A Robust Algorithm for Reading Detection

Christopher S. Campbell
IBM Almaden Research Center
650 Harry Rd
San Jose, CA 95120 USA
Tel: 1-408-927-1784
ccampbel@almaden.ibm.com

Paul P. Maglio
IBM Almaden Research Center
650 Harry Rd
San Jose, CA 95120 USA
Tel: 1-408-927-2857
pmaglio@almaden.ibm.com

ABSTRACT

As video cameras become cheaper and more pervasive, there is now increased opportunity for user interfaces to take advantage of user gaze data. Eye movements provide a powerful source of information that can be used to determine user intentions and interests. In this paper, we develop and test a method for recognizing when users are reading text based solely on eye-movement data. The experimental results show that our reading detection method is robust to noise, individual differences, and variations in text difficulty. Compared to a simple detection algorithm, our algorithm reliably, quickly, and accurately recognizes and tracks reading. Thus, we provide a means to capture normal user activity, enabling interfaces that incorporate more natural interactions of human and computer.

Keywords

Reading detection, gaze-based interfaces, user interest tracking, gaze tracking, attentive systems.

1. INTRODUCTION

With small low-cost cameras now filling our environment, improved sensing promises to increase the richness and speed of interaction with technology. A key question concerning this bandwidth increase is how an abundance of information will be used to improve the interaction experience. Good interactions take place when people accurately interpret behaviors and respond appropriately. If we want computers to provide satisfying and natural interactions, methods must be devised to infer user intentions and interests from user actions.

Given the highly visual nature of windows-based interfaces, gaze direction has been identified as an excellent source for determining user interest. Gaze movement data has led to two very different types of interfaces: command and non-command [2],[8]. Command-based interfaces use gaze location to directly issue commands to the system. For example, gaze can be used to type on a graphical keyboard [10], or select icons or menu items [3]. In contrast, non-command interfaces use gaze information to indirectly tune the system to the user's needs. An example of this is the storyteller system created by Starker and Bolt [11]. This system evaluates the level of interest in various objects and uses this information to determine which objects to talk about.

The typical desktop environment is heterogeneous, displaying a wide range of objects (windows, icons, menus, and text) in different sizes and colors. In this context, gaze patterns follow complex sequences of movements. A number of approaches have been developed to understand how these movements ought to be organized and modeled. Generally, these can be classified into three different levels of analysis: (a) highly detailed low-level micro-events, (b) low-level intentional events, and (c) coarse-level goal-based events.

Highly detailed low-level events typically include such movements as micro-saccades, jitter, nystagmus, and brief fixations, which are studied for their physiological and psychological relevance by vision scientists and psychologists. Low-level intentional events are the smallest coherent units of movement that the user is aware of during visual activity, which include sustained fixations and revisits (e.g., [2]). For example, users are aware of gazing at the cursor and then at an item on the menu bar, but they are not aware of briefly fixating on two other menu items before finding the desired one. Similar to both approaches is that of synthesizing higher-order intentions with low-level events. For example, once fixations and saccades can be reliably detected, then behaviors such as searching can be determined from a particular pattern of events. This approach assumes, however, that intentions are hierarchically organized such that global intentions are composed of a common set of specific intentions.

In contrast, goal-based analysis is concerned with the most general intentions of the user. In the typical windows interface, there are five primary goals to eye movements: (a) inspecting, (b) searching, (c) exploring, (d) performance monitoring, and (e) reading. *Inspecting* is simply looking at a relatively specific object to obtain more information about it. The object could be an icon, an image, or small animation on a web page. *Searching* is scanning around the desktop to find or visually acquire a known object. Searching is required when the object's location can not be determined from foveal or parafoveal vision. *Exploring*, in contrast, is the searching the desktop for some object that is interesting—looking for something to look at. *Performance monitoring* is looking at the cursor, keyboard, or other input/control object to ensure the expected actions are being performed. Typically, performance monitoring occurs while typing, dragging, or steering the cursor through menus. Finally, users may carefully *read* text, or they may quickly *skim* text [7].

Our goal is to develop and test a method for determining when the user is reading rather than searching or otherwise exploring the visual scene. As mentioned, our ultimate aim is to track text the user reads to infer user interests and goals. The paper is organized in four parts. First, we elaborate many of the advantages that can be gained by detecting reading. Second, we detail our algorithm for reading detection. Third, we describe two experiments that test our reading detection method. Finally, we discuss implications and future work..

2. UTILITY OF DETECTING READING

Because windows-based operating systems are ubiquitous and visually intensive, eye-gaze is a valuable way to determine user interest when interacting with computer displays. In many such gaze-based systems, interest in some display object (icon, image, or text) is determined by a fixation threshold. If the user looks at an object long enough, the system infers interest in that object (e.g., [11]). Reading detection provides a much more precise means for determining user interest because it can determine the level of user interest based on the type of user behavior, such as reading (high interest), skimming (medium), or scanning (low interest) as well as capturing the exact words on the screen involved. *Scanning* will be used to refer to the behaviors listed previously except reading (i.e., inspecting, searching, exploring, and monitoring).

We originally implemented our reading detection method to obtain information about what text the user is reading in order to infer user interests and then adapt information displays to user needs. The adaptation process includes recording the text of interest in a user model and using the text to find related information from local machine databases, local area network databases, and wide area network databases. An implemented scenario includes providing web pages related to text the user is currently reading, and then displaying the title of those pages on a scrolling ticker at the bottom of the screen [5].

Another benefit is that more accurate feedback results in more accurate models of the user. Thus, systems can provide relevant and personal assistance for a variety a tasks commonly performed with personal computers, such as searching for information on the web, writing manuscripts, composing e-mail, or looking for a certain type of news. For example, if a user's model shows that articles on Astronomy are always read, news gathering agents could use this knowledge to obtain articles that might be of interest and organize (prioritize) articles that have already been gathered.

Another specific advantage is that by using gaze movement data, computer help systems could be given more context information and therefore provide more accurate help. By analyzing reading data, we can determine which text was re-read, perhaps suggesting confusion, and which words were fixated on, perhaps because of a lack of familiarity. This data can then be used to decide what help topics to suggest and in what order.

Finally, knowledge of whether the user is reading, skimming, or scanning would be useful for creating adaptive peripheral displays [6]. When the user is reading, the display should be as quiet or as

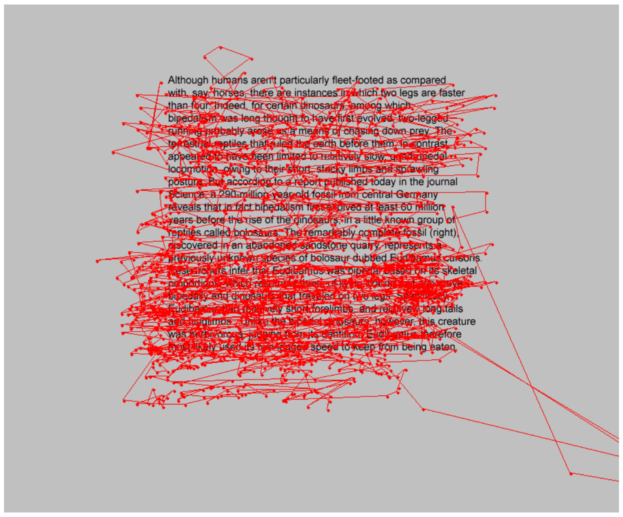
Although humans aren't particularly fleet-footed as compared with, say, horses, there are instances in which two legs are faster than four. Indeed, for certain dinosaurs, among which bipedalism was long thought to have first evolved, two-legged running probably arose as a means of chasing down prey. The terrestrial reptiles that ruled the earth before them, in contrast, appeared to have been limited to relatively slow, quadrupedal locomotion, owing to their short, stocky limbs and sprawling posture. But according to a report published today in the journal Science, a 290-million-year-old fossil from central Germany reveals that in fact bipedalism first evolved at least 60 million years before the rise of the dinosaurs, in a little known group of reptiles called bolosaurs. The remarkably complete fossil (right), discovered in an abandoned sandstone quarry, represents a previously unknown species of bolosaur dubbed Eudibamus cursoris. Researchers infer that Eudibamus was bipedal based on its skeletal [illegible] bipedally and dinosaurs that traveled on two legs. [illegible] Eudibamus had relatively short forelimbs, and relatively long tails and hindlimbs. Unlike the bipedal dinosaurs, however, this creature was herbivorous, judging from its dentition. Eudibamus therefore most likely used its [illegible] speed to keep from being eaten.

Figure 1. Example pattern of eye movements for Participant 1 reading a paragraph of text.

non-distracting as possible by reducing motion and eliminating auditory feedback. However, when the user is scanning, the display can be more assertive with its suggestions, for instance, becoming larger, flashing new information in red, or by making noises for stock market actions

3. SYSTEM AND IMPLEMENTATION

Detecting when a user is reading rather than merely scanning or skimming from eye-gaze patterns is a difficult problem, as low-level eye movements are almost completely automatic (i.e., involuntary). Common eye movement behaviors observed in reading include forward saccades (or jumps) of various length (eye-movements to the right), micro-saccades (small movements in various directions), fixations of various duration, and regressions (eye-movements to the left). These behaviors in turn depend on several factors, including text difficulty, word length, word frequency, font size and color, distortion, user distance to display, and individual differences (e.g., reading speed, intelligence, age, and language skills). For example, as the text becomes more difficult to comprehend, fixation duration increases [4], and the number of regressions increase as well [7]. Figure 1 shows an example eye-gaze pattern during reading.

Our reading detection system relies on three mechanisms: (a) coarse or quantized represented of eye-movements, (b) pooled evidence based detection, and (c) mode switching. First, the eye-movements in both x and y positions are quantized (averaged) over 100 ms intervals. This process removes some of the inaccuracy of current eye-tracking hardware and reduces the influence of micro-saccades. Second, evidence of reading is accumulated until it crosses a threshold value. This is done by incrementing a reading-evidence variable when the eye moves to the right and de-incrementing when the eye moves to the left. If the evidence reaches a threshold, then "reading" is detected and the mode is switched to reading from scanning.

Pooled evidence acts to reduce the influence of eye movements back to previously read words (regressions or revisits) and movements above and below the current line of text. When the evidence threshold is reached, reading is detected and the system switches from scanning to reading mode. Mode switching allows us to essentially interpret the same eye movements differently based on changes in context. For example, large eye movements to the left and slightly up mean within a scanning context that the user is continuing to scan but within a reading context, this movement is more likely to mean that the user is re-reading text and will continue the reading process. Depending on the difficulty of the text, a user may often revisit text encountered several sentences before to clarify ambiguities in the sentence currently being read. If this movement were only allowed to have only one meaning, say that the user is scanning, then the tracking of reading would end prematurely on every revisit. If this movement were to only mean that the user is reading, then this would increase the number of false alarms or times the system detected reading when the user was not reading. Mode switching allows us to account for this behavior in different contexts, and as a result, to produce more robust reading detection as well as continuous, reliable read tracking.

More precisely, the system first quantizes raw data sent from the eye tracking hardware by averaging every 3 data points. This raw data is provided by the eye tracker at a rate of 60 points (x and y positions) per second, but after averaging is reduced to 20 data points per second or one data point every 50 ms. The system is initially in scanning mode, which requires a set of events to occur to switch into reading mode. The events that are tracked include the specific eye movements shown in Table 1. For example, if the eye moves a short distance left then the event is a "regression saccade" but if the eye moves a long distance left then the event is a "scan jump". The distinction among "short", "medium" and "long" distances used to characterize events (in Table 1) reflect a set of adjustable parameters, one for each distance and each direction pair.

The quantized, tokenized stream of eye-movement data is then pooled to determine whether the user is reading. The pooled evidence for reading is calculated by taking the accumulated value of the pooled data and adding the points associated with the current event for both the X and Y axes. Thus, if a "read forward" event occurs for the X axis and a "skim jump" occurs for the Y axis then (10 + -5) = 5 points would be added to the pool. Note that it is possible to have no event occur for the x or y axis if the eye does not move. Every non-event is given 0 points. For this implementation, the pooled evidence that a user is reading must cross a threshold of 30 to switch into reading mode.

Using pooled evidence, the system does not have to look for a specific pattern of events but allows for a wide range of patterns to signal reading. Thus, reading recognition is tolerant to noise, maintains a high hit rate and low false alarm rate. For example, the events "read forward", "skim forward", "skim jump", "read forward", and "read forward" (10 + 5 + -5 + 10 + 10 = 30 points) are sufficient to trigger reading detection. However, these five events may be ordered in different ways -- there are exactly 20 possible permutations. Rather than looking for each of these 20 possible sets of events, pooled evidence allows the system to accumulate mounting evidence despite noise. Thus, increasing noise only delays reading detection but does not block it altogether. Ideally, the fastest period in which reading could be detected occurs if the highly unlikely pattern of three "read forward" events appear in a row. Because we sample in 100 millisecond increments, 3 x 100 = 300 milliseconds or about one-third of a second is the fastest possible reading detection time.

Once the threshold is passed, reading is detected and the mode changes from "scanning" to "reading" mode. In reading mode, the rules for changing back to scanning mode are different. The system records every word read in reading mode until a "scan jump" event is detected. A single "scan jump" event will send the system back into scanning mode. This method of mode switching allows for fairly quick changes in modes while still maintaining reliable read tracking. Reliable read tracking is important because readers will often show a wide range of behaviors while reading, including long pauses on ambiguous words, large regressions to text that may help to disambiguate the current sentence, and moderate forward jumps in anticipation of up-coming text.

Table 1. Tokenization of eye movements and evidence values for reading.

Distance, direction, axis	**Token**	**Points**
Short right X	Read forward	10
Medium right X	Skim forward	5
Long right X	Scan jump	Reset
Short left X	Regression saccade	-10
Medium left X	Skim jump	-5
Long left X	Scan jump	Reset
Short up Y	Skim jump	-5
Medium up Y	Scan jump	Reset
Long up Y	Scan jump	Reset
Short down Y	Anticipatory saccade	0
Medium down Y	Skim jump	-5
Long down Y	Scan jump	Reset
Long, medium left X and short down Y	Reset jump	5

Note: Positive point values indicate evidence supporting reading and negative number indicate evidence against reading.

Previous research has been concerned more specifically with making sense out of complex, low level eye movement data. The eye is constantly moving. Even when one seems to be looking steadily at some object, the eye still makes micro-saccades (small movements), jitters (shaky movements), and nystagmus (compensatory movements to head motion). To provide eye movement data that is closer to what users experience, researchers have attempted to break down or filter complex raw eye movement data into a set of tokens. Jacob's [3][2] work on fixation recognition has formed the core of this research area. The term "fixation" refers to an area of relatively stable gaze that lasts between 30 and 800 milliseconds. Although people are not aware of micro-saccades, they do report areas of fixation. Thus, fixation recognition is an attempt to determine where a user *intended* to look. Jacob's fixation recognition algorithm works by taking a 100 millisecond set of data (6 data points for this implementation) and if the points are all within .5 degrees of visual angle, then a fixation is said to be detected and located at the average point. The fixation continues as long as the gaze points stay within 1.0 degree of this average fixation point.

Obviously, the goal of Jacob’s method differs from our goal of recognizing reading. To compare Jacob's method with our own, let us assume that his method for fixation recognition is used by a simple algorithm for reading detection. For instance, suppose a series of say three fixations to the right signal that reading is detected. However, several problems occur when using this method for reading detection: (a) loss of information, (b) regressions, (c) eye movement on the Y axis, (d) resets to beginning of next line, (e) revisits to previous sentences.

We believe our method is an improvement over this extended algorithm because it: (a) does not throw out any raw eye movement information and also uses partial information by giving fuzzy evidence points for tokens, (b) can ignore regressions using pooled evidence, (c) takes into account eye movements on the Y axis with respect to reading, (d) tokenizes resets as evidence to detect reading, (e) uses mode-switching to track ongoing reading by ignoring revisits.

4. EVALUATION

In what follows, we compare our algorithm with the extended Jacob’s algorithm to determine which performs faster and more reliably under actual reading conditions. More precisely, two experiments investigated recognition accuracy (Experiment 1) and recognition speed (Experiment 2) for the two competing algorithms. Eye movements can be classified as either reading [7] or goal directed searching. We believe that the gaze patterns associated with each are sufficiently different such that they can be distinguished, that is, reading is detected when the user is reading and not when the user is searching. The two algorithms evaluated were (a) the pooled evidence method and (b) the simple method based on Jacob's fixation recognition method. It was hypothesized that reading recognition would differ across participants, but that the pooled evidence method would be systematically faster and more accurate than the competing method.

4.1 EXPERIMENT 1

The first experiment was performed to test the reading detection accuracy of two competing algorithms.

4.1.1 Method

Participants were instructed to perform two tasks: (a) quickly search for a target icon on a screen full of distracter icons, and (b) carefully read a text passage and then answer a multiple-choice question about the passage. These two tasks were presented in random order, and eye movements were recorded.

4.1.1.1 Participants

Four participants were recruited from IBM Almaden Research Center. All had normal or corrected to normal vision.

4.1.1.2 Design and Materials

We used a single factor (task type) within-subjects design with two levels, searching and reading. Each task was presented 20 times for a total of 40 trials presented in random order.

In the search task, participants were presented with a matrix of standard Windows-style icons that had been reduced in size from 1280 x 1024 pixels to 640 x 480 pixels. The search target for every search trial was the Windows Excel Document icon, which varied in position from trial to trial. Participant's used the mouse pointer to select the target icon.

In the reading task, participants were presented with a paragraph of text that filled an area of about 400 x 400 pixels. The text was taken from popular press science magazine articles and averaged 250 words each. The text was presented in 15 point font with a black foreground and a gray background (see Figure 1). After reading the text, comprehension was tested with a four alternative multiple-choice question.

4.1.1.3 Equipment

Participants viewed a 20-inch monitor with a resolution of 1280 x 1024 pixels. The reading passages were shown in a window 400 x 400 pixels, subtending a visual angle of 15.6 horizontal and 15.6 vertical degrees at a distance of 0.43 meters. The search matrices were presented in a window 640 x 480 pixels, subtending a visual angle of 24.1 horizontal and 18.5 vertical degrees at a distance of 0.43 meters.

Gaze direction was obtained by a video camera fitted with an array of LED's that project infrared light to the participant's eye, resulting in a reflectance point on the cornea and the illumination of the pupil. Given the reflectance point and the center point of the pupil, along with a short calibration session, the location of the eye-gaze could be easily calculated to within about 2.5 degrees of visual angle. Images were obtained from the camera through the serial port and processed by a custom software application. The application output one gaze position (x and y screen coordinates) once every 16.666ms (60Hz).

4.1.1.4 Procedure.

Participants were calibrated on the eye tracking hardware to ensure accurate tracking over the area of the screen. A chin rest was used to stabilize the participant’s head during calibration and during the experiment, as the camera was not equipped with servos to adjust to changes in head position. The calibration process lasted about 2 minutes.

Participants then went through a training session in which they performed one trial each of the searching and reading tasks. The

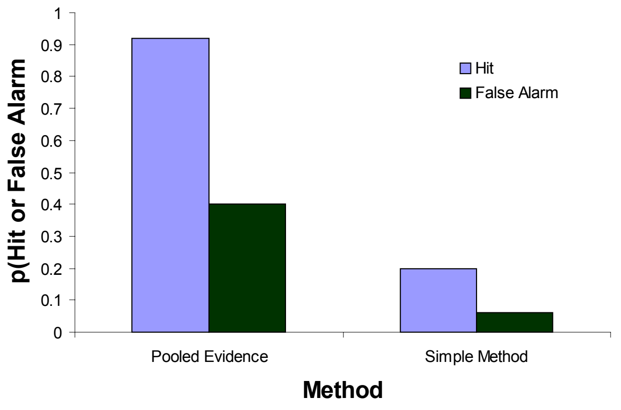

Figure 2. Proportion of hits and false alarms for competing algorithms.

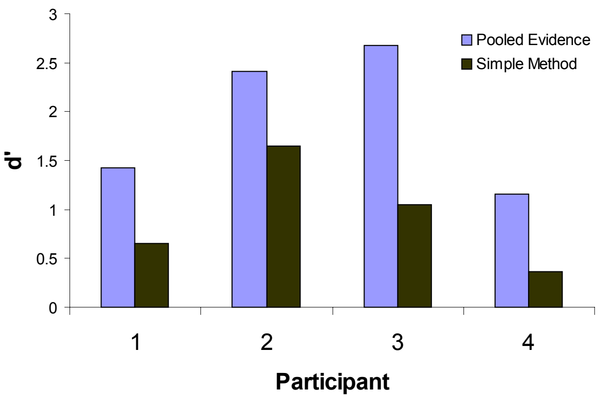

Figure 3. Sensitivity of competing algorithms for all participants.

experimenter monitored participants in the training session and was available for any questions or concerns.

Participants were presented with each task in random order without replacement. If participants failed to click the correct icon or failed to answer the multiple-choice question correctly, they were required to do the trial over. This was instituted to motivate the participants to perform the task carefully and with greater accuracy than speed. After each trial, participants could take a break from the experiment for as long as needed to prevent fatigue.

4.1.2 Results

The raw gaze data were analyzed with both our pooled evidence algorithm and the simple algorithm based on Jacob's fixation recognition method.

For the pooled evidence algorithm, data were first quantized and then tokenized according to the rules in Table 1. The number of tokens was counted for each condition and subject. Finally, the tokenized data were processed for reading with an evidence threshold of 30 points. After the threshold was exceeded (reading detection), reading was tracked until a Scan jump token was found. Finally, the rectangular area on the screen in which reading was detected was calculated with the upper left-hand corner as the first point of detection and the lower right-hand corner as the last point.

For the simple algorithm, the data were processed for fixations by grouping gaze points that occur within 1.0 degree of visual angle (about 14 pixels) and averaging over each group. Fixations were then processed for short movements in the vertical (x) direction to the right. A counter was incremented by one when this movement was observed and set back to zero when a different movement was encountered. When the counter reached a threshold value of 3, reading was detected. That is, reading was detected when 3 successive movements to the right were detected. The threshold of three movements was chosen because a pilot study that set the threshold to four successive movements failed to detect reading on any trials. The area of text read was simply all of the fixations to the right after initial detection. Similar to the other algorithm, the area was calculated as a rectangle based on the first and last reading points.

Figure 2 shows the accuracy of each method as indicated by hits and false alarms. Hits are the proportion of reading trials in which reading was detected. False alarms are the proportion of searching trials in which reading was detected. Over all four participants, the hit rate is high for the pooled evidence method and the false alarm rate is much lower. For the simple method, both hits and false alarms are low.

Though the difference in hits and false alarms seems to favor the pooled evidence method, decision bias could also account for the result --- that is, the pooled evidence algorithm had a more liberal bias than the simple algorithm. Thus, d' was calculated as a bias-free measure of reading detection. The d' measure is the z-score difference between the proportion of hits and the proportion of false alarms--- Z(hits) - Z(false alarms). Figure 3 shows that d' for each participant is significantly higher (i.e., better) for the pooled evidence method than the simple method (Friedman Test Statistic = 4.00, $p < .05$). Thus, the pooled evidence method is more sensitive to reading than the simple method.

4.2 EXPERIMENT 2

Given that the pooled evidence algorithm robustly detects reading versus searching, we now turn our attention to whether the difficulty of the text affects the algorithm and its detection speed.

4.2.1 Method

Participants were presented with a series of passages (each containing about four sentences) that varied in reading difficulty. Participants were instructed to read the text carefully in order to correctly answer a multiple-choice question about the content of the passage. The trial had was repeated until the correct answer was given.

4.2.1.1 Participants

Five participants were recruited from the IBM Almaden Research Center. All had normal or corrected to normal vision.

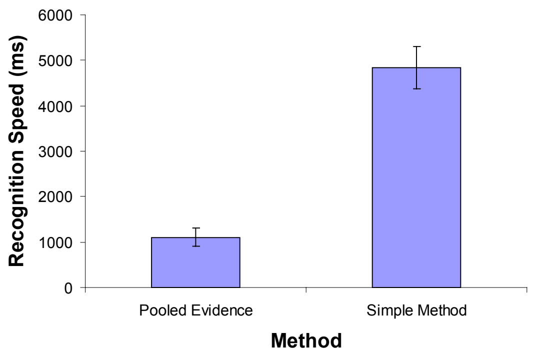

Figure 4. Recognition speed of the two competing algorithms over all participants.

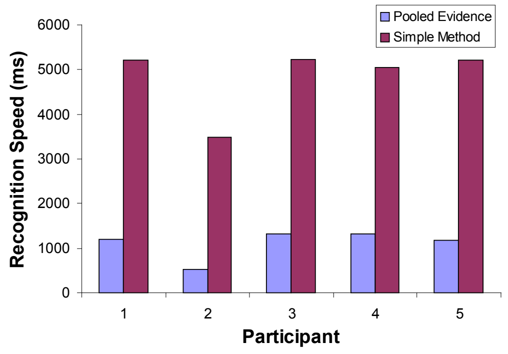

Figure 5. Recognition speed of the two competing algorithms grouped by participant.

4.2.1.2 Design and Materials

We used a single factor (text difficulty) within-subjects design with two conditions (easy or difficult) and fifteen trials per condition (30 trials total). Eye movements were measured the same way as in Experiment 1.

Participants were presented with two types of text, easy and difficult. The easy passages were taken from children's stories with an average Flesch-Kincaid grade level (see [1]) of 4.2 and an average Flesch reading ease score of 79. Reading ease scores range from 0 to 100 with 100 being the easiest. The difficult passages were taken from technical journal articles. The difficult passages contained technical terms, as well as noticeably longer and more complex sentences. The average Flesch-Kincaid grade level of the difficult passages was 12, with a reading ease of 34. Both easy and difficult passages contained the 65 words, but the difficult passages tended to be longer because they contained longer words. The font was 10 point Times Roman, presented in black on a white background.

Participants were required to answer a four-alternative multiple-choice question after each passage. The question was created to focus on the meaning of the passage rather than on the surface form. Thus, participants would not be expected to answer the question correctly if they had merely skimmed the text.

4.2.1.3 Procedure and Equipment

Equipment was the same as in Experiment 1. Before beginning the experiment, participants were instructed to read each passage as carefully as possible and to click the Next button at the bottom of the window when finished. Participants were told that they had to answer a multiple-choice question correctly before proceeding to the next passage.

4.2.2 Results

The pooled evidence algorithm and simple algorithm were compared as in Experiment 1. Our measure of performance, speed, was measured as the first point at which reading was detected.

Consistent with previous work (e.g., [9]), regressions occur significantly more often for difficult text than for easy text, $t(4) = 2.36$, $p = 0.039$. However, there are no significant differences for text difficulty for the other tokens. As anticipated, there were few scan jump tokens in relation to read forward and skim forward tokens. The large eye movements of scan jumps did not occur with high frequency during careful reading. Finally, we see very few reset jumps compared to the other tokens as these events only occurred when the reader reaches the end of a line. Because few lines of text were in each passage, we expected a low frequency of reset jumps.

Figure 4 shows that the pooled evidence algorithm recognized reading much faster than the simple algorithm, $t(4) = 16.31$, $p < 0.05$. Recognition speed of the pooled evidence algorithm ranged from 200 to 3000ms, with an average of just over one second (1106ms). The simple algorithm, however, ranged from 200 to 15000ms with an average of 4840ms or about five seconds. Comparing the percentage of the time the simple algorithm detected reading faster than our pooled evidence algorithm (wins analysis), we find that the simple algorithm was faster only 2 times out of 150 trials (30 x 5 participants) or about 1% of the time. Thus, pooled evidence won 99% of the time.

Figure 5 indicates that the performance of each algorithm was fairly consistent across the different participants. The reading of participant 2 seems to be recognized more quickly for both algorithms, possibly because of better reading skill as there were also fewer regressions and other eye movements that often indicate comprehension difficulties.

5. GENERAL DISCUSSION

Overall, results show that the pooled evidence algorithm (a) is about four seconds faster at detecting reading than the simple method, (b) has a high (nearly 100%) accuracy rate, (c) is reliable across different participants and styles of text. Fast reading detection is critical for interactions that are both timely and responsive. We believe that even the mean recognition time of 1 second given by our algorithm is somewhat slow for fast and natural interaction. Though our algorithm is reliable across participants, we believe a more diverse population should be tested to ensure that reading can be detected even in individuals with moderate to low language skills.

The results also show that text difficulty did not influence gaze patterns as much as expected -- only the number of regressions varied with difficulty. One possible reason for this is that the children's story may have been more difficult to read than given by the Flesch reading ease index. This is because of the highly descriptive nature of the text using many low-frequency words.

Another factor could be that the participants all were of very high reading ability and thus, had little trouble comprehending even the technical journal passages.

Our goal was to create a system that recognizes reading from gaze patterns that occur during normal interaction with a windows-based user interface. Results from other research and the present work have shown that gaze patterns during reading are more complex than might be assumed. Testing a model that assumes a simple reading pattern demonstrated that this method is not sufficient for timely reading recognition. In contrast, our pooled evidence algorithm can reliably, quickly, and accurately recognize and track reading performance.

6. FUTURE WORK

We intend to enhance our reading detection algorithm in several ways. First, in addition to detecting reading, with a few modifications, the algorithm can also detect skimming. The method for detecting skimming includes recording, in reading mode only, the distance of each eye-movement. If the distance is less than some threshold, the words that the eye moved across are classified as "read" but if the distance is greater than some threshold then the words are classified as "skimmed". In other words, if the eye moves quickly over some words then those words were skimmed.

A second method to enhancing our algorithm would be to actively adapt its parameters. We will include parameters that adapt to individual reading speeds and abilities by adjusting parameters that are used to determine the actual values of the distances we called "short", "medium", and "long" in Table 1. If, for example, the system determines that the user is a slow and careful reader, all the distances (for the x axis) should be decreased to optimize performance. If, however, the system determines that the user's reading ability is poor, more regressions will occur and the mode switching threshold should be decreased (to be more sensitive).

A third way to improve the algorithm is to take account of context information to constrain reading detection and improve accuracy and reliability. For instance, it would be useful to know (a) the location of text on the screen, (b) the size of the font, (c) the content of the text on the screen, (d) whether the user is scrolling, navigating, or pointing, and (e) the distance of the user from the screen. Mode switching between reading and scanning could be improved by knowing if the user is looking at a block of text on the screen. Reading detection could also be improved by knowing the size of the text on the retina, as this determines the size of eye movements: the larger the text, the bigger the eye movements in reading. Reading detection would also be improved by knowing the content of the text on the screen. Finally, knowing whether the user is manipulating an input device may also help to determine whether the user is reading, as it is unlikely that the user is reading when navigating, pointing, or scrolling.

7. ACKNOWLEDGMENTS

Thanks to Myron Flickner and Dave Koons for building the eye-tracking system we used. Thanks to Teenie Matlock for comments on a draft.

8. REFERENCES

[1] Flesch, R. (1948). A new readability yardstick. Journal of Applied Psychology, 32, 221-233.

[2] Jacob, R. J. K. (1993). Eye movement-based human-computer interaction techniques: Toward non-command interfaces. In Hartson, D. & Hix, (Eds.)., Advances in Human-Computer Interaction, Vol 4, pp. 151-180. Ablex: Norwood, NJ.

[3] Jacob, R. J. K. (1990). What you look at is what you get: Eye movement-based interaction techniques. Proceedings ACM CHI'90 Human Factors in Computing Systems, pp 11-18.

[4] Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. Psychological Review, 87, 329-354.

[5] Maglio, P. P., Barrett, R., Campbell, C. S., & Selker, T. (1999). Suitor: An attentive user interface. In Proceedings of the International Conference on Intelligent User Interfaces 2000. New York: ACM Press..

[6] Maglio, P. P., & Campbell, C. S. (1999). Tradeoffs in displaying peripheral information. In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2000). New York: ACM Press.

[7] McConkie, G. W. (1983). Eye movements and perception during reading, in K. Rayner (Ed.). Eye movements in reading, Academic Press: NY, pp 65-96.

[8] Nielsen, J. (1993). Noncommand user interfaces, Communications of the ACM, 36(4), pp. 83-99.

[9] Reichle, E. D., Alexander, P., Fisher, D. L., & Rayner, K. (1998). Toward a model of eye movement control in reading. Psychological Review, 105, 125-157.

[10] Salvucci, D. (1999). Inferring intent in eye-based interfaces: Tracing eye movements with process models, Proceedings ACM CHI'99 Human-Factors in Computing Systems, pp 254-261.

[11] Starker, I., & Bolt, R. A. (1990). A Gaze-responsive self-disclosing display, Proceedings ACM CHI'90 Human-Factors in Computing Systems, pp 3-9.