Gaze-enhanced User Interface Design

MANU KUMAR, TERRY WINOGRAD, ANDREAS PAEPCKE, JEFF KLINGNER
Stanford University, HCI Group

The eyes are a rich source of information for gathering context in our everyday lives. 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. 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. Our research explores how gaze information can be effectively used as an augmented input in addition to traditional input devices.

We present several novel prototypes that explore the use of gaze as an augmented input to perform everyday computing tasks. In particular we explore the use of gaze for pointing and selection, scrolling, application switching and password entry. We present 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 while minimizing false activations and without overloading the visual channel. We also 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.

Categories and Subject Descriptors: H5.2. [**Information Interfaces and Presentation**]: User Interfaces-Input devices and strategies, H5.2. [**Information Interfaces and Presentation**]: User Interfaces-Windowing Systems, H5.m. [**Information interfaces and Presentation**]: Miscellaneous.
General Terms: Human Factors, Algorithms, Performance, Design
Additional Key Words and Phrases: Eye tracking, Gaze input, Gaze-enhanced User Interface Design, GUIDe, Pointing and Selection, Eye Pointing, Application Switching, Automatic Scrolling, Scrolling, Saccade detection, Fixation smoothing, Eye-hand coordination, Focus points.

1. INTRODUCTION

The eyes are a rich source of information for gathering context in our everyday lives. We look at the eyes in order 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 [38]. 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

This research was supported by Stanford Media X and the Stanford School of Engineering.
Authors' addresses: Manu Kumar (contact author), Gates Building, Room 382, 353 Serra Mall, Stanford, CA 94305-9035; email: sneaker@stanford.edu

practical for able-bodied users to use gaze as a form of input in addition to keyboard and mouse. Our research explores how gaze information can be effectively used as an augmented input in addition to traditional input devices.

The focus of our research is to augment rather than replace existing interaction techniques. We present several novel prototypes that explore the use of gaze as an augmented input to perform everyday computing tasks. In particular, we explore the use of gaze for pointing and selection, scrolling, application switching and password entry. We present 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 while minimizing false activations and without overloading the visual channel. We also 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.

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

2. CHALLENGES FOR GAZE INPUT

The eyes are fast, require no training and eye gaze provides context for our actions [9, 13, 14, 38]. 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.

Eye Movements are Noisy

As noted by Yarbus [37], 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 60'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 (Figure 1).

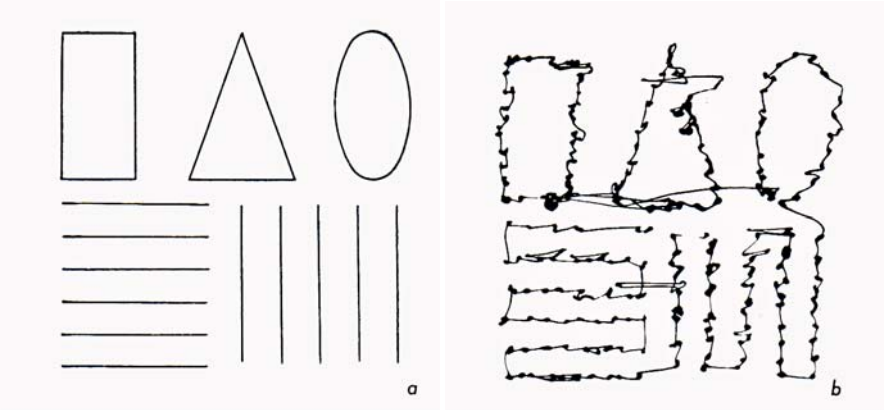


Figure 1. Trace of eye movements when subjects are asked to follow the lines of the figures as smoothly as possible. Source: Yarbus, 1967.

Yarbus also points out that while fixations may appear to be dots in Figure 1, in reality the eyes are not stable even during fixations due to drifts, tremors and involuntary micro-saccades (Figure 2).

Eye Tracker Accuracy

Current eye trackers, especially remote video based eye trackers, claim to be accurate to about 0.5˚ - 1˚ of visual angle[1]. 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 [2, 34]. 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 3). In addition, current eye trackers require calibration (though some require only a one-time calibration). The accuracy of the eye-tracking data usually deteriorates over time due to a drift effect caused by changes in eye characteristics over time [33]. Users' posture also changes over time as they begin to slouch or lean after some minutes of sitting. This results in the position or angle of their head changing. The

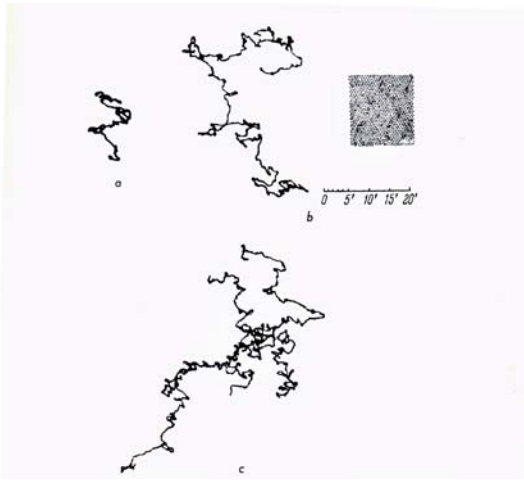


Figure 2. Fixation jitter due to drifts, tremors and involuntary micro-saccades, Source: Yarbus, 1967

[1] Tobii has recently announced an eye tracker that claims an accuracy of 0.25˚, but we have not been able to test this unit as of the time of writing of this manuscript.

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 [3]. 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.

Figure 3. Confidence interval of eye tracker accuracy. Inner circle is 0.5˚. Outer circle is 1.0˚.

Eye Trackers also introduce a sensor lag of about 5-33ms for processing of the data and to determine the current position of the user's gaze. The Tobii eye tracker used in our research claims to have a latency of 35ms.

The Midas Touch Problem

Mouse and keyboard actions are deliberate acts which do not require disambiguation. The eyes, however, are a perceptual organ designed for looking and are an always-on device [14]. It is therefore necessary to distinguish between involuntary or 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" [13].

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

3. GAZE-BASED INTERACTION TECHNIQUES

The Gaze-enhanced User Interface Design project in the HCI Group at Stanford University explores how gaze information can be used as a practical form of input. We have developed several novel interaction techniques which use gaze in addition to existing input modalities such as keyboard and mouse. We present here our work on gaze-based pointing and selection [19] and gaze-enhanced scrolling techniques. We will also briefly touch upon our work in application switching [18], password entry [17], zooming and various gaze-based utilities. However, a detailed discussion of the latter ideas is beyond the scope of this article. Additional papers and information on our research can be found on the project website [16].

4. POINTING AND SELECTION

We began our research by conducting a contextual inquiry into 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 click on links while surfing the Web. 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. For a gaze-based pointing technique to be truly useful, it should support all of the above fundamental pointing 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, trackpad or trackpoint).

RELATED WORK

Jacob [13] 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. [40] 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 at. 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.

Salvucci and Anderson [31] 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 *gaze button* 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. [36] also propose an augmented approach, in which gaze is used to position the cursor, but clicking is still performed using the mouse button. Their approach used automatic and manual adjustment modes. 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 [21] proposes 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 able-bodied users. By contrast, our approach innovates on the interaction techniques to make the interaction fluid and simple for all users.

Follow-on work to MAGIC at IBM [10] 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 [4], 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. [2] 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. The use of a fish-eye lens for magnification is debatable. 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

Figure 4. Using EyePoint - 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.

motion of objects within the lens' field of view in a direction opposite to that of the lens' motion.

Fono and Vertegaal [11] also use eye input with key activation. They show that key activation was preferred by users over automatic activation.

Finally, Miniotas et al. [25] present a speech-augmented eye-gaze interaction technique in 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 in which the user is correcting selection errors using speech.

EyePoint

Our system, EyePoint, uses a two-step progressive refinement process fluidly stitched together in a look-press-look-release action (Figure 4). This makes it possible to compensate for the accuracy limitations of current state-of-the-art eye trackers. Our approach allows 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 [34]. 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 simply 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 brings up 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 4).

To abort an action the user can look away or 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 5). 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.

Figure 5. 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 as soon as 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.

Design Principles

We agreed with Zhai [25] that overloading the visual channel for a motor control task is undesirable. We therefore resolved to push the envelope on the interaction design to determine if there was a way to use eye gaze for practical pointing *without* overloading the visual channel for motor control.

Another basic realization was from Fitts' law - that providing larger targets improves the speed and accuracy of pointing. 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 [10].

As recognized in prior work [2, 11, 21, 24, 39] 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 that would seem natural to users and unlike [2], 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 disabled users as well as able-bodied users.

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.

EyePoint Implementation

The eye tracker constantly tracks the user's eye- movements[2]. A modified version of Salvucci's Dispersion Threshold Identification fixation detection algorithm [32] is used along with our own smoothing algorithm to help filter the gaze data. 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 close to screen boundaries to keep the magnified view fully visible on the screen.

The user then looks at the desired target in the magnified view and releases the hotkey. The user's gaze position is recorded when the hotkey is released. 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 uses a secondary gaze point in the magnified view to refine the location of the target.

The user has to 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.

Our pilot studies also showed that though some users would be looking at the target in the magnified view, the gaze data from their fixation was still noisy. We found that this occurred when the user was looking at the target as a whole (a gestalt view)

[2] If the eye tracker were fast enough, it would be possible to begin tracking only when the hotkey is pressed.

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 by focusing the user's gaze. In most cases, the focus points may be ignored by the user, however, they may be useful when the user wants to select a small target (Figure 5).

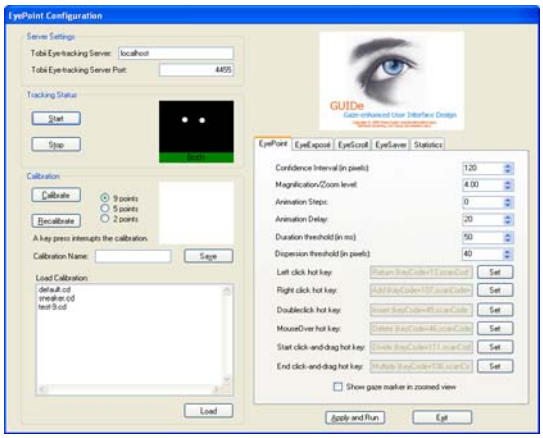


Figure 6. EyePoint Configuration Screen

Some users in our 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 an option (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.

We chose to use the keys on the numeric keypad of an extended keyboard as the default hotkeys for EyePoint (Figure 4 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 much bigger 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 6.

Disabled & Able-bodied Users

EyePoint is designed to work equally well for disabled users and able-bodied users. The hotkey-based triggering mechanism makes it simple 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 trackpad where touching different parts of the trackpad 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 based (breathe in to activate, breathe out to release) triggers. We hypothesize that these will be more effective than dwell-based activation, but have not performed tests. Dwell based activation is also possible in cases where the user does not have the ability to use any alternative approaches. In this case we would propose an approach similar to [21], but with off-screen targets to first select the action/mode, followed by dwell based activation (with audio feedback [23]) of the magnified view.

EVALUATION

We conducted 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 both a quantitative and qualitative evaluation. 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. Since the subjective user experience is also a key measure of the success and impacts adoption, our qualitative evaluation included the user's subjective feedback on using gaze-based pointing. Consistent with Norman's views in Emotional Design [27], 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.

Quantitative Evaluation

We tested speed and accuracy using three independent experiments: a) a real-world web browsing task b) a synthetic pointing only 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 EyePoint. Subjects practiced by clicking on links in a web browser and also performed 60 clicks in the EyePoint training application (Figure 7). Studies lasted a total of 1 hour and included one additional task reported in a separate paper [18]. The *spacebar* key was used as the trigger key for all three EyePoint variations. Animation of the magnified view was disabled as it would introduce an additional delay (user configurable, but generally about 60-100ms).

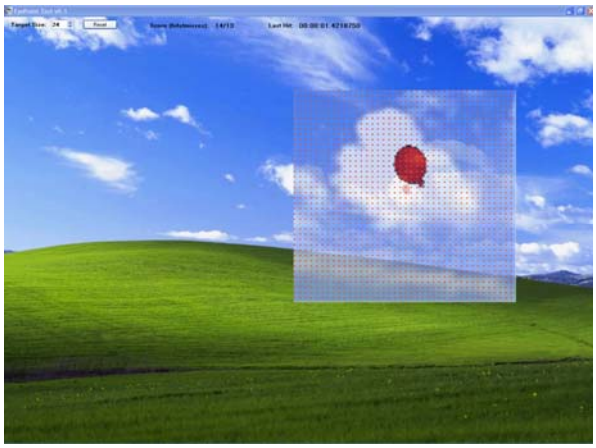


Figure 7. EyePoint training/test application (used for Balloon Study). This screenshot shows the magnified view with focus points.

Web Study

For a real-world 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 8).

Users were instructed to ignore the content of the page and simply click on the highlighted link. Each time they clicked on 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.

Figure 8. EyePoint real-world web-surfing task. The music link in the navigation column on the left has been highlighted in orange.

Balloon Study

For a synthetic task that tested raw pointing speed, we built a custom application that displayed a red balloon on the screen. The user's task was to click on the balloon. Each time the balloon was clicked, it moved to a new location (Figure 7). If the user clicked, but did not hit the balloon, 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 timing data on how long users took to perform the click. The size of the balloon was varied among 22px (the size of a toolbar button), 30px and 40px. The resulting study is a 4x3 within-subjects study (4 techniques, 3 sizes).

Mixed Study

We devised a mixed typing and pointing task in which subjects would have 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 9). 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 [8]). As soon as they had typed the correct word, the system would show the next balloon. The amount of time to correctly type the 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.

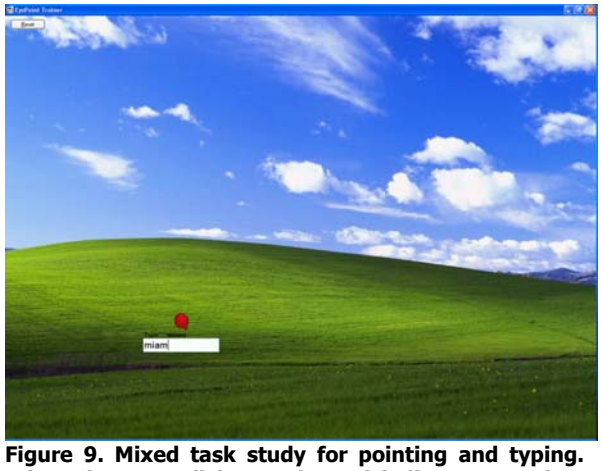


Figure 9. Mixed task study for pointing and typing. When the user clicks on the red balloon, a textbox appears below it. The user must type in the word shown above the textbox.

Qualitative Evaluation

For the qualitative evaluation, users were asked to fill out a questionnaire to provide their comments and

opinions on the interaction techniques. They were asked to evaluate gaze-based pointing and the mouse for speed, accuracy, ease of use and user 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.

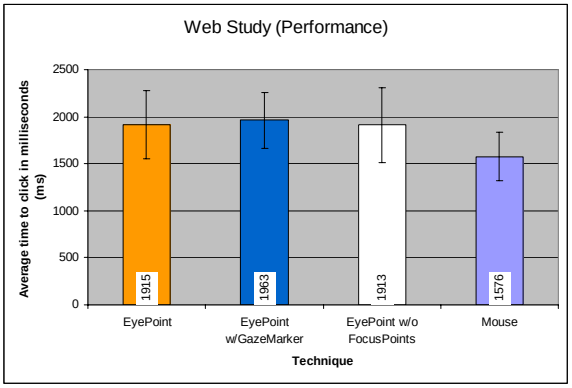


Figure 10. Web Study speed results.

Web Study Results

Figure 10 shows the performance results from the Web Study. A repeated measures ANOVA for technique showed that the performance 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. Performance 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 11 shows the accuracy results for the Web Study. As with the performance results, 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%. Qualitative 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

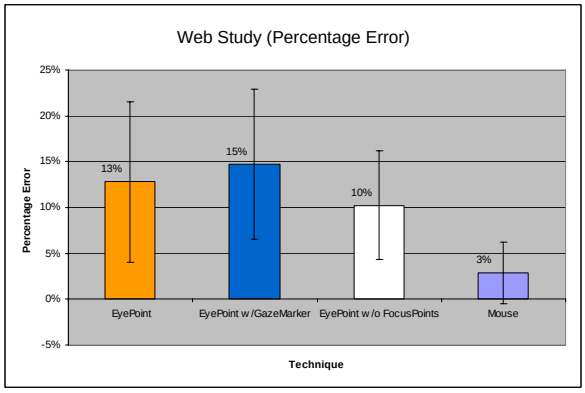


Figure 11. Web Study accuracy results

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.

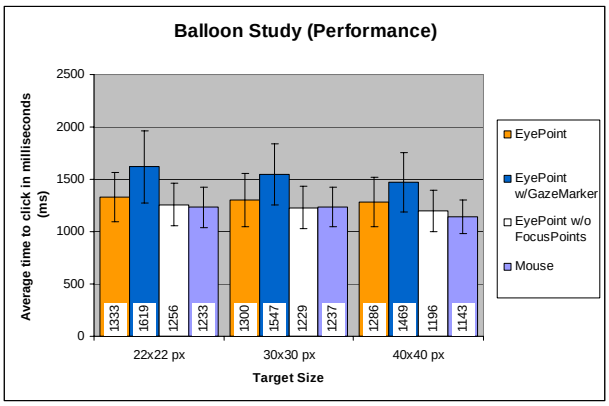


Figure 12. Balloon Study speed results.

Balloon Study Results

Figure 12 shows the performance results for the Balloon Study. EyePoint performs on average about 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 difference between all pairs of techniques except EyePoint with no focus points vs. mouse.

Figure 13 shows the error rates for the Balloon Study. In accordance with theory, 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.

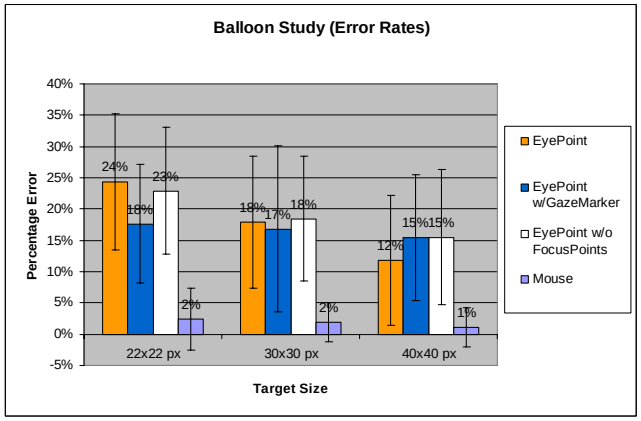


Figure 13. Balloon Study accuracy results.

Qualitative results for the balloon study showed that subjects found the mouse to be faster and more accurate. However, the

gaze-based techniques were 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.

Mixed Study Results

Figure 14 shows the performance results for the total round trip time to point to a target and return to the keyboard. 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 14 also shows the accuracy results from the Mixed Study. It should be noted that while the gaze-based technique had better performance, it had much lower accuracy than the mouse. A paired sample two-tailed *t*-Test showed that the error results are statistically significant with $p < .01$.

Qualitative 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.

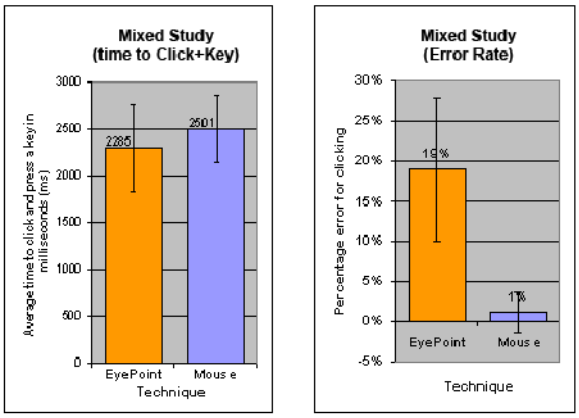


Figure 14. Mixed Study performance/error results.

Analysis and 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:

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

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

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

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 [6, 8]. 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 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 performance of EyePoint is similar to the performance of the mouse and can actually be faster than the mouse in some cases.

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 into subjects for whom the eye tracker worked well and subjects for whom the eye tracker didn't work as well, the error rates for the first group are closer to 10% while those for the second group are closer to about 33%.

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 the next section.

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 thin lenses work better than narrow frames and thick lenses; the former because the rim of the frame doesn't occlude the view of the camera and the latter since it reduces the visual distortion of the eyes.

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. 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 such intense usage.

While our studies randomized trials in order to compensate for learning effects, we did observe learning effects as subjects adapted to the system. For real-life usage one might therefore expect 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.

Subjects strongly preferred EyePoint over using the mouse. They felt that it was more natural since they were already looking at the target when they wanted to point. It 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.

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.

5. IMPROVING THE ACCURACY OF GAZE INPUT

In implementing EyePoint, we found that while the speed of a the gaze-based pointing technique was comparable to the mouse, the error rates were significantly higher. To address this problem, we have since conducted a series of studies to

investigate the source of these errors and identify ways to improve the accuracy of gaze-based pointing. In this section we present the two most important methods we found for improving the accuracy and user experience of gaze-based pointing: an algorithm for real-time saccade detection and fixation smoothing and an algorithm for improving eye-hand coordination. These methods boost the basic performance for using gaze information in interactive applications and in our application made the difference between prohibitively high error rates and practical usefulness of gaze-based interaction.

Saccade Detection and Fixation Smoothing

The nature of eye movements together with the limited accuracy of eye trackers, results in a noisy gaze signal. Prior work on algorithms for identifying fixations and saccades [26, 30, 32, 37] has dealt mainly with post-processing previously captured gaze information. For using gaze information as a form of input, however, 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. This is similar to the velocity threshold technique described in [32], 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 location rather than to the previous measurement. Second, we look ahead one measurement and reject movements over the saccade threshold, which 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 [34]) at saccade thresholds into the gaze data provided to the application.

The algorithm maintains two sets of points: the current fixation window and a potential fixation window. If a point is close to (within a saccade threshold of) the current fixation, 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, 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:

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

The size of the fixation window (n) is capped to include only data points that occurred within a dwell duration [22, 23] of 400-500ms (20 data points for our eye tracker). We do this to allow the fixation point to adjust more rapidly to slight drift in the gaze data.

Figure 15 shows the output from the smoothing algorithm for the x-coordinate of the eye-tracking data. We also show a Kalman filter applied to the entire raw gaze data and a Kalman filter applied within 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 within fixations only does yield comparable results to our one-sided triangular filter discussed above. It is possible that applying a non-linear variant of the Kalman filter [1], 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 is tailored to account for the different forms of eye movements.

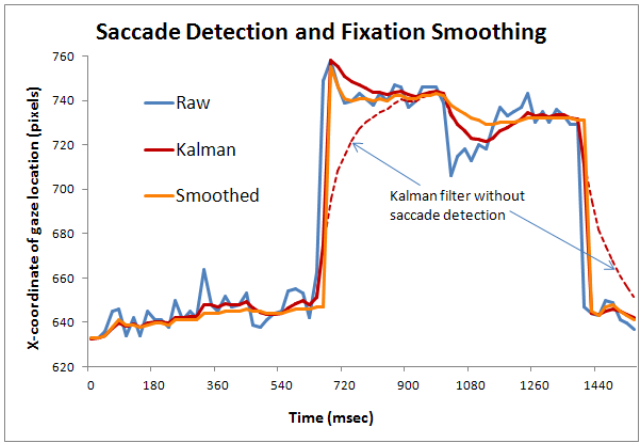


Figure 15. Results of our real-time saccade detection and smoothing algorithm. Note that the one measurement look-ahead prevents outliers in the raw gaze data from being mistaken for saccades, but introduces a 20ms latency on saccade thresholds.

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 significantly

improved the reliability of the results for applications which rely on the real-time use of eye-tracking data. A more detailed description of our algorithm including pseudocode is available in [15].

Eye-hand Coordination

Our research on using a combination of gaze and keyboard for performing a pointing task [19] showed that error rates were very high. Additional work on using gaze-based password entry [17] 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, first trying to perform the task as quickly as possible and second trying to perform the study as accurately as possible. The order of the studies was varied for counter-balancing and trials were repeated in case of an error.

An in-depth analysis of the data and classification of the errors from the two studies revealed multiple sources of error:

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 in the periphery of the screen results in lower accuracy from the eye tracker [3].

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

Late-Trigger errors: caused because the user 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.

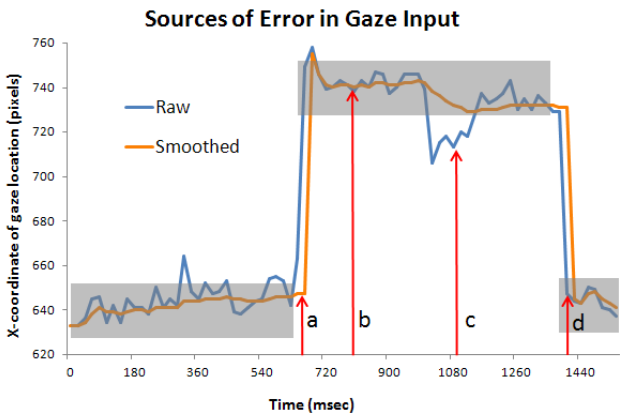


Figure 16. Sources of error in gaze input. Shaded areas show the target region. Example triggers are indicated by red arrows. The triggers shown are all different attempts to click on the upper target region. The trigger points correspond to: a) early trigger error, b) raw hit and smooth hit, c) raw miss and smooth hit, and d) late trigger error.

Figure 16 illustrates the different error types. Figure 17 shows how often each type of error occurred in the two studies.

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 sensor lag, smoothing latency and peripheral vision effects. We simulated this algorithm over the data from the Moving Target study. Figure 18 shows the outcome from the simulated results. It should be noted that both smoothing and early-trigger correction by themselves 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.

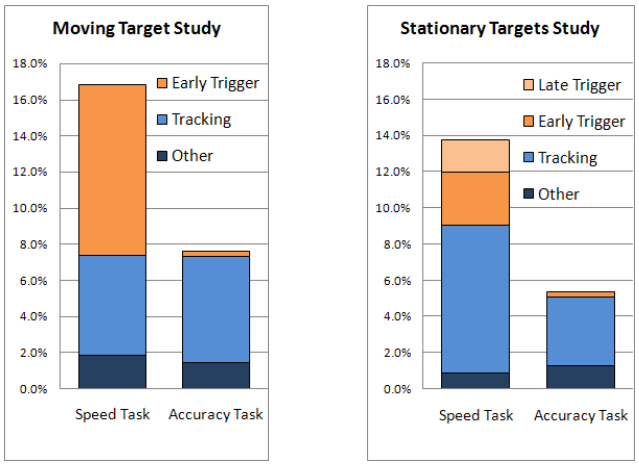


Figure 17. Analysis of errors in the two studies show that a large number of errors in the Speed Task happen due to early triggers and late triggers – errors in synchronization between the gaze and trigger events.

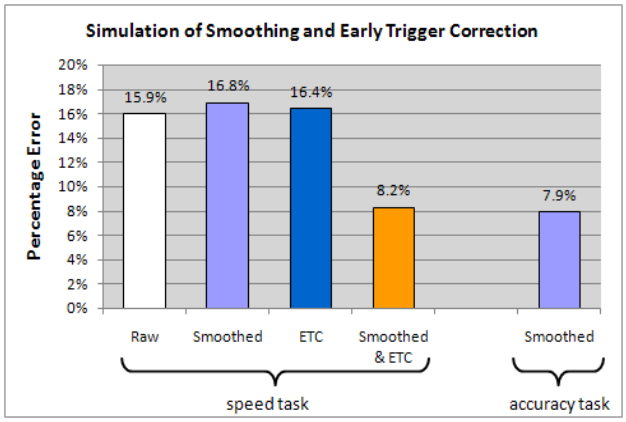


Figure 18. Simulation of smoothing and early trigger correction (ETC) on the speed task for the Moving Target Study shows that the percentage error of the speed task decreases significantly and is comparable to the error rate of the accuracy task.

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 information about the location of the targets. Since our approach has focused on providing a generally applicable technique for gaze-input which does 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.

6. SCROLLING

Scrolling is an inherent 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 [7, 12, 20, 35, 41] 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. However, the Midas Touch problem again becomes an issue when attempting to use gaze for scrolling. We posit that by understanding the characteristics of reading patterns and how users consume visual information [5, 28, 29] 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 that may satisfy different user preferences. We designed and implemented techniques for both manual and automatic scrolling. We also introduce the use of off-screen gaze-actuated buttons or hotspots that allow users to explicitly control document navigation.

Manual Scrolling

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, our contextual inquiry 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.

Gaze-enhanced Page Up / Page Down

We propose a new approach for a gaze-enhanced *Page Down* which uses a *GazeMarker* to always keep users' 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

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 19. 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.

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 19). 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.

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.

Explicit Activation/Deactivation

PC keyboards include a vestigial *Scroll Lock* key which the vast majority of users have never used. To overcome the Midas Touch problem we chose to use explicit activation of any of the automatic scrolling techniques by putting the *Scroll Lock* key back into use for turning the automatic scrolling on and off.

Estimation of Reading Speed

For several of the techniques presented, it is useful to be able to measure the user's reading speed. Previous work [5, 29] has shown that the typical eye movements (fixations and saccades) for a subject reading text conforms to Figure 20. Beymer et al. [5] 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. We present three scrolling techniques that

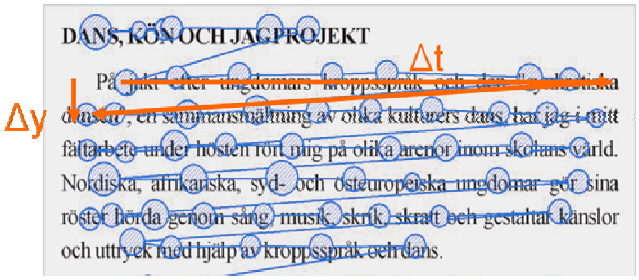


Figure 20. Estimation of reading speed. Vertical pixels viewed per second = Δy/Δt.

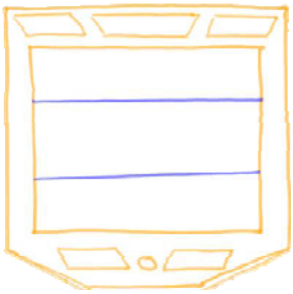


Figure 21. The eye-in-the-middle automatic scrolling technique adjusts the scrolling speed to match the user's reading speed and tries to keep the user's eyes in the middle thrid of the screen.

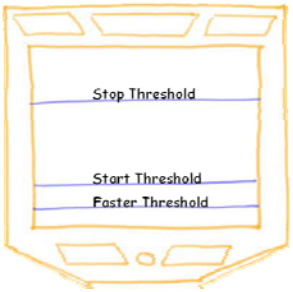


Figure 22. The smooth scrolling with gaze-repositioning technique allows for reading and scanning of content. Scrolling starts and stops depending on the position of the user's gaze with respect to invisible threshold lines on the screen.

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. Each of the techniques presented scrolls 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.

Eye-in-the-middle

The eye-in-the middle technique for automatic dynamically adjusts the rate of the scrolling to keep the user's gaze in the middle third of the screen (Figure 21). 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.

Smooth scrolling with gaze-repositioning

This automatic scrolling approach relies on using multiple invisible threshold lines on the screen (Figure 22). 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 in order 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 scrolling speed. The scrolling speed can be adjusted based on each individual's reading speed.

In our implementation, the positions of the threshold lines were determined by user feedback. In particular, placing the stop threshold line higher on the screen resulted in users 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 transitions from one state to another 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.

Discrete scrolling with gaze-repositioning

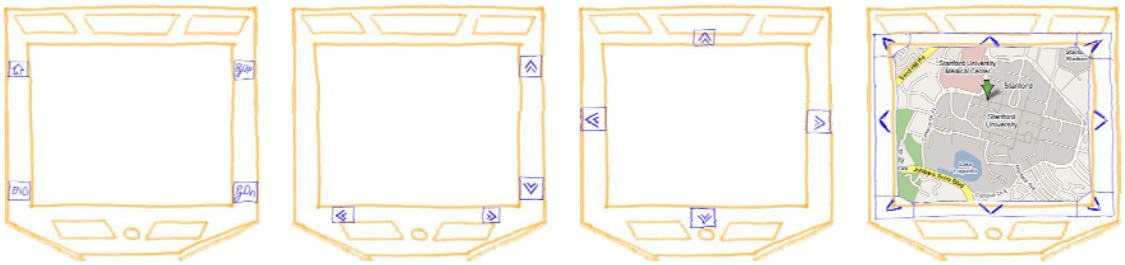
The discrete scrolling with gaze-repositioning approach leverages the gaze-enhanced *Page Up* / *Page Down* technique for manual scrolling and extends it by adding an invisible threshold line near the bottom of the screen. When the user's eyes fall below the threshold the system issues a page down command which results in the GazeMarker being drawn and the page being scrolled (Figure 23). The user's gaze must stay below the threshold for a micro-dwell duration (~150-200ms) before the scroll event triggers. This minimizes the number of false activations caused by 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.

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 23. 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.



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 24. 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.

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

Off-Screen Gaze-Actuated Buttons

The Tobii eye tracker provides sufficient field of view and resolution in order 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 various navigation controls. We implemented several variations of off-screen gaze-actuated buttons for document navigation as seen in Figure 24. 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.

Dwell vs. Micro-Dwell based activation

Document navigation actions have either a discrete nature (such as the *Home*, *End*, *Page Up* and *Page Down* buttons), which require one-time activation or a more continuous nature such as the cursor keys on the keyboard or the controls on a scroll bar. The latter typically require the action to be performed repeatedly.

In order to accommodate both of these actions we implemented two different activation techniques. The first, dwell-based activation, triggers only once when the user looks at the target for at least 400-500ms. For the actions which require continuous input, we chose to use a micro-dwell based activation in which the user looks at the target for at least 150-200ms.

Evaluation

Pilot studies with 10 users indicated that all subjects preferred the gaze-enhanced *Page Up*/*Page Down* technique over the normal *Page Up*/*Page Down*. In

studies with the automatic scrolling techniques, subjects felt that scrolling started when they expected it to, that the scrolling speed was neither too fast nor too slow and that they felt they were in control. Some subjects commented that they initially found it disconcerting to read while the text was moving, but that they got used to it with practice. It is conceivable that, like a teleprompter, subjects may become comfortable with reading moving text with practice. The discrete scrolling with gaze-repositioning technique overcomes the issues of reading moving text and was therefore the preferred approach for gaze-enhanced scrolling. Subjects indicated that they would prefer the gaze-enhanced scrolling techniques over traditional scrolling methods. Systematic studies will need to be done to rigorously evaluate the benefits and shortcomings of these techniques.

We have 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.

7. APPLICATION SWITCHING, PASSWORD ENTRY AND UTILITIES

Our gaze-enhanced application switching technique, EyeExposé [18], combines a full-screen two-dimensional thumbnail view of the open applications with gaze-based selection. We implemented EyeExposé on Microsoft Windows using a Tobii 1750 eye gaze tracker. Figure 25 show how EyeExposé works—to switch to a different application, the user presses and holds down a hotkey. EyeExposé responds by showing a scaled down 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.

The use of eye gaze instead of the mouse for pointing is a natural choice. Whether the user relies on eye gaze or the mouse, the visual search task to find the

Figure 25. 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.

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. Our user studies showed that subjects strongly preferred to use EyeExposé over other application switching techniques such as *Alt-Tab*, the Taskbar or even Exposé with mouse based selection.

Our gaze-based password entry technique [17] uses an on-screen keyboard with gaze-based typing (using either dwell or trigger based activation) to enter passwords and other sensitive information. This approach makes shoulder-surfing largely impractical. While it is easy to see what a user is typing on a keyboard, it is considerably harder to tell what they were looking at. User studies showed that gaze-based password entry requires marginal additional time over using a keyboard, error rates are similar to those of using a keyboard and subjects preferred the gaze-based password entry approach over traditional methods.

We have also developed several smaller utility applications, for instance, an application which activates the screen saver when the user looks away from the screen. When the user looks at the screen again, the screen saver is deactivated. This approach can be used to conserve power on laptop computers. Another application automatically warps the mouse from one screen to another in a multi-monitor setup.

8. CONCLUSION

We have discussed several novel interaction techniques which use gaze information as a form on input for the system. The key insights of our work are 1) to augment rather than replace existing interaction techniques, 2) to focus on the interaction design to mitigate the Midas Touch issues and 3) to improve the technology for interpreting and using eye gaze information as a form of input. We believe that by following these principles it is indeed possible to use gaze as an augmented input to help address the asymmetrical bandwidth problem [38] of contemporary computer systems.

Current technology and economic trends (sensor resolution and cost, processing power), the cost of eye tracking systems is expected to decline rapidly. Designing interaction techniques that incorporate gaze information as a primary input has the potential to radically change human-computer interaction as we know it today.

REFERENCES

1. 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.
2. Ashmore, M., A. T. Duchowski, and G. Shoemaker. Efficient Eye Pointing with a FishEye Lens. In Proceedings of *Graphics Interface*. pp. 203-10, 2005.
3. 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.
4. 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
5. 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.
6. 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.
7. 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.
8. 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.
9. Duchowski, A. T., *Eye Tracking Methodology: Theory and Practice*: Springer. 227 pp. 2003.
10. 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
11. 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.
12. 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.
13. 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.
14. 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.
15. 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
16. Kumar, M., *GUIDe: Gaze-enhanced User Interface Design*, 2006. Stanford. http://hci.stanford.edu/research/GUIDe
17. 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

18. 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
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.
24. McGuffin, M. and R. Balakrishnan. Acquisition of Expanding Targets. In Proceedings of *CHI*. Minneapolis, Minnesota, USA: ACM Press. pp. 57-64, 2002.
25. 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.
26. Monty, R. A., J. W. Senders, and D. F. Fisher, *Eye Movements and the Higher Psychological Functions*. Hillsdale, New Jersey, USA: Erlbaumpp. 1978.
27. Norman, D. A., *Emotional Design: Why we love (or hate) everyday things*. New York: Basic Books. 256 pp. 2004.
28. Poynter Institute and Eyetools, Inc., *Eyetrack III: Online News Consumer Behavior in the Age of Multimedia*, 2004. http://poynterextra.org/eyetrack2004/index.htm
29. Rayner, K. Eye Movments in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin* 124(3). pp. 372-422, 1998.
30. 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.
31. Salvucci, D. D. Intelligent Gaze-Added Interfaces. In Proceedings of *CHI*. The Hague, Amsterdam: ACM Press. pp. 273-80, 2000.
32. 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 Beach Gardens, Florida, USA: ACM Press. pp. 71-78, 2000.
33. Tobii Technology, AB, Drift Effects, in *User Manual: Tobii Eye Tracker and ClearView Analysis Software*. Tobii Technology AB. p. 15, 2006.
34. Tobii Technology, AB, *Tobii 1750 Eye Tracker*, 2006. Sweden. http://www.tobii.com
35. 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.

36. 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.
37. Yarbus, A. L., *Eye Movements and Vision*. New York: Plenum Presspp. 1967.
38. Zhai, S. What's in the Eyes for Attentive Input, *Communications of the ACM*, vol. 46(3): pp. 34-39, March, 2003.
39. 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.
40. 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.
41. 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.