ACNOWLEDMENTS

Firstly, I would like to thank the principal Pro.Dr.Suresh Kumar for kindly allowing me to pursue my seminar and providing students with all necessary infrastructure and facilities. I also take this opportunity to thank the Head of the Computer Science Department, Pro. Preetha Theresa Joy for her valuable approval, suggestions and help rendered.

Secondly, I would like to thank my Seminar coordinator Mr.Murali for his approval, evaluation and the conduct of the seminars. I also like to thank my seminar guide for his help, suggestions and feedback.

Finally I thank my friends, seniors and well wishers who helped me in preparing this seminar.
**ABSTRACT**

The eyes are a rich source of information for gathering context in our everyday lives. Using user's 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 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 focus of this research is to add gaze information and provide viable alternatives to traditional interaction techniques, which users may prefer to use depending upon their abilities, tasks and preferences such as pointing and selection, scrolling and document navigation, application switching, password entry, zooming and other applications.
TABLE OF CONTENTS

1. INTRODUCTION...................................................................................................................

2. BACKGROUND.....................................................................................................................

   2.1. MOTIVATION.................................................................................................................

   2.2. GAZE AS A FORM OF INPUT.........................................................................................

   2.3. HISTORY OF EYE TRACKING.......................................................................................  

       2.3.1. Scleral coil contact lens method............................................................................
INTRODUCTION

The eyes are one of the most expressive features of the human body for nonverbal, 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.

The keyboard and mouse which have long been the dominant forms of input have bandwidth problem. That is the bandwidth from the computer to the user is far greater than the bandwidth from the user to the computer. In this dissertation it is 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. Gaze information can be used as a practical form of input. The 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. 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.

There are some novel interaction techniques which explore the use of gaze as an augmented input to perform everyday computing tasks such as pointing and selection, scrolling and document navigation, application switching, password entry, zooming and other applications. The gaze-based interaction techniques is either comparable to or an improvement over existing traditional mechanisms. The gaze data can be filtered and smoothed and eye-hand coordination for gaze plus trigger activated interaction techniques can be improved for giving better result. Focus points are also provided to help improve the accuracy of eye tracking and the user experience for using gaze-based interaction techniques.
2. BACKGROUND

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. However, even in this short time span increasing amounts of repetitive strain injuries (RSI) have emerged from overuse of the keyboard and mouse. 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.

![Figure 1. Tendonitis: a form of repetitive strain injury (RSI) caused by excessive use of the keyboard and particularly the mouse.](image)

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 much useful functionality. 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. It's found that for research more subtle form of input is needed — eye gaze.

2.2. Gaze as a Form of Input

Why would one want to use eye movements for interactive input?

- 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 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.

**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. 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 used eye tracking studies to examine how people look at pictures. Yarbus 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.

**2.3.1. Scleral coil contact lens method**

The scleral contact lens which was inserted in the eye of the subject, 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.
2.3.2. Electro-oculography (EOG) approach

The eyes of the subject 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.

2.3.3. Head mounted eye tracker

Head mounted eye trackers have been developed to fix the frame of reference for the eyes relative to the motion of the head. Some head mounted eye 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.
2.3.4. The Tobii 1750 eye tracker

The Tobii 1750 eye tracker uses remote video-based eye tracking for desktop eye tracking. 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. This 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. It provides 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.

In Tobii 1750 eye tracker 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.

2.3.5. Other Techniques

In SRI eye tracker approach it should be noted that this unit required the subject’s head to be held stationary. The BlueEyes project at IBM Almaden developed remote video-based eye trackers which used infra-red illumination. Several commercial systems have now been developed which use a similar approach for eye tracking and provide non-encumbering, remote, video-based eye tracking.

2.4. ISSUES OF GAZE INPUT

The eyes are fast, require no training and eye gaze provides context for our actions. 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.
2.4.1 Eye Movements are Noisy
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 subjects are asked to follow the outlines of geometrical figures as smoothly as possible. Yarbus, also points out that while fixations may appear to be dots, in reality, the eyes are not stable even during fixations due to drifts, tremors and involuntary micro saccades.

![Figure 8. Trace of eye movements when subjects are asked to follow the lines of the figures as smoothly as possible.](image)

2.4.2. 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. 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, 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. The accuracy of the eye-tracking data usually deteriorates due to a drift effect caused by changes in eye characteristics over time. 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 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. 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.
2.4.3 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. 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. 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.

3. POINTING AND SELECTION
Everyone using 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.

3.1 Related Work
Zhai et al. 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 MAGIC approach leverages Fitts’ Law 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.
Follow-on work to MAGIC at IBM by Beymer, Farrell and Zhai 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, semantic information is used to predictively select the most likely target with error-correction and refinement done using cursor keys.

### 3.2 EyePoint

EyePoint system uses a two-step progressive refinement process that is fluidly stitched together in a look-press-look-release action. 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 this case, the calibration is performed using the APIs provided in the Software Development Kit for the Tobii 1750 Eye Tracker. 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.
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.

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. 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.

Figure 12. Focus points - a grid of orange dots overlaid on the magnified view helps users focus their gaze.

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

Some points noted from above discussion are 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
   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. A modified version of Salvucci’s Dispersion Threshold Identification fixation detection algorithm is used to determine the location of the current fixation. 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. 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.

### 3.2.3. ADVANTAGES

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, 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.

### 4. SCROLLING
Scrolling is an essential part of our everyday computing experience. 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. Both manual and automatic scrolling is implemented on a Tobii 1750 tracker.

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. This section describes a common problem with the use of the Page Down action and proposes 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.
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 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 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 has shown that the typical eye movements for a subject reading text conform to Figure 27. Beymer et al. 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”.

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 text 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 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 adding an invisible threshold line towards the bottom 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.

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 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 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.

4.3.1 Dwell vs. Micro-Dwell based activation

Document navigation requires either 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. 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. 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.

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.

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. While there has been extensive research in the area of window managers and task management, 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 Windows Flip and Flip3D. 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.

5.1 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 Eye Windows. 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 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.

The accuracy of eye trackers is insufficient to be able to point to small targets. 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.

### 5.2 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 shows 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:

\[
T_{\text{mouse}} = t_{\text{activation}} + t_{\text{visual search}} + t_{\text{acquire mouse}} + t_{\text{acquire cursor}} + t_{\text{move mouse}} + t_{\text{click mouse}}
\]

where \(t_{\text{activation}}\) is the time for the user to press the hotkey or move the mouse to a corner of the screen to activate application switching; \(t_{\text{visual search}}\) is the amount of time it takes the user to locate the target on the screen; \(t_{\text{acquire mouse}}\) is the amount of time it takes the user to move the hands from the keyboard to the mouse; \(t_{\text{acquire cursor}}\) is the amount of time to locate the cursor on the screen and \(t_{\text{move mouse}}\) and \(t_{\text{click mouse}}\) 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:  
\[ T_{\text{eyeexposé}} = t_{\text{activation}} + t_{\text{visual search}} + t_{\text{release}} \]

where \( t_{\text{release}} \) is the time to release the hotkey. We expect \( t_{\text{release}} \) to be considerably  
lower than \( (t_{\text{acquire mouse}} + t_{\text{acquire cursor}} + t_{\text{move mouse}} + t_{\text{click mouse}}) \). 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.

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 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.

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

6.1. Motivation for Eye Tracking

Devices such as Apple’s MacBook laptops include a built-in iSight camera 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.
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 an on-screen keyboard. 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, the users fix their gaze for a moment. The second method is multimodal — 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. 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, 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 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. 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 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 saccade2 detection and fixation smoothing algorithm to provide more reliable data for detecting fixations.

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 Gaze-contingent Semantic Zooming

In each of the scenarios described above the real region of interest is indicated by the user’s gaze and therefore, we propose to use the user’s gaze to indicate the region of interest for zooming. Since most zooming user interfaces use some form of semantic zooming, we call this approach *gaze-contingent semantic 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.

### 7.2 Prototype Implementations

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

#### 7.2.1 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 $P + \hat{\theta}$, where $\hat{\theta}$ 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 zâ, 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.

### 7.2.2 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.

### 7.2.3 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.

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 were 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 multimonitor 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 nonurgent 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 finegrained 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.

9. OVERVIEW

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 realtime 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 gazed-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 (Gazecontingent screen and power save, Gaze-enhanced multi-monitor coordination and No-nag IM windows).

10. CONCLUSION

It is the hope and expectation that eye gaze tracking will soon be 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.

10. REFERENCES


