost from US$5,000-US$40,000. As mentioned in Section 2.4, the eye tracker used for our research costs US$30,000. The high price tag of these systems has resulted in a limited adoption and use of the technology.

The high cost of commercial systems forces has led to numerous efforts to build home-brew eye-tracking systems, as seen in [15, 42, 43, 45, 68, 83, 84]. However, building an eye tracker and researching applications of eye gaze are two very different tasks, which require different skill sets. The former requires in-depth knowledge of Computer Vision, while the latter focuses on the design and evaluation on gaze-based interactions.

This chapter examines the factors that contribute to the high costs of eye-tracking systems and proposes several ideas and strategies that can be used to reduce the costs of these systems, ultimately resulting in more widespread use of the technology. This material complements our presentation of gaze-enhanced user interfaces in the preceding chapters by discussing the prospects for the underlying technology to be made affordable. The information presented in this chapter represents the author's opinions and is not presented as an academic contribution of this dissertation.

10.1 Market Background

Current markets for eye-tracking technologies include: disabled users, usability analysis laboratories in the private sector and universities, and other specialized research uses in the fields of psychology, marketing, defense, and medicine. Unfortunately, while there have been significant advances in eye-tracking technology, the cost of commercial systems remains prohibitive for broad use. Even in the case of disabled users, the number of people who are able to afford such a system is a small fraction of those who could benefit from the technology. Eye-tracking is often not used simply because of the cost factor.

Eye-tracking vendors complain that the lack of "a killer application" has kept the demand for the technology low and therefore, they have to charge the high prices in order to recover their research and development cost and remain in business. Users of eye-tracking complain that the high cost of eye-tracking systems limits the research on the use of eye-gaze in applications and interfaces. Eye tracking is caught in a vicious cycle of high cost and low demand.

10.2 Technology Background

We provide a brief background on eye-tracking, without going into technical details which can be found in several papers including [45, 68, 77, 78, 83]. Remote eye-tracking technology is a specialized application of computer vision. A camera is used to track the location of the center of the pupil (p) with reference to the corneal reflection (g) of one or more infrared glint sources (Figure 56). Since the surface cornea is nearly spherical, the position of the glint remains more or less fixed as the eye moves to focus on different points-of-regard. Eye-tracking systems use the difference vector (p-g) between the pupil position and the corneal reflection to determine the gaze vector. As is evident from Figure 56, it is critical to have sufficient resolution on the subject's eye to be able to get a reasonable estimate of the gaze vector. This necessitates the need for high resolution imaging sensors in eye tracking systems.

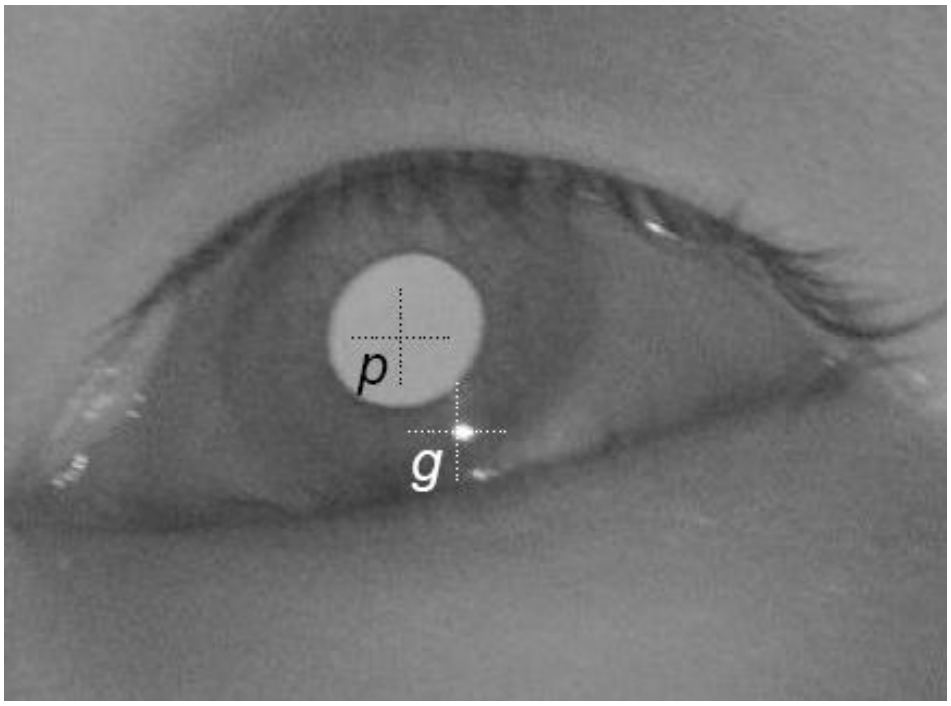


Figure 56. An image of the eye showing the center of the pupil (*p*) and the corneal reflection (*g*). The difference vector (*p-g*) is used to determine the gaze vector. Source: *Theory for Calibration-Free Eye Gaze Tracking* by Amir et al.

Eye tracking may use either a bright-pupil approach — where a set of on-axis illuminants cause a red-eye effect in the subjects eyes making the pupil glow, or a dark-pupil approach — where a set of off-axis illuminants cause the pupil to show as a dark circle. Computer vision techniques are used to locate the center of the pupil and the corneal reflection (s). Research conducted in the BlueEyes project [4], introduced the use of active illumination which uses a difference of the dark pupil and the bright pupil to locate the pupil center. However, as noted by Nguyen [80], there are differences in the bright pupil response of humans.

Once the system has determined the pupil and glint coordinates, a mathematical transform [10, 36] is applied to compute the gaze position on screen. The software for eye tracking must also accommodate the inherent noise in eye movements as discussed in Chapter 2. Systems also need to be robust enough to accommodate head movement and compensate between differences in head

movement and gaze direction. The use of infra-red light sources impacts the ability of the system to operate under direct sunlight or other sources of high infrared light.

10.3 Cost Factors

We classify the costs associated with building a commercial eye-tracker into a) Material costs, b) Research and development costs and c) Business costs. The latter two comprise the dominant factors in the cost of current commercially available eye-tracking systems.

10.3.1 Material Costs

The hardware components of an eye tracker include one or more high resolution, high frame-rate, infrared capable camera(s), the camera lens, IR illumination circuitry and IR illuminants (LEDs), and mechanical parts for housing and creating a fixed frame of reference. Since eye-tracking relies on tracking the corneal reflection which is very small relative to the size of the face, the camera resolution needs to be sufficiently high to get enough pixels on the eye region. It is possible to trade off resolution for field of view by using a zoom lens that focuses on the eye. However, this would severely limit free head movement or require active steering. Current commercial systems rely on using cameras which have a 1-2 megapixel resolution with a 50-60 Hz. frame rate. These cameras are estimated to be in the price range of US$1,000-US$4,000.

10.3.2 Research and Development Costs

Reliable gaze-tracking requires the hardware and the software to work perfectly in concert. The hardware required for the IR illumination varies depending upon the approach used (dark pupil vs. light pupil). The timing of the IR illumination, the camera optics (field of view and zoom), and the geometry of the system (camera position, glint source position, screen position) all play a critical role in determining the final accuracy of the system. The hardware development is explained in several papers [15, 77, 83].

Developing and fine-tuning the software for reliable gaze tracking, including calibration routines, APIs and software for analyzing gaze patterns can take several person-years of software development effort. Most commercial systems rely on custom developed image processing and provide proprietary SDKs and APIs for developing applications using their eye-tracking systems. The investment in research and development dominates in the ultimate cost to users.

10.3.3 Business Costs

Given the current niche markets and low demand for eye trackers, vendors must invest considerable time and resources on marketing and sales. The specialized nature of current eye-tracking systems makes them suitable for use only by experts. Vendors must therefore charge high prices on small volumes (typically tens of units on an annual basis) to see a return on their investment. The high price of the systems in turn requires a high-touch sales process, which requires vendors to have an expensive sales force that needs to travel and do live demonstrations to close sales.

In addition, current systems are not robust enough to operate under all conditions, creating the need for hands-on customer support. The combination of technological issues (hardware and software development) and market/business issues result in eye-tracking continuing as a boutique industry.

10.4 Technology Trends

Higher resolution and higher frame rate cameras are becoming available at lower prices. The advent of cell phone cameras and low cost web cameras has made image sensors a commodity item which can be cheaply sourced. Furthermore, the proliferation of the USB 2.0 standard now provides adequate bus bandwidth to capture high resolution images at high frame rates. Moore's law has made adequate processing power available to perform complex image processing in real time and still leave enough cycles for other applications. The cost of image processing therefore becomes a smaller proportion of the CPU over time.

10.5 Cost-Lowering Approaches

We present a series of ideas and strategies that may be used to lower the cost of eye-tracking systems.

10.5.1 Use of Mass-market Image Sensors

Using commercial-over-the-shelf (COTS) cameras is the obvious cost-cutting approach for reducing the material costs for eye-trackers. Megapixel resolution web cameras are now available for a fraction of the cost of the expensive, custom cameras used in machine vision applications. These cameras use standard USB or FireWire interfaces, thereby eliminating the need for any special hardware and software for image acquisition.

Consumer web cameras come equipped with an IR filter, which prevents them from working in the IR spectrum. In addition, since COTS cameras are mostly color cameras, the presence of the Bayer pattern on the CCD also reduces the effective resolution of the camera when working in the infrared spectrum. However, it is possible to perform minor modifications on consumer webcams to make them work in the IR spectrum and we illustrate how to do so in [57]. Given the low cost of consumer web cameras, it is conceivable to have high-resolution, high frame-rate, grayscale, IR sensitive image sensor mass-produced at very low cost.

10.5.2 Use of Multiple cameras

Desktop eye-tracking systems suffer from the limited field of view of the camera. As explained earlier, the image of the eye-region must be sufficiently zoomed-in, in order to provide adequate pixels for processing. In most desktop use scenarios, the majority of head motion occurs in the horizontal plane. Therefore, it is possible to use a multi-camera (stereo) setup with a fixed geometry to increase the horizontal field of view without sacrificing resolution. In addition, stereo cameras can provide a more accurate depth estimate and account for a wider range of head movement including head rotation. It is our expectation that current systems would be limited to using a stereo setup with two cameras due to bus bandwidth and processing limitations.

Figure 57. The low-cost prototype in development uses commercial-over-the-shelf cameras modified to work in the infrared spectrum. The glint source pictured above uses IR LEDs (invisible to the human eye).

10.5.3 Build on Existing Image Processing Libraries

To control the cost of software development, it is possible to build eye-tracking software on top of existing computer vision libraries such as OpenCV [25], which provide packaged functions for image processing and machine learning. The openEyes [68] project uses OpenCV as its foundation.

10.6 Low-Cost Prototype

To test the feasibility of the ideas above we built our own low cost prototype. We used the Logitech QuickCam Pro 4000 camera and modified the camera to work in the infrared spectrum as described in [57]. Figure 1 shows an image of the modified web cameras and the infrared glint source used for prototyping. We were

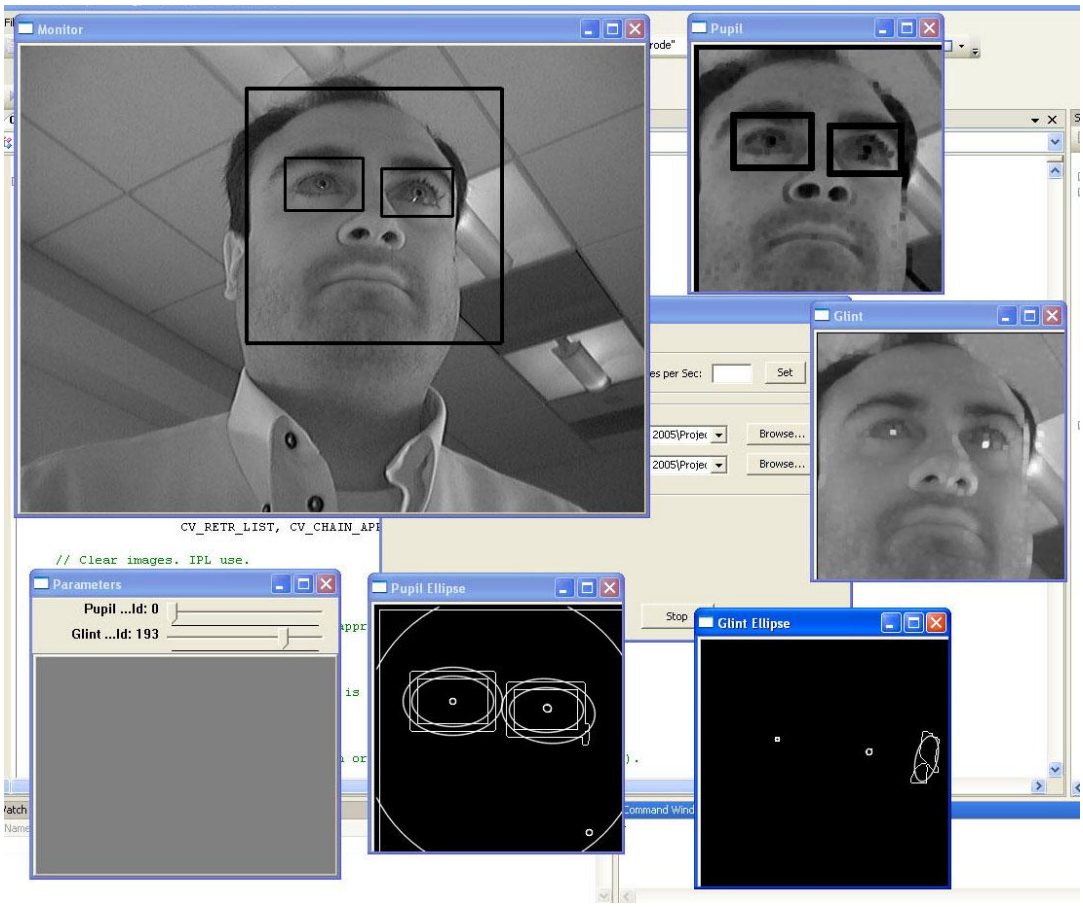


Figure 58. A screenshot of prototype software built using open source Computer Vision libraries (OpenCV) which uses machine learning to identify faces in the image. It then looks within the face region to identify the eyes. Simple image processing (erosion/dilation) helps to separate the pupil and glint images. Ellipse-fitting provides the center of the pupil and the glint which can then be used to determine the point-of-regard.

successfully able to capture both streams and verify the feasibility of a multi camera solution using a regular desktop PC.

We prototyped software using the open source OpenCV library. The software uses the HaarFaceDetector in OpenCV to identify faces in the captured image. To find eye-regions, we trained a classifier set using over 3000 sample eye images. Once the HaarFaceDetector finds a face within the captured image, the eye-models are used to isolate eye-regions. Simple erosion, dilation followed by ellipse fitting makes it possible to determine the location of the pupil center (p) and the corneal reflection (g) which can then be used to estimate the point of regard.

We were successfully able to identify the pupil centers and the location of the glints using our prototype (Figure 58). It should be noted however, that the resolution of the current consumer web cameras is insufficient for any reasonable amount of tracking. The number of pixels between the pupil center and the glint do not provide enough information for reliable eye tracking. Other solutions that have used web cameras have placed the web camera close to the user's eye, as in [43]. The availability of the low-cost high resolution image sensor is an essential requirement.

10.7 Mass Market Strategy

The primary impediment to low-cost eye-tracking is not the technology but the business issues relating to the supply and demand of eye-trackers. It is necessary to innovate both the technology and the business models used by current eye-tracking vendors.

Vendors must begin to use commodity parts in order to reduce the material costs of the systems. The product must be designed to be usable by everyday users and not only by experts. This means reducing the learning curve for using such systems and making them robust enough to reduce the burden of customer support. Research and development costs must be amortized over a longer period of time and over a larger number of units.

One novel approach as proposed by Amir et al.[11] takes part of the eye-tracking system, specifically, eye-detection and embeds it in hardware making it possible to have a simple USB peripheral (Figure 59). The device does all image processing in hardware and therefore does not consume any significant CPU resources. The output from the system is the low bandwidth eye position data, which can be transmitted over USB. Innovative approaches, such as the one above, can be instrumental in making the technology robust, simple, inexpensive and ultimately more widely used.

While it is important for vendors to charge high prices to recoup their R&D cost, a mass market strategy has the potential to grow the size of the market for eye tracking.

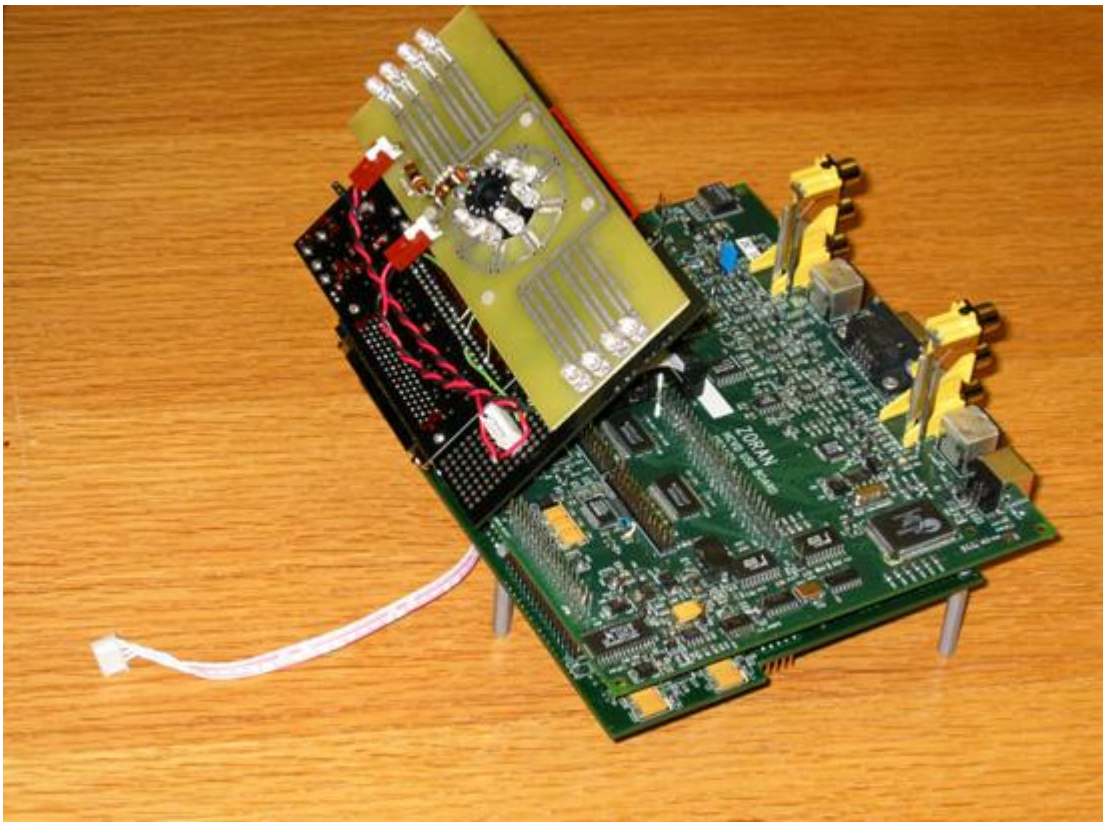


Figure 59. Amir et al.'s prototype of a hardware eye-detection sensor. Image processing is done on an on-board FPGA, making this a lightweight peripheral that can be connected via USB.

10.8 Summary

In this chapter we highlighted some of the factors that contribute to the high costs of eye-tracking systems and proposed several ideas and strategies that can be used to reduce the costs of these systems, ultimately resulting in more widespread use of the technology. The drive towards low-cost eye tracking has been recognized by the community and at the Eye Tracking Research and Applications Symposium held in San Diego in March 2006 the community backed the IPRIZE [7] — a 1 million-dollar grand challenge which aims to achieve a ten-fold improvement in eye-tracking technology, while at the same time making it affordable for the average person.

11 Conclusion

This chapter summarizes the contributions of this dissertation and synthesizes the knowledge gained over the course of this research by identifying the challenges in designing gaze-based interactions and presenting our guidelines for addressing these challenges.

11.1 Summary of Contributions

This dissertation presented several novel ***interaction techniques that use gaze information as a practical form of input***. In particular, it introduced a new technique for pointing and selection (Chapter 3) using a combination of eye gaze and keyboard. This approach overcomes the accuracy limitations of eye trackers and does not suffer from the Midas Touch problem. The pointing speed of this technique is comparable to that of a mouse. The original results showed a higher error rate than the mouse, which was addressed further in Chapter 9.

Chapter 4 introduced several techniques for gaze-enhanced scrolling, including the gaze-enhanced page up / page down approach which augments manual scrolling with additional information about the user's gaze position. It also introduced three techniques for automatic scrolling. These techniques are explicitly activated by the user; they scroll text in only one direction and can adjust the speed of the scrolling to match the user's reading speed. Additionally, it introduces the use of gaze-activated off-screen targets that allow the placement of both discrete and continuous document navigation commands on the bezel of the screen.

This dissertation also introduces the use of eye gaze for application switching (Chapter 5) and password entry (Chapter 6). It also revealed a fundamental problem

with using gaze as part of a zooming interface — zooming interfaces tend to magnify the error in the accuracy of the eye tracker (Chapter 7). Chapter 8 discussed several additional applications and interaction techniques that use gaze as a form of input.

This dissertation also presented new ***technologies for improving the interpretation of eye gaze as a form of input***. In particular, Chapter 9 revisits and deepens the exploration of some of the common underlying issues with eye tracking. We presented an algorithm for saccade detection and fixation smoothing, identified and addressed the problem of eye hand coordination when using gaze in conjunction with trigger-based activation and explored the use of focus points to provide users with a visual marker to focus on when using a gaze-based application.

Finally, Chapter 10 addresses the missing link by providing a discussion of the prospects for eye tracking to be made affordable and available for widespread use.

In keeping with the thesis statement in Chapter 1, the work of this dissertation shows that ***gaze can indeed be used as a practical form of input***. The following sections of this concluding chapter synthesize the lessons learnt from this research in the form of a list of challenges for design interaction and our proposed guidelines for addressing these challenges.

11.2 Design Challenges for Gaze Interaction

The design of interactions that incorporate gaze poses some unique challenges for interaction designers. In addition to overcoming the limitations of eye tracker accuracy, designers also need to be wary of several other issues.

Eye movements are noisy: Eye movements occur in the form of fixations and saccades. Even within fixations the eye jitters due to micro-saccades, drift and tremors. Any application that relies on using gaze data must be robust enough to tolerate this noise. The applications must have a robust model for interpreting gaze information in order to extract the right information from the noisy signal. This dissertation presented a saccade detection and fixation smoothing algorithm in Chapter 9 that can help to address this challenge.

Eye tracker accuracy: Eye trackers are only capable of providing limited accuracy, which imposes limits on the granularity at which eye tracking data can be used. The interaction design must account for this lack of eye tracker accuracy and be able to overcome it in a robust manner. In addition, the tracking accuracy may differ from person to person. Any application that uses gaze must provide sufficient controls to customize the implementation for an individual user. The interaction techniques described in this dissertation used several ways to overcome the accuracy issue. EyePoint (Chapter 3) uses magnification, the scrolling techniques in Chapter 4 use thresholds which are less sensitive to accuracy, EyeExposé (Chapter 5) and EyePassword (Chapter 6) use large targets.

Sensor lag: It is virtually impossible for the eye tracker to provide true real-time eye tracking. Since there will always be a lag between when the user looks at something and when the eye tracker detects the new gaze location, applications must accommodate this lag. In addition, algorithms that smooth and filter eye tracking data may introduce additional processing delays, which also need to be accounted for by the application designer. Section 9.2 of this dissertation explores this topic and accommodates for the latency in gaze data in the simulation.

The Midas Touch problem: As discussed in Chapter 2, the Midas Touch problem is the most critical design challenge when designing gaze-based interactions. It necessitates the disambiguation of when the user is looking and when the user intends to perform an action. Failure to do so can result in false activations which are not only annoying to the user but can be dangerous since they can accidentally trigger actions that the user may not have intended. By focusing on the design of the interaction techniques presented, as seen in the preceding chapters, it is possible to overcome the Midas Touch problem.

Maintaining the natural function of the eyes: The common misconception for gaze-enhanced interactions is that users will be winking and blinking at their computers. Such actions overload the normal function of the eyes and unless the user has no alternatives, they can be both fatiguing and annoying for the user. It is imperative for any gaze-based interaction technique to maintain the natural function of the eyes and not overload the visual channel. Other than the dwell-based

password entry and the use of off-screen targets, all the techniques presented in this dissertation are designed to maintain the natural function of the eyes.

Feedback: Designers need to rethink how they provide feedback to the user in the case of a gaze-based interaction. Providing visual feedback forces users to move their gaze to look at the feedback. Such an approach could lead to a scenario where the natural function of the eye is no longer maintained. This problem is illustrated by the example of providing visual feedback in a language-model based gaze typing system. The user must look at the keys to type, but must look away from the keys in order to examine the possible word options. Designers must therefore give careful thought to how the feedback is provided. Using an alternative channel such as audio feedback or haptic feedback may be more suitable for some applications which require the eyes to be part of the interaction technique. EyePassword (Chapter 6) provided users with audio feedback.

11.3 Design Guidelines for Gaze Interaction

Based on our experience with the design and evaluation of gaze-based interaction techniques, we would recommend the following guidelines for any designers using gaze as a form of input:

Maintain the natural function of the eyes: As mentioned in the previous work by Zhai, Jacob and others, it is imperative to maintain the natural function of the eye when designing gaze-based interactions. Our eyes are meant for looking. Using them for any other purpose overloads the visual channel and is generally undesirable for any gaze-based application. There are exceptions to this rule, such as when designing interfaces for disabled users who may not have the ability to use an alternative approach. However, in general, all gaze-based interactions should try to maintain the natural function of the eyes.

Augment rather than replace: Designers should consider using gaze as an augmented input. Attempts to replace existing interaction techniques with a gaze-only approach may not be as compelling as augmenting traditional techniques and devices with gaze information. Using gaze to provide context and as a proxy for the user's attention and attention can enable the development of new interactions when

used in conjunction with other modalities. In the techniques presented in this thesis, we use gaze in conjunction with the keyboard or mouse.

Focus on interaction design: The design of the interaction when using gaze-based applications is the most effective approach for overcoming the Midas Touch problem. Designers must consider the natural function of the eyes, the number of steps in the interaction, the amount of time it takes, the cost of an error/failure, the cognitive load imposed upon the user and the amount of fatigue the interaction causes among other things. The focus on interaction design was one of the key insights for this dissertation.

Improve the interpretation of eye movements: Since gaze-data is at best a noisy source of information, designers should carefully consider how to interpret this gaze data to estimate the user's attention and or intention. This may include using algorithms to improve the classification and analysis of gaze data, pattern recognition and using semantic information or additional sensor data to augment the designer's interpretation of the user's gaze. Chapter 9 of this dissertation addresses some of the issues with interpretation of eye gaze.

Task-oriented approach: Gaze may not be suitable for all applications! It is important to consider the task at hand when designing the gaze-based interaction. In some cases it is likely that other input modalities may be better suited. For example, using gaze to change radio stations in a car may not be a very good idea for obvious reasons. Using gaze-based pointing in applications such as Photoshop, which require fine grained motor-control, would also be undesirable. Designers must consider the task/use scenario before using gaze-based interaction.

Active vs. passive use of gaze information: Eye tracking as a form of input can be used either in an *active* mode, where the gaze is used to directly control/influence a certain task or in a *passive* way where the gaze is used to inform the system but the effect of the user's gaze may not be immediately apparent or may be communicated indirectly. We illustrate this point with eye tracking in cars. Using gaze to control the changing of radio station in the car would fall into the category of an active use of gaze information, i.e. the user must actively look at the device to perform the action. By contrast, using the user's gaze to let the car know that the

user is not looking at the road and then informing the user with a beep would be a passive use of eye gaze since in this case the user did not need to consciously perform an action. Designers should consider ways in which they can use gaze information passively before attempting to use active gaze-based control since passive use of gaze information has a better chance of maintaining the natural function of the eyes.

Attentive User Interfaces: As previously noted, gaze serves as a proxy for the user's attention and intention. Consequently application designers can leverage this information to design interfaces that blend seamlessly with the user's task flow. Gaze can be used to inform an interruption model of the user, making it possible to design interactions that are less intrusive and decrease the cognitive load. Chapter 8 of this dissertation presents several examples of attentive user interfaces (Gaze-contingent screen and power save, Gaze-enhanced multi-monitor coordination and No-nag IM windows).

11.4 Concluding Remarks

It is the author's hope and expectation that eye gaze tracking will soon be

Figure 60. Concept eye tracker — here the Apple MacBook Pro has been shown with two black bars in the top bezel, which can conceal the infrared illuminants necessary for eye tracking.

available in every desktop and laptop computer and its use as a standard form of input will be ubiquitous. As discussed in Chapter 10, technology and economic trends may soon make it possible for this vision to become a reality. Figure 60 shows a “concept” low-cost mass-market eye tracker, which could be easily incorporated into the bezel of a contemporary laptop. The combination of low-cost eye tracking and gaze-based interaction techniques has the potential to create the environment necessary for gaze-augmented input devices to become mass-market.

As eye-tracking devices improve in quality and accuracy and decrease in cost, interaction designers will have the ability to sense the user’s attention and intention. This has the potential to revolutionize traditional keyboard-and-mouse-centric interactions. The best form of human computer interaction is one that the user never even notices. Using gaze information has the potential to propel interactions in this direction.

Bibliography

[1] *Apple Mac OSX Aqua User interface feature: Exposé*, 2006. http://www.apple.com/macosx/features/expose/

[2] *Apple MacBook iSight camera*. Apple Computer: Cupertino, California, USA. http://www.apple.com/macbook/isight.html

[3] *Google Maps*. Google, Inc.: Mountain View, CA. http://maps.google.com

[4] *IBM BlueEyes Project*. IBM: Almaden. http://www.almaden.ibm.com/cs/BlueEyes/index.html

[5] *Microsoft PowerToys for Windows XP: Alt-Tab Replacement - TaskSwitch*. http://www.microsoft.com/windowsxp/downloads/powertoys/xppowertoys.mspx

[6] *PassFaces: patented technology that uses the brain's natural power to recognize familiar faces*. PassFaces Corporation. http://www.passfaces.com/products/passfaces.htm

[7] *RSI: Repetitive Strain Injury*, 2007. Wikipedia. http://en.wikipedia.org/wiki/Repetitive_strain_injury

[8] *Windows Vista; The Features*, 2006. http://www.microsoft.com/windowsvista/features/default.mspx

[9] Agarawala, A. and R. Balakrishnan. Keepin' It Real: Pushing the Desktop Metaphor with Physics, Piles and the Pen. In Proceedings of *CHI*. Montréal, Québec, Canada: ACM Press. pp. 1283-92, 2006.

[10] Amir, A., M. Flickner, and D. Koons, Theory for Calibration Free Eye Gaze Tracking. 2002, IBM Almaden Research.

[11] Amir, A., L. Zimet, A. Sangiovanni-Vincentelli, and S. Kao. An Embedded System for an Eye-Detection Sensor. *Computer Vision and Image Understanding, CVIU Special Issue on Eye Detection and Tracking* 98(1). pp. 104-23, 2005.

[12] Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing* 50(2). pp. 174, 2002.

[13] Ashmore, M., A. T. Duchowski, and G. Shoemaker. Efficient Eye Pointing with a FishEye Lens. In Proceedings of *Graphics Interface*. pp. 203-10, 2005.

[14] Asonov, D. and R. Agrawal. Keyboard Acoustic Emanations. In Proceedings of *IEEE Symposium on Security and Privacy*. Oakland, California, USA: IEEE. pp. 3-11, 2004.

[15] Babcock, J. S. and J. B. Pelz. Building a lightweight eyetracking headgear. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Antonio, Texas, USA: ACM Press. pp. 109-13, 2004.

[16] Baudisch, P., E. Cutrell, K. Hinkley, and A. Eversole. Snap-and-go: Helping Users Align Objects Without the Modality of Traditional

Snapping. In Proceedings of *CHI 2005*. Portland, Oregon: ACM Press. pp. 301-10, 2005.

[17] Bederson, B. B., J. Grosjean, and J. Meyer. Toolkit Design for Interactive Structured Graphics. *IEEE Transaction on Software Engineering* 30(8). pp. 1-12, 2003.

[18] Bederson, B. B. and J. D. Hollan. Pad++: A Zoomable Graphical Interface System. In Proceedings of *CHI*. Denver, Colorado, USA: ACM Press. pp. 23-24, 1995.

[19] Bederson, B. B., J. D. Hollan, J. Stewart, D. Rogers, A. Druin, and D. Vick. A Zooming Web Browser. *Human Factors and Web Development*, 1997.

[20] Beinhauer, W. A Widget Library for Gaze-based Interaction Elements. In Proceedings of *ETRA: Eye Tracking Research and Applications Symposium*. San Diego, California, USA: ACM Press. pp. 53-53, 2006.

[21] Benko, H. and S. Feiner. Multi-Monitor Mouse. In Proceedings of *CHI 2005 Extended Abstracts*. Portland, Oregon: ACM Press. pp. 1208-11, 2005.

[22] Berger, Y., A. Wool, and A. Yeredor. Dictionary Attacks Using Keyboard Acoustic Emanations. In Proceedings of *Computer and Communications Security (CCS)*. Alexandria, Virginia, USA, 2006.

[23] Beymer, D., S. P. Farrell, and S. Zhai. System and method for selecting and activating a target object using a combination of eye gaze and key presses. USA Patent 2005, International Business Machines Corporation

[24] Beymer, D. and D. M. Russell. WebGazeAnalyzer: A System for Capturing and Analyzing Web Reading Behavior Using Eye Gaze. In

Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 1913-16, 2005.

[25] Bradski, G. R. The OpenCV Library, *Dr. Dobb's Software Tools for the Professional Programmer*, November 2000.

[26] Buswell, G. T., *How People Look at Pictures: A Study of the Psychology of Perception in Art*: The University of Chicago Press pp. 1935.

[27] Buxton, W. Chunking and Phrasing and the Design of Human-Computer Dialogues. In Proceedings of *IFIP World Computer Congress*. Dublin, Ireland. pp. 475-80, 1986.

[28] Card, S. K., W. K. English, and B. J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys, for text selection on a CRT. *Ergonomics* 21(8). pp. 601-13, 1978.

[29] Chen, M. Leveraging the asymmetric sensitivity of eye contact for videoconference. In Proceedings of *CHI*. Minneapolis, Minnesota: ACM Press. pp. 49-56, 2002.

[30] Cockburn, A., J. Savage, and A. Wallace. Tuning and Testing Scrolling Interfaces that Automatically Zoom. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 71-80, 2005.

[31] Czerwinski, M., G. Smith, T. Regan, B. Meyers, G. Robertson, and G. Starkweather. Toward Characterizing the Productivity Benefits of Very Large Displays. In Proceedings of *INTERACT*: IOS Press, 2003.

[32] Dalton, A. B. and C. S. Ellis. Sensing User Intention and Context for Energy Management. In Proceedings of *9th Workshop on Hot Topics in Operatings Systems (HotOS IX)*: USENIX Press, 2003.

[33] Damany, S. and J. Bellis, *It's Not Carpal Tunnel Syndrome! RSI Theory and Therapy for Computer Professionals*. Philadelphia: Simax pp. 2000.

[34] Douglas, S. A. and A. K. Mithal. The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. In Proceedings of *CHI*: ACM Press. pp. 411-16, 1994.

[35] Dragunov, A. N., T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. TaskTracer: a desktop environment to support multi-tasking knowledge workers. In Proceedings of *IUI*. San Diego, California, USA: ACM Press. pp. 75-82, 2005.

[36] Duchowski, A. T., *Eye Tracking Methodology: Theory and Practice*: Springer. 227 pp. 2003.

[37] Duchowski, A. T., N. Cournia, B. Cumming, D. McCallum, and A. Grampadhye. Visual Diectic Reference in a Collaborative Environment. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Antonio, Texas, USA. pp. 35-40, 2004.

[38] Farrell, S. P. and S. Zhai. System and method for selectively expanding or contracting a portion of a display using eye-gaze tracking. USA Patent 2005, International Business Machines Corporation

[39] Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47(6). pp. 381-91, 1954.

[40] Fono, D. and R. Vertegaal. EyeWindows: Evaluation of Eye-Controlled Zooming Windows for Focus Selection. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 151-60, 2005.

[41] Golle, P. and D. Wagner, *Cryptanalysis of a Cognitive Authentication Scheme*, International Association for Cryptologic Research, July 31 2006.

[42] Hansen, D. W. and J. P. Hansen. Eye Typing with Common Cameras. In Proceedings of *Eye Tracking Research and Applications (ETRA) Symposium*. San Diego, California: ACM Press. pp. 55, 2006.

[43] Hansen, D. W., D. MacKay, and J. P. Hansen. Eye Tracking off the Shelf. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Antonio, Texas, USA: ACM Press. pp. 58, 2004.

[44] Henderson, D. A., Jr. and S. K. Card. Rooms: The Use of Multiple Virtual Workspaces to Reduce Space Contention in a Window-Based Graphical User Interface. *ACM Transactions on Graphics* 5(3). pp. 211-43, 1986.

[45] Henessey, C., B. Noureddin, and P. Lawrence. A Single Camera Eye-Gaze Tracking System with Free Head Motion. In Proceedings of *ETRA: Eye Tracking Research and Applications Symposium*. San Diego, California, USA: ACM Press. pp. 87-94, 2006.

[46] Hinckley, K., E. Cutrell, S. Bathiche, and T. Muss. Quantitative analysis of scrolling techniques. In Proceedings of *CHI*. Minneapolis, Minnesota, USA: ACM Press. pp. 65-72, 2002.

[47] Hoanca, B. and K. Mock. Screen Oriented Technique for Reducing the Incidence of Shoulder Surfing. In Proceedings of *International Conference on Security and Management (SAM)*. Las Vegas, Nevada, USA, 2005.

[48] Hoanca, B. and K. Mock. Secure Graphical Password System for High Traffic Public Areas. In Proceedings of *ETRA - Eye Tracking Research*

and Applications Symposium. San Diego, California, USA: ACM Press. pp. 35, 2006.

[49] Hutchings, D. R., G. Smith, B. Meyers, M. Czerwinski, and G. Robertson. Display Space Usage and Window Management Operation Comparisons between Single Monitor and Multiple Monitor Users. In Proceedings of *AVI*. Gallipoli (LE), Italy: ACM Press. pp. 32-39, 2004.

[50] Hutchings, D. R. and J. Stasko. mudibo: Multiple Dialog Boxes for Multiple Monitors. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 1471-74, 2005.

[51] Huynh, D. F., S. M. Drucker, P. Baudisch, and C. Wong. Time Quilt: Scaling up Zoomable Photo Browsers for Large, Unstructured Photo Collections. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 1937-40, 2005.

[52] Jacob, R. J. K. The Use of Eye Movements in Human-Computer Interaction Techniques: What You Look At is What You Get. In Proceedings of *ACM Transactions in Information Systems*. pp. 152-69, 1991.

[53] Jacob, R. J. K. and K. S. Karn, Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises, in *The Mind's eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyona, R. Radach, and H. Deubel, Editors. Elsevier Science: Amsterdam. pp. 573-605, 2003.

[54] Jobs, S. P. and D. J. Lindsay. Computer interface having a single window mode of operation. USA Patent 2005, Apple Computer, Inc.

[55] Kuhn, M. G., Electromagnetic Eavesdropping Risks of Flat-Panel Displays, in *4th Workshop on Privacy Enhancing Technologies, LNCS*. Springer-Verlag: Berlin / Heidelberg. pp. 23–25, 2004.

[56] Kumar, M., *GUIDe Saccade Detection and Smoothing Algorithm*. Technical Report CSTR 2007-03, Stanford University, Stanford 2007. http://hci.stanford.edu/cstr/reports/2007-03.pdf

[57] Kumar, M., *IRcam: Instructions for modifying a consumer web camera to work in the infrared spectrum*, 2006. Palo Alto, California. http://www.sneaker.org/projects/IRcam/index.html

[58] Kumar, M., T. Garfinkel, D. Boneh, and T. Winograd, *Reducing Shoulder-surfing by Using Gaze-based Password Entry*. Technical Report CSTR 2007-05, Stanford University, Stanford 2007. http://hci.stanford.edu/cstr/reports/2007-05.pdf

[59] Kumar, M., T. Garfinkel, D. Boneh, and T. Winograd. Reducing Shoulder-surfing by Using Gaze-based Password Entry. In Proceedings of *Symposium on Usable Privacy and Security (SOUPS)*. Pittsburgh, PA: ACM Press, 2007.

[60] Kumar, M., A. Paepcke, and T. Winograd, *EyeExposé: Switching Applications with Your Eyes*. Technical Report CSTR 2007-02, Stanford University, Stanford 2007. http://hci.stanford.edu/cstr/reports/2007-02.pdf

[61] Kumar, M., A. Paepcke, and T. Winograd. EyePoint: Practical Pointing and Selection Using Gaze and Keyboard. In Proceedings of *CHI*. San Jose, California, USA: ACM Press, 2007.

[62] Kumar, M. and T. Winograd, *Gaze-enhanced Scrolling Techniques*. Technical Report CSTR 2007-11, Stanford University, Stanford, CA 2007. http://hci.stanford.edu/cstr/reports/2007-11.pdf

[63] Kumar, M. and T. Winograd. GUIDe: Gaze-enhanced UI Design. In Proceedings of *CHI*. San Jose, California, USA: ACM Press, 2007.

[64] Kumar, M., T. Winograd, and A. Paepcke. Gaze-enhanced Scrolling Techniques. In Proceedings of *CHI*. San Jose, California, USA: ACM Press, 2007.

[65] Laarni, J. Searching for Optimal Methods of Presenting Dynamic Text on Different Types of Screens. In Proceedings of *NordiCHI*. Arhus, Denmark: ACM Press. pp. 219-22, 2002.

[66] Lankford, C. Effective Eye-Gaze Input into Windows. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. Palm Beach Gardens, Florida, USA: ACM Press. pp. 23-27, 2000.

[67] LC Technologies, I., *The EyeGaze Communication System,* 2006. LC Technologies: McLean, Virginia. http://www.eyegaze.com/2Products/Disability/Disabilitymain.htm

[68] Li, D., J. Babcock, and D. J. Parkhurst. openEyes: A Low-Cost Head-Mounted Eye-Tracking Solution. In Proceedings of *ETRA; Eye Tracking Research and Applications Symposium*. San Diego, California, USA: ACM press, 2006.

[69] Maeder, A., C. Fookes, and S. Sridharan. Gaze Based User Authentication for Personal Computer Applications. In Proceedings of *International Symposium on Intelligent Multimedia, Video and Speech Processing*. Hong Kong: IEEE. pp. 727-30, 2004.

[70] Majaranta, P., A. Aula, and K.-J. Räihä. Effects of Feedback on Eye Typing with a Short Dwell Time. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Antonio, Texas, USA: ACM Press. pp. 139-46, 2004.

[71] Majaranta, P., I. S. MacKenzie, A. Aula, and K.-J. Räihä. Auditory and Visual Feedback During Eye Typing. In Proceedings of *CHI*. Ft. Lauderdale, Florida, USA: ACM Press. pp. 766-67, 2003.

[72] Majaranta, P. and K.-J. Räihä. Twenty Years of Eye Typing: Systems and Design Issues. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. New Orleans, Louisiana, USA: ACM Press. pp. 15-22, 2002.

[73] McGuffin, M. and R. Balakrishnan. Acquisition of Expanding Targets. In Proceedings of *CHI*. Minneapolis, Minnesota, USA: ACM Press. pp. 57-64, 2002.

[74] Miniotas, D., O. Špakov, I. Tugoy, and I. S. MacKenzie. Speech-Augmented Eye Gaze Interaction with Small Closely Spaced Targets. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Diego, California, USA: ACM Press. pp. 67-72, 2006.

[75] Monrose, F., M. K. Reiter, and S. Wetzel. Password hardening based on keystroke dynamics. *International Journal of Information Security* 1(2). pp. 69-83, 2002.

[76] Monty, R. A., J. W. Senders, and D. F. Fisher, *Eye Movements and the Higher Psychological Functions*. Hillsdale, New Jersey, USA: Erlbaum pp. 1978.

[77] Morimoto, C., D. Koons, A. Amir, and M. Flickner. Pupil Detection and Tracking Using Multiple Light Sources. *Image and Vision Computing* 18(4). pp. 331-36, 2000.

[78] Morimoto, C. H., A. Amir, and M. Flickner. Free Head Motion Eye Gaze Tracking Without Calibration. In Proceedings of *CHI*. Minneapolis, Minnesota, USA: ACM Press. pp. 586-87, 2002.

[79] Newman, M. W., J. Lin, J. I. Hong, and J. A. Landay. DENIM: An Informal Web Site Design Tool Inspired by Observations of Practice. 18(3). pp. 259-324, 2003.

[80] Nguyen, K., C. Wagner, D. Koons, and M. Flickner. Differences in the Infrared Bright Pupil Response of Human Eyes. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. New Orleans, Louisiana, USA: ACM Press. pp. 133-38, 2002.

[81] Norman, D. A., *Emotional Design: Why we love (or hate) everyday things*. New York: Basic Books. 256 pp. 2004.

[82] Norman, D. A. and D. Fisher. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors* 25(5). pp. 509-19, 1982.

[83] Ohno, T. and N. Mukawa. A Free-head, Simple Calibration, Gaze Tracking System That Enables Gaze-Based Interaction. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. San Antonio, Texas, USA. pp. 115-22, 2004.

[84] Ohno, T., N. Mukawa, and S. Kawato. Just Blink Your Eyes: A Head-Free Gaze Tracking System. In Proceedings of *CHI*. Ft. Lauderdale, Florida, USA: ACM Press. pp. 950-51, 2003.

[85] Oviatt, S. Ten Myths of Multimodal Interaction. *Communications of the ACM* 42(11). pp. 74-81, 1999.

[86] Pascarelli, E. and D. Quilter, *Repetitive Strain Injury: A Computer User's Guide*: John Wiley & Sons, Inc. 240 pp. 1994.

[87] Poynter Institute and Eyetools, Inc., *Eyetrack III: Online News Consumer Behavior in the Age of Multimedia*, 2004. http://poynterextra.org/eyetrack2004/index.htm

[88] Qvarfordt, P. and S. Zhai. Conversing with the User Based on Eye-Gaze Patterns. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 221-30, 2005.

[89] Rayner, K. Eye Movments in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin* 124(3). pp. 372-422, 1998.

[90] Robertson, G., *et al.* Scalable Fabric: Flexible Task Management. In Proceedings of *AVI*. Gallipoli (LE), Italy: ACM Press. pp. 85-89, 2004.

[91] Robertson, G. G., *et al.* The Task Gallery: a 3D window manager. In Proceedings of *CHI*. The Hague, Amsterdam: ACM Press. pp. 494-501, 2000.

[92] Roth, V., K. Richter, and R. Freidinger. A PIN-Entry Method Resilient Against Shoulder Surfing. In Proceedings of *CCS: Conference on Computer and Communications Security*. Washington DC, USA: ACM Press. pp. 236-45, 2004.

[93] RSA Security, I., *RSA SecurID Authentication.* http://www.rsasecurity.com/node.asp?id=1156

[94] Sadasivan, S., J. S. Greenstein, A. K. Gramopadhye, and A. T. Duchowski. Use of Eye Movements as a Feedforward Training for a

Synthetic Aircraft Inspection Task. In Proceedings of *CHI*. Portland, Oregon, USA: ACM Press. pp. 141-49, 2005.

[95] Salvucci, D. D. Inferring Intent in Eye-Based Interfaces: Tracing Eye Movements with Process Models. In Proceedings of *CHI*. Pittsburgh, Pennsylvania, USA: ACM Press. pp. 254-61, 1999.

[96] Salvucci, D. D. Intelligent Gaze-Added Interfaces. In Proceedings of *CHI*. The Hague, Amsterdam: ACM Press. pp. 273-80, 2000.

[97] Salvucci, D. D. and J. H. Goldberg. Identifying Fixations and Saccades in Eye-Tracking Protocols. In Proceedings of *ETRA: Eye Tracking Research & Applications Symposium*. Palm Bech Gardens, Florida, USA: ACM Press. pp. 71-78, 2000.

[98] Sibert, L. E. and R. J. K. Jacob. Evaluation of Eye Gaze Interaction. In Proceedings of *CHI*. The Hague, Amsterdam: ACM Pres, 2000.

[99] Simonite, T. Tactile passwords could stop ATM 'shoulder-surfing', *New Scientist*, October 6, 2006.

[100] Smith, G., *et al.* Groupbar: The Taskbar Evolved. In Proceedings of *OZCHI*: ACM Press. pp. 1-10, 2003.

[101] Song, D. X., D. Wagner, and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In Proceedings of *10th USENIX Security Symposium*. Washington DC, USA: The USENIX Association, 2001.

[102] Suo, X. and Y. Zhu. Graphical Passwords: A Survey. In Proceedings of *Annual Computer Security Applications Conference*. Tucson, Arizona, USA, 2005.

[103] Tan, D. S., P. Keyani, and M. Czerwinski. Spy-Resistant Keyboard: Towards More Secure Password Entry on Publicly Observable Touch

Screens. In Proceedings of *OZCHI - Computer-Human Interaction Special Interest Group (CHISIG) of Australia*. Canberra, Australia: ACM Press, 2005.

[104] Thorpe, J., P. C. van Oorschot, and A. Somayaji. Pass-thoughts: authenticating with our minds. In Proceedings of *New Security Paradigns Workshop*. Lake Arrowhead, California, USA: ACM Press. pp. 45-56, 2005.

[105] Tobii Technology, AB, Drift Effects, in *User Manual: Tobii Eye Tracker and ClearView Analysis Software*. Tobii Technology AB. p. 15, 2006.

[106] Tobii Technology, AB, *MyTobii Communication Software*, 2006. Tobii Technology AB: Danderyd, Sweden. http://www.tobii.com//default.asp?sid=555

[107] Tobii Technology, AB, *Tobii 1750 Eye Tracker*, 2006. Sweden. http://www.tobii.com

[108] Tobii Technology, A., *MyTobii P10 - portable eye-controlled communcation device*, 2006. http://www.tobii.com

[109] Wallace, A., J. Savage, and A. Cockburn. Rapid Visual Flow: How Fast Is Too Fast? In Proceedings of *5th AUIC: Australasian User Interface Conference*. Dunedin: Australian Computer Society, Inc. pp. 117-22, 2004.

[110] Ware, C. and H. H. Mikaelian. An Evaluation of an Eye Tracker as a Device for Computer Input. In Proceedings of *CHI + Graphics Interface*. Toronto, Ontario, Canada: ACM Press. pp. 183-88, 1987.

[111] Weinshall, D. Cognitive Authentication Schemes Safe Against Spyware (Short Paper). In Proceedings of *IEEE Symposium on Security and Privacy*. Oakland, California, USA: IEEE, 2006.

[112] Welch, G. and G. Bishop, *An Introduction to the Kalman Filter*. Technical Report TR 95-041, University of North Carolina, Chapel Hill 1995 (updated 2006).

[113] Wiedenbeck, S., J. Waters, L. Sobrado, and J.-C. Birget. Design and Evaluation of a Shoulder-Surfing Resistant Graphical Password Scheme. In Proceedings of *AVI*. Venezia, Italy: ACM Press. pp. 177-84, 2006.

[114] Yamato, M., A. Monden, K.-i. Matsumoto, K. Inoue, and K. Torii. Button Selection for General GUIs Using Eye and Hand Together. In Proceedings of *AVI*. Palermo, Italy: ACM Press. pp. 270-73, 2000.

[115] Yarbus, A. L., *Eye Movements and Vision*. New York: Plenum Press pp. 1967.

[116] Zhai, S. What's in the Eyes for Attentive Input, *Communications of the ACM*, vol. 46(3): pp. 34-39, March, 2003.

[117] Zhai, S., S. Conversy, M. Beaudouin-Lafon, and Y. Guiard. Human On-line Response to Target Expansion. In Proceedings of *CHI*. Ft. Lauderdale, florida, USA: ACM Press. pp. 177-84, 2003.

[118] Zhai, S., C. Morimoto, and S. Ihde. Manual and Gaze Input Cascaded (MAGIC) Pointing. In Proceedings of *CHI*. Pittsburgh, Pennsylvania, USA: ACM Press. pp. 246-53, 1999.

[119] Zhai, S., B. A. Smith, and T. Selker. Improving Browsing Performance: A study of four input devices for scrolling and pointing tasks. In Proceedings of *IFIP Interact*. Sydney, Australia. pp. 286-92, 1997.

[120] Zhuang, L., F. Zhou, and J. D. Tygar. Keyboard Acoustic Emanations Revisited. In Proceedings of *Computer and Communications Security (CCS)*. Alexandria, Virgina, USA: ACM Press. pp. 373-82, 2005.