Eye Typing Using Word and Letter Prediction and a Fixation Algorithm

I. Scott MacKenzie & Xuang Zhang

Dept. of Computer Science and Engineering
York University
Toronto, CANADA
{mack, xuang}@cse.yorku.ca

Abstract

Two eye typing techniques and a fixation algorithm are described. Similar to word prediction, letter prediction chooses three highly probable next letters and highlights them on an on-screen keyboard. Letter prediction proved promising, as it was as good as word prediction, and in some cases better. The fixation algorithm chooses which button to select for eye-over highlighting. It often chooses the desired button even if another button is closer to the fixation location. Error rates were reduced when using the fixation algorithm combined with letter prediction; however, the algorithm was sensitive to the correctness of the first several letters in a word.

ACM Classification: H.5.2 [Information Interfaces and Presentation]: User Interfaces

Keywords: Eye tracking, eye typing, word prediction, letter prediction, fixation algorithm, button size

1 Introduction

Humans obtain considerable information through their eyes. For example, we attend to many on-screen regions and targets while interacting with graphical user interfaces (GUIs). Compared with a mouse, eye movement is fast. It consists of fixations (pausing to acquire information) and saccades (rapid jumps to another location). Fixations can be long or as short as 200 ms [Collins and Blackwell 1974], while saccades are inherently quick, taking about 30-120 ms [Majaranta and Räihä 2002].

Using the eye, selection techniques include dwelling on a target or separately pressing a hardware button while looking at a target. Dwelling too long hinders performance as users become impatient while waiting for selection; too short and inadvertent selection occurs. For example, with a brief dwell time of around 200 ms, selection may occur even when the user fixates on the object only to obtain information. Dwell times typically range from as low as 400 ms [Ware and Mikaelian 1987] to over 1000 ms [Špakov and Miniotas 2004].

2 Eye Typing

Research on entering text via the eye – eye typing – extends over 25 or so years, as reviewed in several sources [Collins and Blackwell 1974; Hutchinson et al. 1989; Lankford 2000; Majaranta and Räihä 2002; Salvucci 1999]. However, none of the techniques reviewed uses next-letter prediction and highlighting combined with a fixation algorithm to improve eye typing speed and accuracy.

Copyright © 2008 by the Association for Computing Machinery, Inc.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.
ETRA 2008, Savannah, Georgia, March 26–28, 2008.
© 2008 ACM 978-1-59593-982-1/08/0003 $5.00

We now describe and rationalize our fixation algorithm and implementation, and follow with tentative late breaking results of an experiment to test our algorithm and interaction technique.

2.1 Fixation algorithm

Our fixation algorithm determines which button on the keyboard receives "eye-over" highlighting. It has several components, including the keyboard geometry, the current fixation location, knowledge of previous characters, and a language model (a word-frequency list). The algorithm works such that the button with eye-over highlighting is not necessarily the button closest to the measured fixation location. This is described as follows.

First, we define *tolerantDrift*, a radius distance from the fixation location. For each button having its center point within tolerantDrift, the algorithm builds a word stem, given the letters already entered. For each word stem, a weighting factor is computed and the button associated with the highest weighting appears with eye-over highlighting (gray). Four weighting methods were considered:

0 None (always choose the closest button)
1 Weighting = word stem frequency divided by the squared distance to the center of the button
2 Weighting = word stem frequency divided by the distance to the center of the button
3 Weighting = word stem frequency

The algorithm may err if fixation is near a high probability letter. For example, it would often take "a" instead of "z". To prevent this, a fixation within half the radius of the center of a button is always considered on that button.

Figure 1 gives an example. After inputting "th", the user wants to fixate on "e"; however, the computed fixation has drifted to the edge of "d". (This is shown as a crosshair to illustrate the algorithm. Neither the crosshair nor the tolerantDrift circle is visible to the user.) The algorithm considers weightings for word stems "thd", "thr", "ths", and "the". "the" wins. The eye-over highlighting is on "E" instead of the "D". In other words, the user thinks he/she is fixating on the correct letter.

We developed a model to calculate error rates given the button size, the fixation drift, the tolerantDrift, the weighting method, and the number of previously entered characters considered. Our model assumes English text input on a keyboard with a QWERTY layout. Button diameters of 76 and 100 pixels were considered along with a fixation drift of 40, 50, 60, 70, 80 or 90 pixels from the center of a desired button, and tolerantDrift of 55, 60, 65, 70, 75, 80, 85, and 90 pixels.

Figure 2 shows the model's output for small buttons, considering 0 previously entered letters. The lowest error rate was 35.5%, coincident with tolerantDrift = 70 pixels and weighting method 1, where the weighting was the word stem frequency divided by

the squared distance to the center of the button. The results were similar for large buttons.

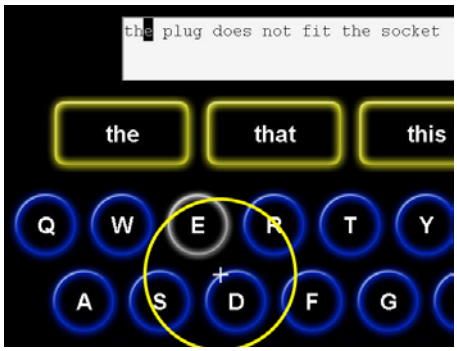


Figure 1. Fixation algorithm (see text for discussion)

Of course, the error rate drops as more previously entered – and presumably correct – letters are considered. This is shown in Figure 3. The left of the figure uses weighting method 0 (i.e., the algorithm is not used). Each bar shows the average of the error rates over the range of fixation drifts noted above, assuming such drift occurs after 0, 1, 2, or 3 letters of correct input at the beginning of a word. The right of the figure uses weighting method 1 with tolerantDrift = 70 pixels. This time, however, where fixation drift occurs, the fixation algorithm kicks in to help choose the correct button for eye-over highlighting. The improvement is clearly seen. After correctly entering one letter, the predicted error rate dropped to below 15%. The error rate was only 4% after three letters of input. This is a promising result.

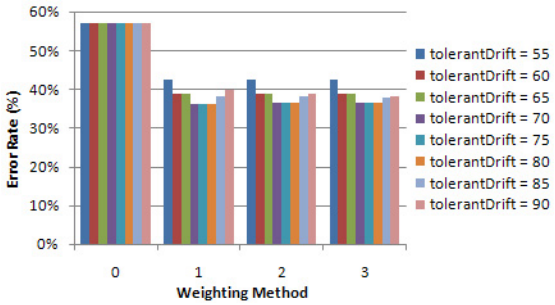


Figure 2. Predicted error rates vs. weighting method and tolerantDrift radius in pixels.

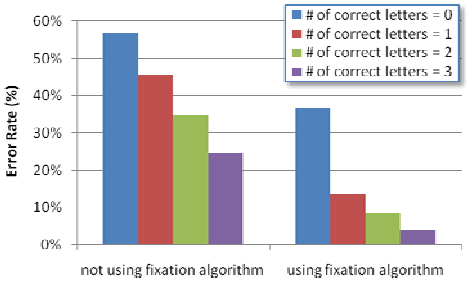


Figure 3. Predicted error rates vs. algorithm settings for small buttons

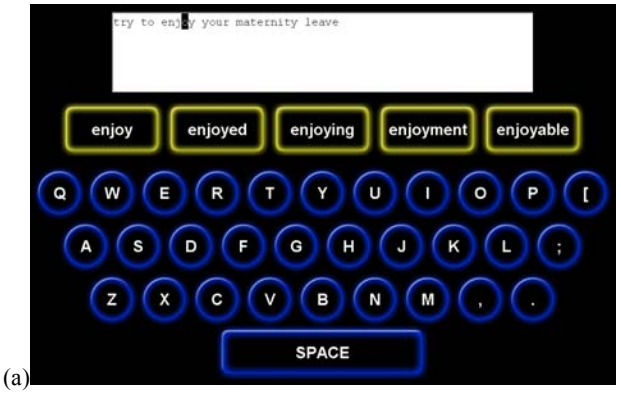
Clearly, the fixation algorithm's success is predicated on the correctness of the first few letters in a word. If a user enters a wrong first letter, the rest of the word could be wrong. This might cause users to select the wrong letter. The extent this occurs in practice requires empirical testing in an experiment.

2.2 Word Prediction

For decades, researchers have sought to improve eye typing speed and accuracy using, for example word prediction and word completion. As entry proceeds, a list of candidate words is produced [MacKenzie et al. 2006; Wang et al. 2001] (Figure 4) and the user selects the desired word, if present. These techniques can reduce the number of keystrokes per character (KSPC) of text entered [MacKenzie 2002], and may increase text entry speed.

It is tricky to predict the word after a SPACE. While some systems use semantic information from previously entered text [Hansen et al. 2002], we used a simple approach. Following SPACE, candidates are chosen from the most frequent words in English, such "the", "of", and "an".

Increasing the size of the candidate word list reduces KSPC, but increases the visual scan time, since there are more words to consider. We used a list size of five, as there is little improvement in KSPC for larger lists [MacKenzie 2002; MacKenzie et al. 2006]. Another consideration for eye tracking is the need for large buttons: a large candidate list is simply not practical.



(a)

(b)

Figure 4. On-screen keyboard with word prediction (a) big buttons (100 pixels) (b) small buttons (76 pixels)

2.3 Letter prediction

Similar to word prediction, letter prediction may accelerate eye typing by highlighting a few highly probable next letters on the on-screen keyboard (Figure 5). Our implementation highlights three letters. If the desired letter is highlighted, users can select faster since the number of stimuli is reduced from 26 to 3. However, the highlighting is distracting if the desired letter is not among them. In fact, users might need more time to find the desired letter.

Figure 5. On-screen keyboard with letter prediction. The text phrase at the top is presented to the user for input.

2.4 Implementation details

Since there is no tactile feedback, visual and audio feedback are important for eye typing. Our system used four distinct neon-glow buttons, from left to right, normal, letter-predict, eye-over, and clicked (Figure 6).

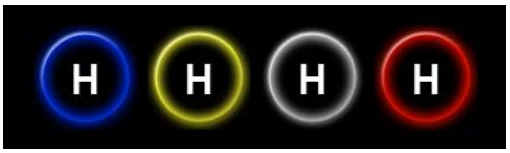


Figure 6. Button highlighting

A window of six seconds was allowed to select a button, otherwise a time-out error was recorded.

Our testing system presented users with text phrases to enter. As entry proceeds, the next letter to enter is shown in reverse video. With correct entry, highlighting simply progresses through the phrase, letter by letter. If a timeout occurs, an "X" with a gray background is shown (Figure 7). If there is a typing mistake, the inputted letter with a gray background is shown. It was anticipated that gaze shifts would be minimized since the presented and transcribed text are superimposed.

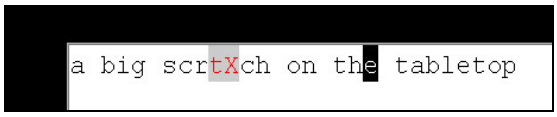


Figure 7. Text area

Our system also used beep-on-error for a typing error. For correct keystrokes, we also added a click sound similar to a key press on a mechanical typewriter.

We next describe an experiment to empirically test our system.

3 Methodology

3.1 Participants

We used ten paid participants (8 male, 2 female; 19-34 years; 3-10 hours per day computer usage). Half had prior experience with eye tracking. All had normal vision. Eight were right-eye dominant, two left-eye dominant, as determined using the test described by Collins and Blackwell [1974].

3.2 Apparatus

We used an Arrington Research *ViewPoint*™ head-fixed eye tracker with the camera focused on a participant's dominant eye (Figure 8). The monitor was a 19" 1280×1024 LCD at a distance of ~60 cm. The eye tracker sampled at 30 Hz with accuracy of 0.25°-1.0° visual arc (~10-40 pixels). Calibration was performed initially with re-calibration as needed. Software screen snaps are shown in Figures 4, 5, and 8. The fixation algorithm used weighting method 1 and tolerantDrift = 70 pixels.


Figure 8. Head-fixed eye tracking system

3.3 Procedure

Participants were briefed on the goal of the experiment and on the eye tracker and software. They were seated in front of the apparatus and calibrated, and then given a few practice phrases before testing. For input, participants "pointed" with the eye and "selected" with the CONTROL key. A key press was used for selection since it is fast and offers an interesting variation to dwell-time selection [Ware and Mikaelian 1987].

Text was generated from a set of 500 phrases [MacKenzie and Soukoreff 2003]. For each trial, a popup window presented a phrase to enter. Participants viewed and memorized the phrase, then entered it as quickly and accurately as possible. Time started upon pressing an OK button and stopped with the last letter in a phrase. Ten phrases were entered for each test condition.

3.4 Design

There were eight test conditions organized in a 2 × 2 × 2 within subjects design. There were three independent variables:

Prediction mode:	W	- Word prediction
	L	- Letter prediction
Fixation algorithm:	O	- On (weighting method 1)
	F	- Off
Button size:	B	- Big
	S	- Small

Letter codes are used to identify test conditions (e.g., WOS = Word prediction, algorithm On, Small buttons; or W-S = Word prediction, Small buttons). Conditions were assigned using nested counterbalancing to offset learning effects.

The total number of phrases entered was 800 (10 participants × 2 prediction modes × 2 fixation algorithms × 2 button sizes × 10 sentences per condition).

As typical of eye tracking experiments, considerable raw data were collected. These were filtered, reduced, and aggregated into sixteen dependent variables, of which only entry speed (words per minute) and error rate (%) are presented here.

4 Results and Discussion

4.1 Entry speed

Entry speed ranged from 10.8 wpm to 12.3 wpm (see Figure 9). These rates are reasonable for eye input. They lie between the 7 wpm reported by Marajanta et al. [2006] and the 25 wpm reported by Ward and MacKay [2002].

In comparing entry speeds, letter prediction was about 10% faster than word prediction with small buttons (W-S vs. L-S in

Figure 9) and similar to word prediction with large buttons (L-B vs. W-B). This suggests that letter prediction is useful if the screen size is small.

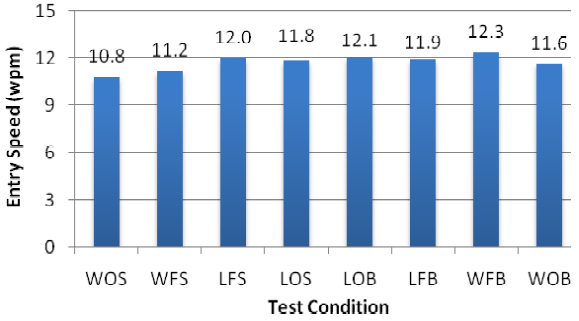


Figure 9. Entry speed vs. test condition

Entry speeds were about 10% faster with big buttons when using word prediction (W-S vs. W-B). The bigger size made it easier to recognize predicted words, which resulted in a faster entry speed. However, there was little or no improvement in entry speed when using letter prediction, likely because all conditions used the QWERTY layout. Letter prediction would likely be better with an unfamiliar layout, such OPTI [MacKenzie and Zhang 1999] or Metropolis [Zhai et al. 2000].

Contrary to expectations, entry speeds when using the fixation algorithm were lower than not using the algorithm for both letter and word prediction, except LOB vs. LFB. The parameters in the algorithm clearly need further consideration and refinement.

4.2 Error rate

Error rates (Figure 10) were lower with big buttons. There was a near 35% improvement for letter prediction using our fixation algorithm (LOB vs. LOS). Evidently, big buttons helped participants fixate near the center. Even with minor drift or calibration error, participants' fixation point did not tend to move outside the big button.

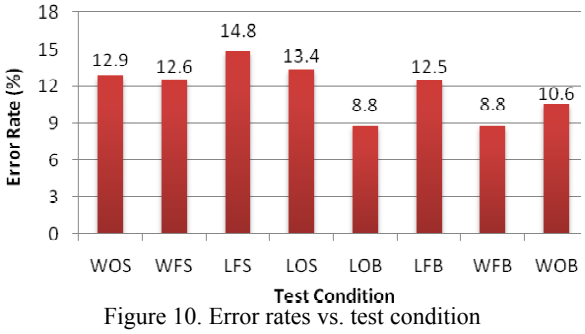


Figure 10. Error rates vs. test condition

The fixation algorithm yielded an improvement in error rate of about 10% to 30% when using letter prediction (LO- vs. LF- in Figure 10). However with word prediction, error rates were higher. If the first letters of a word were wrong, the fixation algorithm was more likely to choose the wrong letter, unless fixation was in the central region of the button. Participants exhibited some calibration problems and minor fixation drift during the experiment. From our observations, incorrect entry of the first few letters in a word was a significant problem for these participants, and in these cases there was little or no benefit in the use of the fixation algorithm. An implementation variation might be to turn on letter prediction only after, say, the second letter in a word. This would fully enlist the user in getting the first few letters correct.

Letter prediction was hoped to maintain the same entry speed and error rate as word prediction, while avoiding the distraction of candidate words. Although error rates for letter prediction were higher than with word prediction when the button size was small, entry speeds were faster. Thus, letter prediction through highlighting of highly-probable next letters holds promise.

Time-out errors were under 2% for all test conditions.

5 Conclusion

Letter prediction was as good as word prediction, and in some cases better. Our fixation algorithm proved sensitive to the correctness of the first letters in a word. It reduced error rates, but only in letter prediction mode.

More work is planned to develop a dynamic fixation algorithm to better adapt to different situations.

References

COLLINS, J. F. & BLACKWELL, L. K. 1974. Effects of eye dominance and retinal distance on binocular rivalry, *Perceptual Motor Skills*, *39*, 747-754.

HANSEN, D. W., HANSEN, J. P., NIELSEN, M., JOHANSEN, A. S., & STEGMANN, M. B. 2002. Eye typing using Markov and active appearance models, *Proc 6th Workshop App Computer Vision* (IEEE) 132-136.

HUTCHINSON, T. E., WHITE JR., K. P., MARTIN, W. N., REICHERT, K.C., & FREY, L.A. 1989. HCI using eye-gaze input, *IEEE Trans Systems Man Cybernetics*, *19*, 1527-1534.

LANKFORD, C. 2000. Effective eye-gaze input into Windows, *ETRA 00* (ACM) 23-27.

MACKENZIE, I. S. 2002. KSPC as a characteristic of text entry techniques, *Proc MobileHCI 02* (Springer-Verlag) 195-210.

MACKENZIE, I. S., CHEN, J., & ONISZCZAK, A. 2006. Unipad: Single-stroke text entry with language-based acceleration, *NordiCHI 06*, (ACM), 78-85.

MACKENZIE, I. S. & SOUKOREFF, R. W. 2003. Phrase sets for evaluating text entry techniques, *Proc CHI 03* (ACM) 754-755.

MACKENZIE, I. S. and ZHANG, S. X. 1999. The design and evaluation of a high-performance soft keyboard, *Proc CHI 99* (ACM) 25-31.

MAJARANTA, P., MACKENZIE, I.S., AULA, A., & RÄIHÄ, K.-J. 2006. Effects of feedback and dwell time on eye typing speed and accuracy, *UAIS*, *5*, 199-208.

MAJARANTA, P. & RÄIHÄ, K.-J. 2002. Twenty years of eye typing, *Proc ETRA 02* (ACM) 15-22.

ŠPAKOV, O. & MINIOTAS, D. 2004. On-line adjustment of dwell time for target selection by gaze, *Proc NordiCHI 04* (ACM) 203-206.

SALVUCCI, D. D. 1999. Inferring intent in eye-based interfaces, *Proc CHI 99* (ACM) 254-261.

WANG, J., ZHAI, S., & SU, H. 2001. Chinese input with keyboard and eye-tracking, *Proc CHI 01* (ACM) 349 - 356.

WARD, D. J. & MACKAY, D. J. C. 2002. Fast hands-free writing by gaze direction, *Nature*, *418*, 838.

WARE, C. & MIKAELIAN, H. H. 1987. Evaluation of eye tracker for computer input, *Proc CHI+GI 87* (ACM) 183-188.

ZHAI, S., HUNTER, M., & SMITH, B. A. 2000. Metropolis keyboard, *Proc UIST 00* (ACM) 119-128.