GlanceWriter: Writing Text by Glancing Over Letters with Gaze

Wenzhe Cui*
Stony Brook University
Stony Brook, NY, USA
wecui@cs.stonybrook.edu

Rui Liu*
Stony Brook University
Stony Brook, NY, USA
ruiliu1@cs.stonybrook.edu

Zhi Li
Stony Brook University
Stony Brook, NY, USA
zhili3@cs.stonybrook.edu

Yifan Wang
Stony Brook University
Stony Brook, NY, USA
yifanwang@cs.stonybrook.edu

Andrew Wang
University of Massachusetts Amherst
Amherst, MA, USA
andhwang@umass.edu

Xia Zhao
Stony Brook Medicine
Stony Brook, NY, USA
xia.zhao@stonybrookmedicine.edu

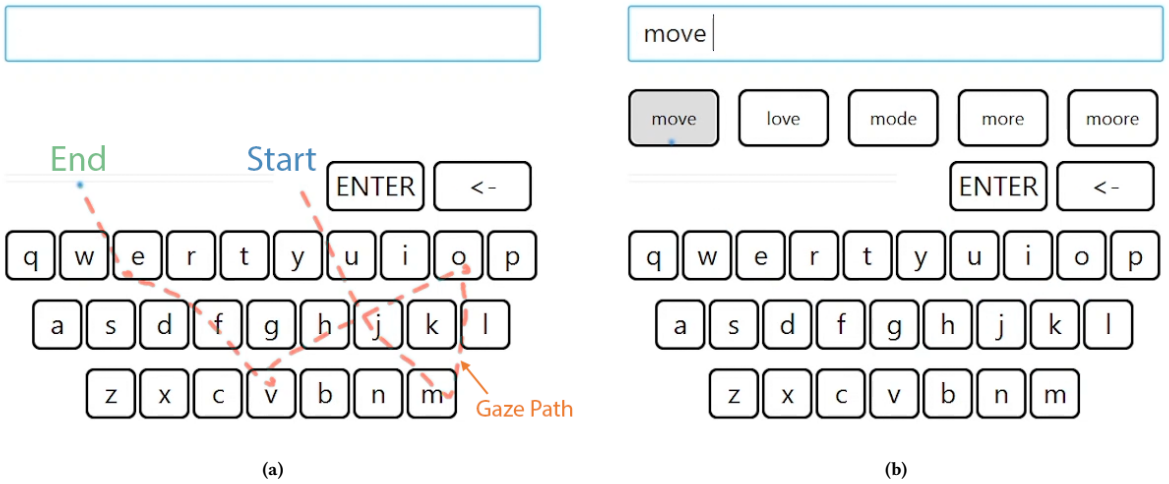
Sina Rashidian
Verily
San Francisco, CA, USA
srashidian@verily.com

Furqan Baig
UIUC
Urbana, IL, USA
fbaig@illinois.edu

I.V. Ramakrishnan
Stony Brook University
Stony Brook, NY, USA
ram@cs.stonybrook.edu

Fusheng Wang
Stony Brook University
Stony Brook, NY, USA
fusheng.wang@stonybrook.edu

Xiaojun Bi
Stony Brook University
Stony Brook, NY, USA
xiaojun@cs.stonybrook.edu



(a) (b)

Figure 1: This figure shows how a user enters "move" via GlanceWriter. (a) The user moves eye-gaze position (the blue dot) to the keyboard area from a place above the keyboard, glances over 'm', 'o', 'v' and 'e', and moves the gaze position away from the keyboard to signal the end of gaze gesture. No dwelling or "reverse crossing" [13] is required during the process. (b) GlanceWriter decodes the gaze path and shows the results above the keyboard. The user then dwells on "move" with gaze to confirm it.

*Co-first authors ordered alphabetically.

CHI '23, April 23–28, 2023, Hamburg, Germany

ABSTRACT

Writing text with eye gaze only is an appealing hands-free text entry method. However, existing gaze-based text entry methods introduce eye fatigue and are slow in typing speed because they

ACM ISBN 978-1-4503-9421-5/23/04…$15.00
https://doi.org/10.1145/3544548.3581269

often require users to dwell on letters of a word, or mark the starting and ending positions of a gaze path with extra operations for entering a word. In this paper, we propose GlanceWriter, a text entry method that allows users to enter text by glancing over keys one by one without any need to dwell on any keys or specify the starting and ending positions of a gaze path when typing a word. To achieve so, GlanceWriter probabilistically determines the letters to be typed based on the dynamics of gaze movements and gaze locations. Our user studies demonstrate that GlanceWriter significantly improves the text entry performance over EyeSwipe, a dwell-free input method using "reverse crossing" to identify the starting and ending keys. GlanceWriter also outperforms the dwell-free gaze input method of Tobii's Communicator 5, a commercial eye gaze-based communication system. Overall, GlanceWriter achieves dwell-free and crossing-free text entry by probabilistically decoding gaze paths, offering a promising gaze-based text entry method.

CCS CONCEPTS

• **Human-centered computing → Text input**; **Interaction design**; **Interaction devices**; *User studies*.

KEYWORDS

eye gaze, text input, gesture input, word gesture

ACM Reference Format:
Wenzhe Cui, Rui Liu, Zhi Li, Yifan Wang, Andrew Wang, Xia Zhao, Sina Rashidian, Furqan Baig, I.V. Ramakrishnan, Fusheng Wang, and Xiaojun Bi. 2023. GlanceWriter: Writing Text by Glancing Over Letters with Gaze. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3544548.3581269

1 INTRODUCTION

Eye-gaze is an important modality for hands-free text entry in the modern Post-PC computing era. It frees users from typing on a physical keyboard, which is especially advantageous in scenarios where physical keyboards are unavailable such as in virtual or augmented reality environments [38]. Eye-gaze text entry is also beneficial for individuals with motor disabilities who may not be able to speak [19, 46]. A common gaze-based text entry method [21, 32, 35] is dwell-based: a user dwells on a letter for a certain period of time (e.g., 600 ms [26]) to type it. However, frequent dwelling can be time-consuming and cumbersome. A sizable amount of research has been conducted to enable dwell-free text entry [14, 25, 26, 40]. One promising approach is to use gesture typing (or ShapeWriting) [14, 24, 29], which allows a user to enter a word by glancing over keys without deliberately dwelling on them. EyeSwipe [26] is one such technique and is demonstrated to be effective.

One major challenge faced by gaze-based gesture typing is that it is hard to determine the starting and ending positions of a gaze path. While touch-based gesture typing can rely on the finger landing on or leaving the screen to identify the starting and ending of the gesture, gaze-based input does not have these distinct "landing on" or "taking off" moments as the gaze trajectory is always active. EyeSwipe addressed this challenge by "reverse crossing" [13]: the user moves the gaze out of the starting or ending key to an icon above the key and returns to the starting or ending key again to signal the starting or ending letter of a word. However, this method can be time-consuming and requires extra effort. TAGSwipe [25] addressed this challenge by using an extra input modality: the user needs to use a hand to press on a touchscreen to signal the starting or ending position of a gaze path. This method requires hand movement and a touchscreen device which is not suited to scenarios where gaze is the only available input modality.

In this paper, we present GlanceWriter, as shown in Figure 1, a dwell-free and crossing-free text entry method, which allows users to enter words by glancing over letters one by one, without any "reverse crossing" or dwelling actions. GlanceWriter employs a new decoding algorithm that probabilistically determines which letter is entered by examining the dynamics of gaze movements and gaze locations. It can also probabilistically determine which keys correspond to the starting/ending letters, thus eliminating the need for "reverse crossing" actions.

Our user studies show that GlanceWriter is feasible and easy to learn for eye-gaze text input. It improves the text entry performance over EyeSwipe, with a speed increase from 6.49 WPM to 10.89 WPM and an error rate reduction from 6.85% to 2.71%. We also compared GlanceWriter with a commercial gaze-based text entry method of Communicator 5 in Tobii Dynavox I-12+ [10]. GlanceWriter improves the word-level text entry speed from 7.41 WPM to 9.54 WPM and reduces the error rate from 16.32% to 12.89%.

2 RELATED WORK

Our research is related to eye gaze-based text entry techniques, including dwell-based methods, dwell-free methods, gesture typing techniques, and eye-gaze techniques.

2.1 Dwell-based Gaze Text Entry

Dwell-based gaze text entry is the most common method for gaze typing: a user is required to look at a key that they would like to pick and fixate on it for a certain duration to select [33, 43]. This duration, also called dwell time, is the key factor that slows the text entry process and induces eye fatigue. The typical dwell time is between 400 and 1000 ms. Even though we can use shorter dwell time to increase typing speed, it also increases the risk of typing errors or unwanted selections due to the *Midas Touch problem* [22]. Majaranta et al. [32] conducted a longitudinal study to learn users' gaze-based typing using an adjustable dwell time. Isomoto et al. [21] presented a dwell time reduction technique for gaze-based target acquisition by adopting Fitts' Law and achieved an average dwell time of 86.7 ms with a 10.0% Midas Touch rate. Mott et al. [35] presented a cascading dwell gaze typing to dynamically adjusts the dwell time of keys and conducted experiments with ALS patients to show that this technique has the potential to improve gaze typing. In general, language models such as word prediction can also accelerate the dwell-based typing methods [9, 42].

2.2 Dwell-free Gaze Text Entry

To overcome the drawback of dwell-based gaze input, previous research has explored using gesture typing [24, 29] to achieve dwell-free gaze input. However, traditional touch-based gesture typing methods cannot be directly used for gaze-based gesture typing. A fundamental difference between them is that in touch-based

gesture typing, the finger gesture has well-defined starting and ending points (i.e. landing the finger on the screen to start a gesture and lifting the finger up to end the gesture), while in gaze-based gesture typing, it is not clear when is the start and end of the gesture. EyeSwipe, a dwell-free gaze typing method proposed by Kurauchi et al. [26], requires a user to do a "reverse crossing" [13] to mark the starting and ending of the gesture input. The built-in keyboard in Communicator 5 of Tobii Dynavox devices [10] also supports dwell-free gaze text entry. However, there is no public document explaining how to achieve it. It is a "black box" to users and outside developers. Nevertheless, we compared GlanceWriter with this keyboard in our second user study (Section 5). Some works also introduce other modalities to help with faster typing. TAGSwipe [25], a touch-assisted gaze swipe method for text entry, uses touch to confirm the starting and ending positions of a gaze path. HGaze typing [14], proposed by Feng et al., combines the simplicity of head movement with the gaze-based gesture input where a user could nod or shake her head for common commands such as deletion and revision, to provide efficient and comfortable dwell-free text entry.

There are also other solutions for dwell-free gaze-based text entry. For example, Sarcar et al. proposed EyeK [40], a gaze-based text entry system which types a key by moving the gaze position inside, outside and inside the key. Huckauf et al. proposed pEYEWrite [18], an expandable pie menu with letter groups. TAGSwipe [25] uses a touch press/release on an extra touch surface to signal the starting/ending positions of a gaze path and decodes the gaze path using a Dynamic Time Warping algorithm.

Dasher [47], a gaze-based zooming interface designed by Tuisku et al., also showed a higher text entry rate compared to a dwell-based keyboard [39]. Pedrosa et al. proposed Filteryedping [37], a dwell-free technique that can filter out unintentionally selected letters from users' input based on string matching. Kurauchi et al. proposed Swipe&Switch [27], a text-entry interface that allows users to swipe and switch on different keyboard areas to improve gaze-based interaction with a better user experience. Swipe&Switch removes "reverse crossing"; however, it still requires a clear definition of the starting and ending letter positions by gaze fixation: "either the first fixation or any other fixation longer than a hidden dwell-time (set to 700ms during our experiment)." [27] In contrast, GlanceWriter does not require any sort of fixation for entering letters because any letters near the gaze trajectory are considered and assigned probabilities.

Although many previous works achieved dwell-free gaze input, they either require marking the starting and ending positions of a gaze path such as "reverse crossing" [26], touching [25], fixation [27], or require deterministically looking at all the letters that compose the word to be typed [37]. GlanceWriter eliminates the need for these actions and achieves dwell-free and crossing-free input by probabilistically decoding the gaze path.

In addition to eliminating the need to specify the starting and ending positions of a gaze gesture, the probabilistic decoding algorithm of GlanceWriter is another differentiator against other dwell-free gaze input methods (e.g., Filteryedping [37]). GlanceWriter represents the probabilities of entering letters with key scores and calculates them by examining the dynamic of gaze trajectories such as gaze motion stability and distance to keys. In contrast, Filteryedping [37] does not introduce probability and decodes a gaze gesture based on filtering. It determines letters by examining which keys the user looks at and matches the letter sequence with words in the dictionary. By introducing the probability in decoding, GlanceWriter can quantify the uncertainty of entering a word prior to involving the word frequency, which is a principled approach of handling noises in text entry.

2.3 Gesture Typing

Gesture typing is a text entry technique that has been widely used on touchscreen mobile devices [6, 24, 29, 54–56], and it is also known as word-gesture typing or shape writing. Users may draw a path that passes the letters of a word one by one to input it, instead of typing on each individual key, thus improving the input speed. The idea of writing words as shapes was introduced in Shorthand Aided Rapid Keyboarding (SHARK$\hat{2}$) [24, 54]. It uses shape and location channels to measure the distance between gestures made by the user on a virtual keyboard to words in a lexicon, then to determine the input word.

There have been many works that aim at improving gesture input. For example, Alsharif et al. proposed to use Long Short Term Memory (LSTM) neural network for gesture decoding [1]. Bi et al. proposed a bi-manual gesture keyboard extending the gesture input from one finger to multiple fingers [3]. Yu et al. proposed to improve gesture typing by incorporating head movement information [53]. On the other hand, there are works that extend gesture typing to other devices or modalities. For example, tilt-based gesture typing [51], where the user draws a gesture by tilting the device, and others like mid-air gesture input [34], back of device gesture input [7], gesture input on a watch [15] and on a ring [16]. There are also some works investigated on gesture typing without visual feedback [4, 59]. The present work further extends this paradigm to gaze input.

2.4 Tracking Eye Gaze

Eye-gaze tracking measures eye movement relative to the head or point of gaze, and it enables gaze-based interactions. The gaze-tracking technology is becoming increasingly available and there are a bunch of gaze trackers for us to choose from. For example, there are readily commercial products like trackers from Tobii [1], SMI REDn [2], Eyelink 1000 plus [3], etc., and they are widely used in eye-gaze related research works [11, 20, 25, 41, 44, 45]. These gaze trackers are peripheral devices to a computer, and usually have good gaze tracking quality, i.e. the accuracy is less than 2°. However, one major disadvantage of them is that they usually cost from several thousand dollars up to more than ten thousand dollars.

To make gaze-tracking more pervasive, researchers have been devoted to enabling gaze-tracking on daily devices by using the embedded camera. For example, Wood et al. developed EyeTab [50], which uses the front-facing RGB camera of a tablet for gaze tracking. Huang et al. proposed an in-situ gaze tracking method on smartphones based on the glint of the screen on the user's cornea [17].

[1] https://gaming.tobii.com/product/eye-tracker-5/, https://www.tobiipro.com/, https://www.tobiidynavox.com/
[2] https://imotions.com/hardware/smi-redn-scientific/
[3] https://www.sr-research.com/eyelink-1000-plus/

Papoutsaki et al. proposed WebGazer [36], a JavaScript library that uses webcams to enable gaze tracking for web browsers. Li et al. used the ARKit to enable gaze-tracking on an iPad with TrueDepth camera [28]. Many works use Deep Learning techniques to support gaze-tracking [23, 49, 57]. Even though embedded camera-based gaze tracking has been investigated for years, the state-of-the-art tracking quality is still lower than those commercial products, because the tracking quality is limited by the camera. In this paper, we used the Tobii Dynavox as our gaze tracker for its high tracking accuracy.

3 GLANCEWRITER DECODING ALGORITHM

A core component of GlanceWriter is the decoder, which probabilistically maps a gaze path to a word in the dictionary. The starting/ending moment of a gaze path is defined by the moment the gaze path enters/exits the soft keyboard. In our implementation, the gaze path was sampled at 100 Hz. In other words, it consisted of a sequence of gaze points sampled at every 0.01 second. Figure 1 shows how GlanceWriter works for typing the word "move".

The challenge of dwell-free and crossing-free gaze input is that it is hard to determine the starting (or ending) position on a gaze path for entering the first (or last) letter of a word. A gaze path is defined by the movements of the gaze entering and exiting the keyboard area. The starting position for entering the first letter (or the ending position for entering the last letter) could be any position on the defined gaze path. Therefore, the traditional gesture typing decoder such as SHARK2 [24] is unsuitable because it requires knowledge of the positions along the gesture corresponding to the starting and ending letters of a word. We have created a new decoder to address this challenge. It probabilistically determines which letter is typed based on gaze dynamics, including the starting and ending letters. The gaze dynamics are the characteristics of the motion of the gaze point, such as the speed of gaze points and the distances between a gaze point and key centers around it. In this section, we explain how the decoding algorithm works.

3.1 Data Structure

To map a gaze path to a word, it is important to represent and store words in the decoder properly. We implemented a trie structure [48] (prefix tree) to store all the words in the lexicon as shown in Figure 2a. The root is associated with an empty character. Each of the other nodes is associated with a character. Each node consists of at most 26 children, corresponding to 26 letters in English. All the children of a node have a common prefix. The prefix is the characters from their parent to the root in a reversal order. If a node is the last letter of a word, we store the word in that node. If the word contains consecutive identical letters, these letters are merged into the same node. For example, in Figure 2a, "more" and "moore" both can be found in node 'e' following 'm', 'o', 'r' since "oo" is merged at node 'o'. To optimize the calculation, for each node, we also store the key score (explained later) and the sum of scores from current nodes backward to root.

3.2 State of Node (Key)

In the trie, each node has two states RELEASE and HOLD. RELEASE means the gazing point has never reached the node. HOLD means the node has been visited by a gaze path. Thus, the initial state of a node is set as RELEASE. As the red dot showed in Figure 2b, when the gaze path moves into a key i, the decoder first finds the words which start with i and marks the corresponding starting node as HOLD. It also sets the score k_i of the key i to the node. In the meantime, the decoder also checks all the HOLD nodes. If their next nodes are identical to the key i and marked as RELEASE, the decoder marks them as HOLD. Same as the initial nodes, the decoder also sets k_i to these nodes. When the gaze path is moving inside the key, as the blue dot showed in Figure 2b, the decoder updates the key's score of all the HOLD nodes identical to the key. A key's score is updated only when it is greater than the previous value.

When the gazing path reaches the last letter (node) of a word w, the decoder sets the word as an output candidate. The score of the word S_w is the sum of its key scores of all its nodes, as shown in Figure 2a. After the gazing point moves out of the key of a candidate's last node, the decoder keeps the word in the input candidates for a certain duration T; after that the word is removed from the candidates.

When the gazing point moves out of the upper bound of the keyboard, the decoding process stops. The decoder sorts the candidates by their scores in descending order and shows the top 5 candidates as the input suggestions. In the above example, the word "move" has the highest score, followed by words "more", "mode" and so on.

3.3 Key Score

The key score of a key i (denoted by $K(i)$) represents the possibility of a key being the key a user wants to type. It is determined by two properties of gaze points on a gaze path: 1) the distance between the point and the key center (distance score), and 2) the gazing stability (stability score). Our calculation was based on the intuition that when the user intends to input a letter of a word, the gazing point on that letter is close to its key center and stays still in the boundary of a key.

1. Distance score $D(i, p)$. We assume that the distance (pixel) from a gaze point p to the center of the key i follows a Gaussian distribution. It means that if the Euclidean distance of a gazing point p to a key center is d, the probability of the key being the target of the point can be calculated using Gaussian probability density function:

$$D(i,p) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(d-\mu)^2 \Big/ 2\sigma^2}, \tag{1}$$

where μ is 0. We empirically set σ as 0.4. For a given key, we used the gaze point that was closest to the key center to calculate the distance score. We used a Gaussian distribution because such a distribution ensures that a gaze point near the key center will result in a high distance score while the distance score will decrease rapidly as the gaze point moves away from the key center. Previous research [28] has shown that a Gaussian distribution works well for describing the probability of selecting a target based on gaze points. We followed the previous work [28] to adopt a Gaussian function here.

2. Gaze stability score $G(i, p)$. It measures to what degree a gaze point p stays still inside a key i. We first calculated the average speed of gaze point in a L-size (L = 30 pixels, determined empirically)

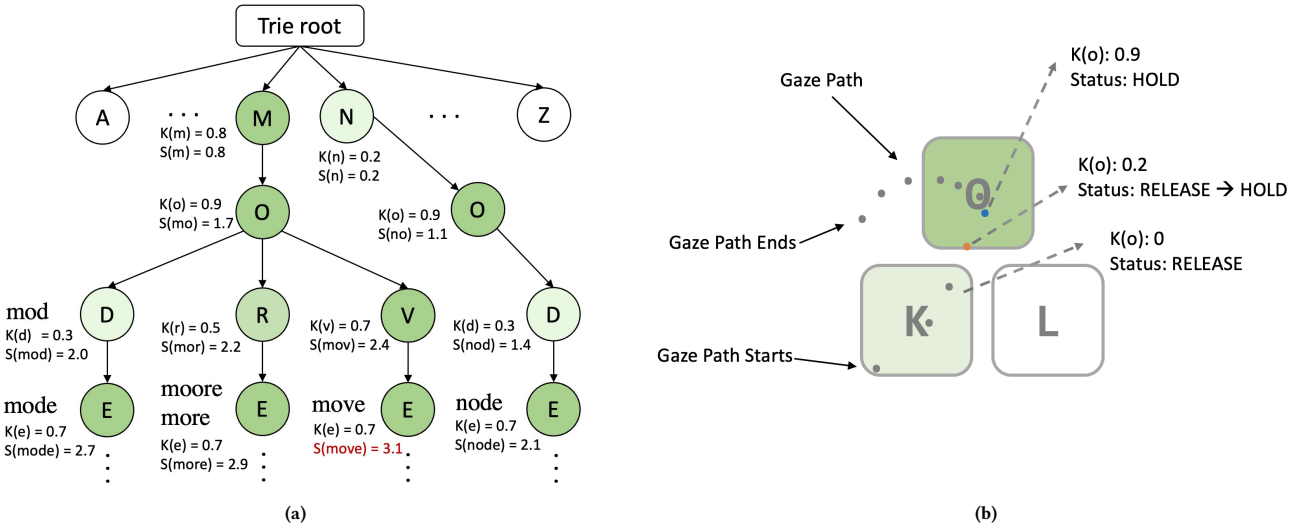


Figure 2: An example of typing the word "move". (a) The data structure (trie) of GlanceWriter. The Key score of a key i is denoted by K(i), while the score of a word, denoted by S(word), is the summation of its key scores from root to itself. (b) States and candidates change with eye-gaze points. The gaze points are moving from key 'K' to 'O', marking the key 'O' as HOLD. The score is reflected by the color intensity for illustration purposes.

window right before p, which is the average over instant speeds of gaze points within the L-size window. We then used the reciprocal of this average speed as the stability score. We chose a L-size moving window to make sure that moving gaze points were sampled for calculating the stability score. It avoids the scenario in which all the sampled gaze points were stationary, which could result in a very high value for the reciprocal of the average speed.

The key score $k(i,p)$ for a given gaze point p at key i is the product of distance score and stability score:

$$k(i,p) = D(i,p) \cdot G(i,p), \tag{2}$$

The key score $K(i)$ for a given key i is the maximum of $k(i,p)$ over all the gaze points within the boundary of the key i:

$$K(i) = \max_{p \in P} k(i,p), \tag{3}$$

where P includes all gaze points within the boundaries of key i. The max function is implemented to keep the highest score a key can get for a given gaze path.

3.4 Word Score

The word score $S(w)$ represents how likely a user intends to input a word w with a given gaze path. It is the summation of key scores for corresponding letters:

$$S(w) = \sum_{i \in M} K(i), \tag{4}$$

where M includes all nodes (letters) in the trie for the word w (e.g., Figure 2).

3.5 Combining Word Score with Language Model

The basic principle of word-gesture decoding [24, 55] is to combine the probability of a candidate word estimated from the gazing points (a.k.a. spatial probability $c(w)$) and from the language context (a.k.a. language probability $l(w)$) to obtain the overall probability of a word w being the intended input given an input gesture:

$$Score(w) = \frac{l(w)c(w)}{\sum_{i \in W} l(i)c(i)}, \tag{5}$$

where W is a lexicon containing i words. We follow the same principle (Equation 5) for decoding.

More specifically, GlanceWriter approximates the spatial probability $c(w)$ with word score $S(w)$:

$$c(w) = \frac{S(w)}{\sum_{k \in C} S(k)}, \tag{6}$$

where C includes top t ($t = 10$) words with highest word scores. For language probability $l(w)$, we use a bi-gram language model (size: 7 million bi-grams) which is trained over the Corpus of Contemporary American English (COCA) [8] (2012 to 2017). The Corpus contains over 5 million sentences. After obtaining $c(w)$ and $l(w)$, the decoder calculates $Score(w)$ according to Equation 5, and outputs the top N word according to $Score(w)$. The decoding process ends here. Only words in the corpus will be considered. Algorithm 1 uses pseudocode to explain how the decoder calculates key score, word score, and combines them with a language model to decode a gaze path into a word.

This decoding algorithm in general should work for a virtual keyboard at any scale and with gaze trackers of different accuracy. That said, the parameters such as the size of L-size window and the standard deviation of the Gaussian distribution in Equation 1

should be empirically determined to suit keyboards at particular scales and for the specific eye tracker. We determined the values for keyboard size and eye tracker used in Experiment I.

Algorithm 1 GlanceWriter Decoding Algorithm

procedure GET DECODING RESULT
  RN : children of root node
  HN : list of trie nodes in HOLD state
  WC : list of word candidates with scores
  T : time limit to hold a word in WC
input:
  $i \leftarrow$ key where current eye-gaze point hovering on
  $k_i \leftarrow$ key score where current eye-gaze point hovering on
process:
  Trim HN, keep top 50 nodes sorted by their sum of key scores to the root
  for each $N_R \in RN$ **do** ▷ Scan the first letter of each word
    if i is the letter of N_R **and** HN doesn't contain N_R **then**
      add N_R to HN
  for each $N_H \in HN$ **do** ▷ Scan each node in HOLD list
    $k_h \leftarrow$ key score of N_H
    $S_h \leftarrow$ sum of the key scores from N_H to its root
    if i equals the letter of N_H **then**
      if $k_i > k_h$ **then** ▷ Update scores of a current node
        $S_h = S_h - k_h + k_i$
        $k_h = k_i$
    $NC \leftarrow$ children of N_H
    for each $N_C \in NC$ **do** ▷ Scan next letters
      $k_c \leftarrow$ key score of N_C
      $S_c \leftarrow$ sum of the key scores from N_C to its root
      if i equals the letter of N_C **then**
        $k_c = k_i, S_c = k_i + S_h$
        add N_C to HN ▷ add a child node to HOLD list
    Remove the words which are stored in WC more than T
    if N_H is the last letter of words **then**
      $W_h \leftarrow$ list of words in N_H
      for each $w \in W_h$ **do**
        if w not in WC **then**
          add w and its sum score to WC
output:
  Sort the WC by the sum score of each word and output top five word in WC

4 EXPERIMENT I

We first conducted a controlled experiment to compare GlanceWriter with EyeSwipe [26] in a text entry task, as one of our main goals was to eliminate the "reverse crossing" [13]. We aimed to understand whether the use of probabilistic decoding of gaze path would improve the text entry performance.

4.1 Participants and Apparatus

Fourteen users without disabilities (3 females) between 24 and 32 years old (average 27.36±2.46) participated in this study. All of them were familiar with the QWERTY keyboard layout. Their median familiarity with gesture typing was 4 (1: have never used gesture typing; 5: frequently use gesture typing). All participants had normal or correct-to-normal vision; 10 of them wore glasses and they could choose whether or not to wear glasses during the experiment. A Tobii Dynavox I-12+ (12.1-inch screen, Aspect ratio: 4:3, 1024×768 pixels resolution, Intel Celeron Qual Core Processor J1900, 4GB DDR3 RAM, 256GB SSD, Tobii IS4 eye tracking module) running Windows 10 was used in the experiment.

4.2 Design and Procedure

Experiment I was a text transcription task. We compared GlanceWriter with EyeSwipe [26], a dwell-free method. The typing interface of experiment is shown in Figure 3. We implemented the GlanceWriter decoder following the description in Section 3, and designed a keyboard interface as shown in Figure 3(b). The eye-gaze position was displayed as a blue point. As visual feedback, we highlighted the key when the gaze point is hovering on it. After drawing the gesture via gaze, a suggestion bar of 5 candidate words shows up below the text field. The user then confirms the intended word by focusing on the right candidate. A shrink-down animation served as the visual feedback helping the user to confirm the selection. The user needs to delete and re-enter a word if errors occur. Re-input was not counted as an error but was reflected in the total time cost. To delete the last word on the input field, the user needs to focus on the "<-" (backspace) button same as selecting a word. If the user needs to cancel a gaze path, she can move the gazing point out of the upper bound of the keyboard and starts to input the word once again. We also implemented EyeSwipe following the description in the previous work [26]. Selections of candidate words, enter, and backspace are performed by focusing on the highlighted key with shrink-down animation in GlanceWriter and by "reverse crossing", as described in EyeSwipe [26]. Both GlanceWriter and EyeSwipe used the same interface, and the same language model with 10K unique words as described in Section 3.5. The dwell time for word selection is set to 600ms in GlanceWriter.

EyeSwipe and GlanceWriter used different decoders. We re-implemented EyeSwipe following the description in the original paper [26], using a DTW-based decoding algorithm. In contrast, GlanceWriter used the probabilistic decoding algorithm described in Section 3. We used different decoders because our main purpose was to investigate whether the newly proposed decoding algorithm would outperform the existing dwell-free DTW-based input method. It was also necessary to use different decoders because the original DTW-based decoder [26] did not work for GlanceWriter. The DTW-based decoder requires knowledge on the starting and ending letters that are unavailable in GlanceWriter.

We adopted a within-subject design. Participants were asked to input the same set of phrases above the text field using two different gesture typing methods: EyeSwipe and GlanceWriter. To reduce the learning effect, we counter-balanced the order by dividing participants into two groups: half of the participants used EyeSwipe first while the other half used GlanceWriter first. We randomly selected 20 phrases from a subset of the MacKenzie and Soukoreff phrase set [31, 52]. The same set of phrases was used across two conditions and users. There were 4 sessions in each condition (EyeSwipe and

GlanceWriter). Each session contained 5 phrases. Participants were encouraged to type as fast and accurately as they could.

To begin with, participants were demonstrated how the Tobii eye tracking systems work, how to do calibration and use their eye gaze to navigate the program. The participants performed calibration before each session. Before starting the formal sessions of each method, participants completed a 10-minute warm-up session to familiarize themselves with each method. The phrases in the warm-up session were different from those in the formal test. Due to the participants' familiarity with eye gaze input and QWERTY layout, they were able to obtain eye-gaze text entry skills during the warm-up session. In the formal test, participants were allowed to take a break between sessions. Re-calibration was allowed during breaks. Each participant was able to complete the whole study within 40 minutes.

At the end of the study, the participants completed a questionnaire to report their subjective ratings and answered an open-ended question, "What's the most difficult part of the method you've just tried?" The subjective ratings were about mental demand, physical demand, comfort, learnability, scaled from 1 to 10, and overall preference of each typing method, scaled from 1 to 5.

In total, the study included: 14 participants × 2 methods × 20 phrases = 560 trials.

4.3 Results

Input Speed. This metric indicates how fast a user could input text with each method. It was measured by word per minute (WPM). We followed the calculation proposed in the previous work [30]:

$$WPM = \frac{|S-1|}{T} \times \frac{1}{5}, \tag{7}$$

where S is the length of the transcribed text in characters with space, and T is the elapsed time in minutes from start of the input to finishing the last word in the phrase. Figure 4 shows the average input speed in EyeSwipe and GlanceWriter conditions. The means (SD) were 6.49(SD = 1.69) for EyeSwipe and 10.89(SD = 2.62) for GlanceWriter. A dependent t-test showed the difference was significant ($t_{13} = 8.79, p < 0.001$).

Word Error Rate. In both two conditions, the input was on the word level. The error rate was measured using word error rate [2, 58]:

$$r = \frac{D(E,T)}{|T|} \times 100\%, \tag{8}$$

where $D(E,T)$ is the word edit distance between the transcribed phrase E and the target phrase T, and $|T|$ is the number of words in T. The word edit distance is the minimum number of basic word-level operations needed to transform the transcribed phrase into the target phrase. These basic operations are insertion, replacement, and deletion. The means (SD) were 6.85%(SD = 5.49%) for EyeSwipe and 2.71%(SD = 3.22) for GlanceWriter. A dependent t-test showed the difference was significant ($t_{13} = -2.69, p = 0.02$).

Subjective Rating. Participants were asked to provide a numerical rating on mental demand, physical demand (1: least demanding, 10: most demanding), comfort (1: very uncomfortable, 10: very comfortable) and learnability (1: very difficult to learn, 10: very easy to learn). Mental demand describes how much mental effort is required. Physical demand describes how much physical effort is required. We also let participants rate each method by their overall preference (1: very dislike, 5: very like). As shown in fig. 5, participants were in favor of GlanceWriter across all questions. A Wilcoxon Signed-Ranks Test indicated that the overall preference of GlanceWriter Keyboard was significantly higher than that of EyeSwipe (Z = 3.18, p = 0.001).

5 EXPERIMENT II

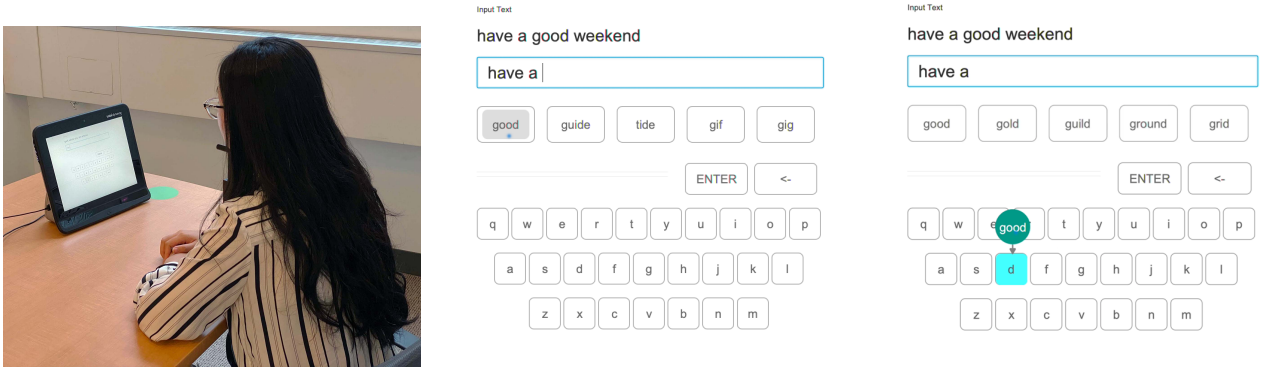
We conducted Experiment II to compare GlanceWriter with the built-in keyboard in Tobii Communicator 5 [10], which is also a dwell-free gaze text entry method. In this experiment, we aimed to understand the performance of GlanceWriter in comparison with a commercially available product, which represents the status quo of gaze-based input technology. As there is no publicly released document explaining how Communicator 5 decodes gaze paths, we could not reproduce its decoder. We therefore tested it as a "black box" as explained later.

5.1 Participants and Apparatus

Twelve users without disabilities (3 females) between 23 and 33 (average 26.92±2.96) participated in this study. All of them were familiar with the QWERTY keyboard layout. Their median familiarity with gesture typing was 4. All participants had normal or correct-to-normal vision; 7 of them wore glasses, and they could choose whether or not to wear glasses during the experiment. The participants in study I and II were different. A Tobii Dynavox I-13+ (13.3-inch screen, Aspect ratio: 16:9, 1920×1080 pixels resolution, Intel(R) Core(TM) i5-7300U, 8GB LPDDR4, 256GB SSD, Tobii IS5 eye tracking module) running Windows 10 was used for gaze collection and text entry. A MacBook Pro (16-inch screen, Aspect ratio: 16:10, 3072×1920 pixels resolution, 2.6GHz 6-Core Intel Core i7 Processor, 16GB 2667 MHz DDR4 Memory, AMD Radeon Pro 5300M Graphics) was used to display the text phrases and record time in the experiment.

5.2 Design and Procedure

Experiment II was a text transcription task. The design of this experiment was the same as Experiment I: we adopted a within-subject design. Participants were asked to type the same set of phrases as shown above in the text field using two different gesture typing methods: GlanceWriter and a dwell-free typing method in Communicator 5. We also did counterbalancing on the orders to reduce the learning effect. We randomly selected 20 phrases from a subset of the MacKenzie and Soukoreff phrase set [31, 52]. The same set of phrases was used across two conditions and users. There were 4 sessions in each condition (Communicator 5 and GlanceWriter). Each session contained 5 phrases. Participants were encouraged to type as fast and accurately as they could. Different from Experiment I, if they felt it was difficult to correct errors, they could skip error correction for both conditions. We provided a different input instruction from Experiment I because the first-level interface in Communicator 5 supported sentence deletion only, which made word-level error correction very difficult. To perform a single word-level deletion, a user needs to navigate to the second-level UI of Communicator 5, which includes 1) selecting the "Abc" button (Figure 6), 2) picking the word to be deleted, 3) typing the



(a) (b) (c)

Figure 3: (a) A participant is entering a phrase with GlanceWriter. (b) A screenshot of the GlanceWriter interface. (c) A screenshot of the EyeSwipe interface, which is identical to the interface for GlanceWriter. The user is confirming the input word "good" with "reverse crossing".

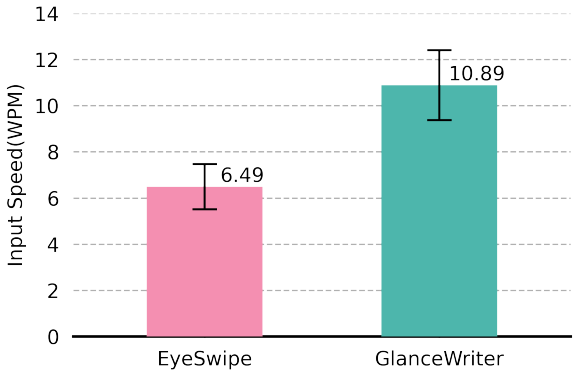


Figure 4: The Mean (95% *CI*) input speed (WPM) by input method

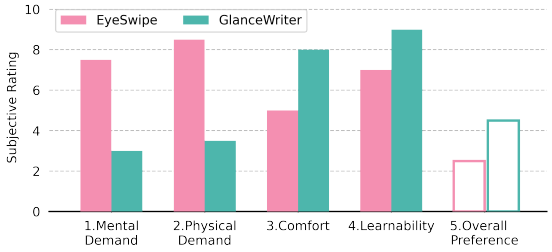


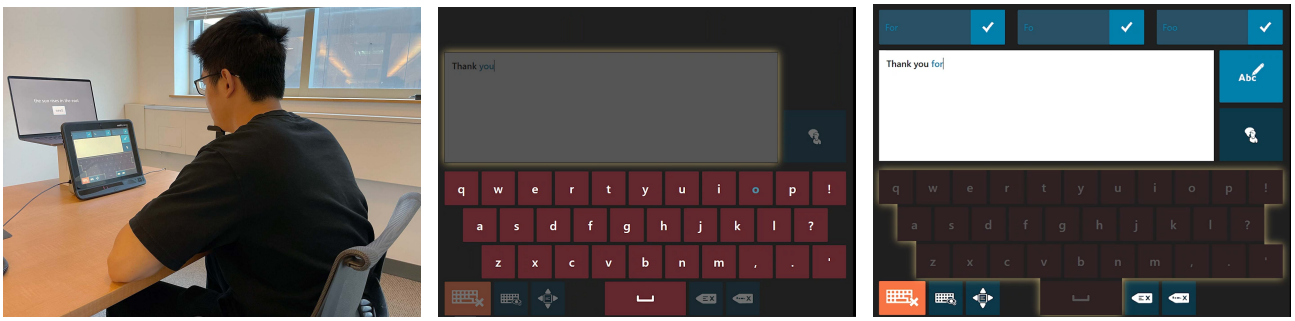
Figure 5: Medians of subjective ratings. For measures 1 and 2 regarding demand, a lower rating is better. For measures 3-5, a higher rating is better. GlanceWriter received more favorable ratings in all categories.

word again and correcting it if necessary, and 4) returning to the main interface. Each of these steps takes at least one dwell action, which would be slow and error-prone, increasing the cognitive load and causing eye fatigue. Therefore, we allowed the participants to skip error correction if they wanted to in order to reduce the negative impact of the cumbersome error correction experience of Communicator 5. We did not want the error-correct interface design to become the dominant factor determining the text entry performance.

5.2.1 Tobbi Communicator 5. The Tobii Communicator 5 keyboard supports both word-level and phrase-level input (as described in its manual): a user could enter a word or phrase with a continuous gaze path on the keyboard. We compared GlanceWriter with The Tobii Communicator 5 for the word-level text input only because GlanceWriter was designed as a word-level input method. The word-level input is also the most common text entry method as it provides confirmation for each word after it is entered.

The interface of Communicator 5 keyboard is shown in Figure 6b and Figure 6c. The interface includes four areas (from top to bottom): Suggestion area, Text field, Keyboard area and Function Area. In the Suggestion area, three candidate buttons were shown after every word input. In Text field, there are two buttons to the right of the text field: the button with "Abc" written on it provides alternative candidates and the button with a "speaker" icon provides a speaker. In the Function area at the bottom, five buttons are listed (from left to right) as 1) Exit keyboard, 2) Switch keyboard, 3) Symbols, 4) Clear All, and 5) Clear the last sentence. The space key in the middle of the utility bar is part of the keyboard. When a user intends to select a target via gaze, a loading icon in red will show up at the center of the target, and it takes one dwell time at the icon to select the target. There are two different modes in which Communicator 5 keyboard prevents unnecessary controls during typing. When a user enters the typing mode, as shown in Figure 6b, only the Keyboard area will be highlighted while the other areas are disabled. On the contrary, the suggestion, text field and utility areas will be highlighted when a user exits typing mode and enters the edit mode as shown in Figure 6c.

In Communicator 5 keyboard, the switch of typing mode and edit mode is done by dwelling at corresponding regions. When a



(a) (b) (c)

Figure 6: (a) A participant is entering a phrase with Communicator 5. (b) A screenshot of the Communicator 5's keyboard interface. The keyboard is activated after the gaze position enters the keyboard. (c) A screenshot of the Communicator 5's text field interface. The keyboard is deactivated after the gaze position exits the keyboard.

user wants to type, she looks at the keyboard region to start typing. The key on which her eye gaze hovers is highlighted in blue as visual feedback during typing. The user needs to look at the text region (including suggestion area, text field and function area) to exit after she finishes typing. If she wants to cancel a gaze path, she can re-enter the typing mode to input the intended word. In edit mode, the top candidate word appears in the text region while the following three candidates appear in the suggestion region. The dwell time of mode switching and candidate selection is set to 600ms.

In the Communicator 5 condition, we adopted the default interface and settings of Communicator 5 as shown in Figure 6b and Figure 6c. Because the Communicator 5 keyboard was a built-in keyboard that could not be used to enter text in other applications, we presented the phrases and recorded the duration on a separate computer (Figure 6a). To keep the condition identical, we hid the target phrases in the GlanceWriter interface and displayed them on the same separate computer as in Communicator 5. The input interface of GlanceWriter (without target phrase) was the same as designed in Experiment I.

5.2.2 Procedure. To begin with, participants were demonstrated how the Tobii eye-tracking system works, how to do calibration and how to type using the Communicator 5 and GlanceWriter dwell-free keyboard. The participants performed calibration before the experiment. Before starting the formal session, each participant completed a 10-minute warm-up session of 10 sentences which was different from the formal testing. We used a MacBook Pro to display the text to be transcribed. As shown in Figure 6a, the MacBook Pro was placed behind the Tobii device, where the user could look at the text without changing her position. There was a text and a "next" button on the monitor. The font size of the text was 80pt. The text color was set as white, while the background color was gray.

A participant clicked the "next" button when she finished the phrase. The elapsed time was the duration for entering the phrase. The experimenter also manually copied every phrase a user entered after the experiment as it cannot be stored in a computer. The input speed was calculated according to the length of transcribed text and the elapsed time collected by the timer. The error rate was calculated according to the word edit distance from the display and transcribed texts.

At the end of the study, the participants also completed a questionnaire to report their subjective rating about mental demand, physical demand, comfort, learnability, and overall satisfaction with each typing method.

In total, the study included: 12 participants × 2 methods × 20 phrases = 480 trials.

5.3 Results

Figure 7 shows the average input speed in Communicator 5 and GlanceWriter conditions. The means (SD) was 7.41(SD = 1.32) for Communicator 5 and 9.54(SD = 2.10) for GlanceWriter. A dependent t-test showed the difference was significant (t_{11} = 3.64, p = 0.003). For word error rate, the means (SD) were 16.32%(SD = 7.55%) for Communicator 5 and 12.89%(SD = 9.49) for GlanceWriter. A dependent t-test showed the difference was not significant ($t_{11} = -0.89, p = 0.31$).

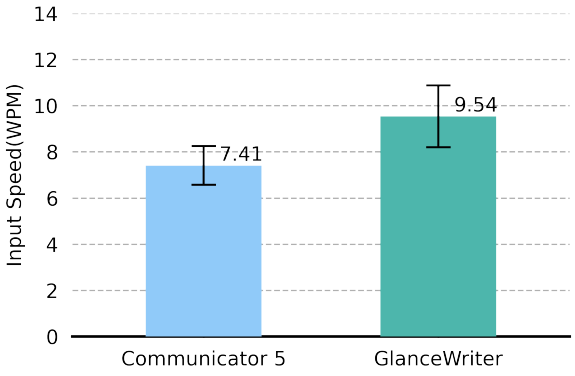


Figure 7: The Mean (95% CI) input speed (WPM) by input method

The typing speed of GlanceWriter was close to that in Experiment I, but the error rate of GlanceWriter in Experiment II was

higher than the error rate of GlanceWriter in Experiment I. There are two reasons. First, the eye-tracking devices were different. In experiment I, we used a Tobii Dynavox I-12+ (12.1-inch screen with Tobii IS4 eye tracker module), while in experiment II we used a Tobii Dynavox I-13+ (13.3-inch with Tobii IS5 eye tracker module). The differences in eye-tracking devices could lead to different input performances. Second, participants were allowed to have some uncorrected errors in Experiment II, to reduce the negative impact of the cumbersome error correction experience in Communicator 5 as explained in Section 5.2 Design and Procedure. More specifically, although participants were instructed to input as fast and accurately as they could, they were allowed to skip error correction if they felt it was difficult to do so. This instruction was different from the instruction in Experiment I, where participants were instructed to input "as fast and accurately as they could". The difference in instruction could result in a higher error rate of GlanceWriter in Experiment II.

Same as Experiment I, participants were asked to provide subjective ratings on mental demand, physical demand, comfort and learnability. Participants also rated their overall preference for each method. As shown in Figure 8, participants were in favor of GlanceWriter across all questions. A Wilcoxon Signed-Ranks Test indicated that the overall preference of GlanceWriter Keyboard was significantly higher than that of Communicator 5 ($Z = 1.86$, $p = 0.03$).

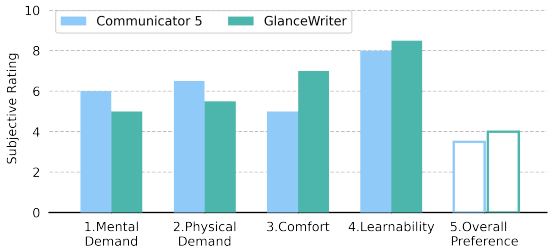


Figure 8: Medians of subjective ratings. For measures 1 and 2, a lower rating is better. For measures 3-5, a higher rating is better. GlanceWriter received more favorable ratings in all categories.

6 GENERAL DISCUSSION AND FUTURE WORK

The two user studies showed that GlanceWriter is a promising gaze-based text entry method. It outperformed EyeSwipe and the built-in keyboard of Communicator 5 in speed, error rate reduction and subjective ratings.

6.1 Input Speed

GlanceWriter has a higher speed than EyeSwipe because it uses a probabilistic decoder and does not require signaling the starting and ending positions of a gaze path. Our analysis of the study data in Experiment I showed that the mean (SD) duration of "reverse crossing" were 1.33 ± 0.17 (second) for the starting letter and 1.44 ± 0.19 (second) for the ending letter, which contributed to the slow speed of EyeSwipe. 11 out of 14 participants commented "To start entering and confirm a word was hard with EyeSwipe " (because of "reverse crossing") or similar answers to the open-ended question "What's the most difficult part in EyeSwipe?". In contrast, very few participants commented on the most difficult part of GlanceWriter. The average speed of EyeSwipe in our study was slower than the value reported in the previous work about EyeSwipe [26]. It was probably because in our study, each user spent only 20 minutes typing on EyeSwipe, while in the previous work, a user spent 4 sessions (10 minutes each) over two days. Additionally, different users were recruited in different studies. They may have different expertise in gaze input, text input, and different ages, which may contribute to the discrepancy in input speed between our study and the study reported in the previous work [25].

GlanceWriter is faster than the built-in keyboard in Communicator 5, which also uses a dwell-free text entry method. Two factors may contribute to the slower speed of Communicator 5. First, Communicator 5 took around 600 ms (default setting) to activate or deactivate the keyboard as the gaze enters or exits the keyboard area. These activation and deactivation processes slow users down on Communicator 5. One participant answered, "It took too much time to select text and keyboard" to the most difficult part of the open-end question. In contrast, on GlanceWriter, a user activated (or deactivated) the keyboard by entering (or exiting the keyboard) with gaze position. Second, it was difficult to correct word-level errors in Communicator 5. The first-level UI of Communicator 5 only supports deleting the entire sentence, not deleting a word within a sentence, as previously explained. We also examined the trials in which no correction occurred. The input speed was 10.21 WPM for GlanceWriter and 7.84 WPM for Communicator 5. GlanceWriter was still faster than Communicator 5 for trials where no correction occurred, indicating that the decoding algorithm also contributed to the superior performance of GlanceWriter.

6.2 Decoding Accuracy

The study results showed that the newly proposed decoding algorithm has higher decoding accuracy than the DTW-based algorithm in EyeSwipe: the error rate of GlanceWriter was lower than both EyeSwipe and Communicator 5. We calculated the total number of error correction actions performed by participants in each condition. Such error-correction actions were regarded as backspace or undo. In Experiment I, the number of total corrections was 277 (0.99 per trial) for EyeSwipe and 69 (0.25 per trial) for GlanceWriter. With a smaller number of error correction actions in GlanceWriter, it still had a lower error rate than EyeSwipe, indicating that GlanceWriter had higher decoding accuracy in EyeSwipe.

Experiment II showed that the overall error rate of GlanceWriter was lower than that of Communicator 5. It may be partially due to the smaller number of error correction actions in Communicator 5. The number of error correction actions was 13 (0.05 per trial) for Communicator 5 and 41 (0.17 per trial) for GlanceWriter in Experiment II. Users made a smaller number of error corrections in Communicator 5 mainly because its top-level interface supported sentence-level deletion only and it was difficult to correct word-level errors in Communicator 5, as previously described. We could not draw a definite conclusion about which algorithm was more

accurate because GlanceWriter had a lower error rate but it also had a higher number of error correction actions.

6.3 Subjective Ratings

The subjective ratings were also in favor of GlanceWriter over EyeSwipe and Communicator 5. GlanceWriter had less mental and physical demands, was more comfortable to use, and had higher learnability than both EyeSwipe and Communicator 5 (Figures 5 and 8). Users commented that it took extra time to activate and deactivate the keyboard on Communicator 5, and the inconvenience of correcting word-level errors made Communicator 5 hard to use and learn. Especially they commented that not including word-level error correction on the first-level UI made the error correction difficult to perform. Participants also commented that it required efforts to perform "reversal crossing" actions in EyeSwipe, which resulted in more mental and physical demands. In contrast, they felt GlanceWriter was easy to use and learn. Eliminating the need to specify starting and ending letters makes the input experience smooth. They could start entering a word by simply moving the gaze point into the keyboard and end the entering process by moving the gaze point away from the keyboard.

Gaze input in general requires high mental and physical demands. As shown in Figure 8, the medians of mental and physical demands for both GlanceWriter and GlanceWriter were all equal to or above 5 on a 1- 10 scale. The gaze position was difficult to control, partially due to the limitation of the gaze tracking device and the unintentional gaze movement during the interaction. Practicing gaze input requires high mental and physical demands, which was also the reason we provided sufficient practice time for each method before evaluation. Designers and researchers who would deploy gaze input methods in the real world should keep in mind that gaze input could increase mental and physical demands during the interaction.

6.4 Handling Instability of Gaze Input

One advantage of the decoding algorithm of GlanceWriter is that it uses a probabilistic approach for decoding: it uses probability to quantify the uncertainty of entering a letter. For example, if a letter is entered with stable gaze input, it should have a high stability score (Equation 2) which in turn increases its key score, indicating that the corresponding letter has a high probability of being included. If the gaze input is unstable (e.g., the gaze focus is jumping around), the stability and key scores will be low.

We also designed the gaze path canceling method in GlanceWriter to accommodate the potential instability of gaze input. To cancel a path in GlanceWriter, a user simply moves her gaze outside the keyboard. This approach is in contrast to the canceling methods in EyeSwipe [26] and Communicator 5, which require dwelling or "reverse crossing" actions for canceling a gaze path. Dwelling requires a precise selection of a button and reverse crossing requires drawing a specific gesture with the gaze path, both of which are more susceptible to the instability of gaze input than simply moving the gaze outside the keyboard region (GlanceWriter).

6.5 Future Work

We discovered that the stability of gaze input depends on the area on the screen. Participants commented that it was easier to glance over keys in the center of the screen than over keys near the edges. This finding is consistent with the finding of previous work [12]. It suggests that in the future we may need to take the location-depended accuracy into consideration in decoding. Additionally, we may consider using more powerful language models, such as GPT-3 [5], to boost the text entry performance as gaze input tends to be noisy.

As with many other gestural decoding algorithms (e.g., [24]), the algorithm of GlanceWriter was designed to decode words within a dictionary only. Out-of-vocabulary words (OOV) need to be entered by other methods. For example, on a touchscreen gesture keyboard, such words are often entered by tap typing. For gaze-based input, a user could revert to dwell-based methods for entering OOV words. It is worth investigating how to handle OOV words for gaze input in future research.

7 CONCLUSION

We present GlanceWriter, a gaze-based hands-free text entry method that allows entering text by glancing over letters on a virtual keyboard. One major advantage of GlanceWriter over existing gaze-based input methods (e.g., [26]) is that GlanceWriter eliminates the need to dwell on letters, or perform "reverse crossing" to specify the starting and ending positions of a gaze path for a word. Instead, GlanceWriter adopts a novel decoding algorithm to probabilistically determine which letter is entered by examining the dynamics of gaze movements and gaze locations. Our controlled experiment demonstrates that GlanceWriter achieves a 67% increase in typing speed over EyeSwipe with an error rate of only 2.71%. It also outperforms a commercially available gaze-based text entry method in Tobii Communicator 5, improving the typing speed by 28.7% and reducing the error rate to 12.89%. Our study suggests that glancing over letters for text entry is both feasible and promising, and our contribution lies in the novel decoding algorithm that enables this method.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their insightful comments and our user study participants. We thank Bridging Voice for providing the eye tracking device. This work was supported by DOD CDMRP award W81XWH2210408, NIH award R01EY030085, and NSF award 2113485.

REFERENCES

[1] Ouais Alsharif, Tom Ouyang, Françoise Beaufays, Shumin Zhai, Thomas Breuel, and Johan Schalkwyk. 2015. Long short term memory neural network for keyboard gesture decoding. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2076–2080.

[2] Xiaojun Bi, Shiri Azenkot, Kurt Partridge, and Shumin Zhai. 2013. Octopus: Evaluating Touchscreen Keyboard Correction and Recognition Algorithms via. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. ACM, New York, NY, USA, 543–552. https://doi.org/10.1145/2470654.2470732

[3] Xiaojun Bi, Ciprian Chelba, Tom Ouyang, Kurt Partridge, and Shumin Zhai. 2012. Bimanual gesture keyboard. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 137–146.

[4] Syed Masum Billah, Yu-Jung Ko, Vikas Ashok, Xiaojun Bi, and IV Ramakrishnan. 2019. Accessible gesture typing for non-visual text entry on smartphones. In

Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 1–12.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL]

[6] Wenzhe Cui, Jingjie Zheng, Blaine Lewis, Daniel Vogel, and Xiaojun Bi. 2019. Hotstrokes: Word-gesture shortcuts on a trackpad. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[7] Wenzhe Cui, Suwen Zhu, Zhi Li, Zheer Xu, Xing-Dong Yang, IV Ramakrishnan, and Xiaojun Bi. 2021. BackSwipe: Back-of-device Word-Gesture Interaction on Smartphones. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[8] Mark Davies. 2018. The corpus of contemporary American English: 1990-present.

[9] Antonio Diaz-Tula and Carlos H Morimoto. 2016. Augkey: Increasing foveal throughput in eye typing with augmented keys. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 3533–3544.

[10] Tobii dynavox. 2021. Communicator 5. https://www.tobiidynavox.com/software/windows-software/communicator-5/.

[11] Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th annual ACM symposium on user interface software & technology*. 457–466.

[12] Anna Maria Feit, Shane Williams, Arturo Toledo, Ann Paradiso, Harish Kulkarni, Shaun Kane, and Meredith Ringel Morris. 2017. *Toward Everyday Gaze Input: Accuracy and Precision of Eye Tracking and Implications for Design*. Association for Computing Machinery, New York, NY, USA, 1118–1130. https://doi.org/10.1145/3025453.3025599

[13] Wenxin Feng, Ming Chen, and Margrit Betke. 2014. Target reverse crossing: a selection method for camera-based mouse-replacement systems. In *Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments*. 1–4.

[14] Wenxin Feng, Jiangnan Zou, Andrew Kurauchi, Carlos H Morimoto, and Margrit Betke. 2021. Hgaze typing: Head-gesture assisted gaze typing. In *ACM Symposium on Eye Tracking Research and Applications*. 1–11.

[15] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 3817–3821.

[16] Aakar Gupta, Cheng Ji, Hui-Shyong Yeo, Aaron Quigley, and Daniel Vogel. 2019. Rotoswype: Word-gesture typing using a ring. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.

[17] Michael Xuelin Huang, Jiajia Li, Grace Ngai, and Hong Va Leong. 2017. Screenglint: Practical, in-situ gaze estimation on smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2546–2557.

[18] Anke Huckauf and Mario H Urbina. 2008. Gazing with pEYEs: towards a universal input for various applications. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. 51–54.

[19] Chi-Shin Hwang, Ho-Hsiu Weng, Li-Fen Wang, Chon-Haw Tsai, and Hao-Teng Chang. 2014. An eye-tracking assistive device improves the quality of life for ALS patients and reduces the caregivers' burden. *Journal of motor behavior* 46, 4 (2014), 233–238.

[20] Jalal Ismaili et al. 2017. Mobile learning as alternative to assistive technology devices for special needs students. *Education and Information Technologies* 22, 3 (2017), 883–899.

[21] Toshiya Isomoto, Toshiyuki Ando, Buntarou Shizuki, and Shin Takahashi. 2018. Dwell time reduction technique using Fitts' law for gaze-based target acquisition. In *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. 1–7.

[22] Robert JK Jacob. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 11–18.

[23] Liu Jigang, Bu Sung Lee Francis, and Deepu Rajan. 2019. Free-head appearance-based eye gaze estimation on mobile devices. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 232–237.

[24] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology* (Santa Fe, NM, USA) *(UIST '04)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1029632.1029640

[25] Chandan Kumar, Ramin Hedeshy, I Scott MacKenzie, and Steffen Staab. 2020. TAGSwipe: Touch Assisted Gaze Swipe for Text Entry. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12.

[26] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos Morimoto, and Margrit Betke. 2016. EyeSwipe: Dwell-free text entry using gaze paths. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1952–1956.

[27] Andrew Kurauchi, Wenxin Feng, Ajjen Joshi, Carlos H Morimoto, and Margrit Betke. 2020. Swipe&Switch: Text Entry Using Gaze Paths and Context Switching. In *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 84–86.

[28] Zhi Li, Maozheng Zhao, Yifan Wang, Sina Rashidian, Furqan Baig, Rui Liu, Wanyu Liu, Michel Beaudouin-Lafon, Brooke Ellison, Fusheng Wang, et al. 2021. BayesGaze: A Bayesian Approach to Eye-Gaze Based Target Selection. In *Graphics Interface 2021*.

[29] Yu-Hao Lin, Suwen Zhu, Yu-Jung Ko, Wenzhe Cui, and Xiaojun Bi. 2018. Why is gesture typing promising for older adults? comparing gesture and tap typing behavior of older with young adults. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*. 271–281.

[30] I. Scott MacKenzie. 2015. *A Note on Calculating Text Entry Speed*.

[31] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI EA '03)*. ACM, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[32] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. 2009. Fast gaze typing with an adjustable dwell time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 357–360.

[33] Päivi Majaranta and Kari-Jouko Räihä. 2007. Text entry by gaze: Utilizing eye-tracking. *Text entry systems: Mobility, accessibility, universality* (2007), 175–187.

[34] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: a mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1073–1082.

[35] Martez E. Mott, Shane Williams, J. Wobbrock, and M. Morris. 2017. Improving Dwell-Based Gaze Typing with Dynamic, Cascading Dwell Times. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (2017).

[36] Alexandra Papoutsaki, Patsorn Sangkloy, James Laskey, Nediyana Daskalova, Jeff Huang, and James Hays. 2016. WebGazer: Scalable Webcam Eye Tracking Using User Interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI, 3839–3845.

[37] Diogo Pedrosa, Maria Da Graça Pimentel, Amy Wright, and Khai N Truong. 2015. Filteryedping: Design challenges and user performance of dwell-free eye typing. *ACM Transactions on Accessible Computing (TACCESS)* 6, 1 (2015), 1–37.

[38] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. 2017. Gaze+ pinch interaction in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction*. 99–108.

[39] Daniel Rough, Keith Vertanen, and Per Ola Kristensson. 2014. An evaluation of Dasher with a high-performance language model as a gaze communication method. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. 169–176.

[40] S. Sarcar, P. Panwar, and T. Chakraborty. 2013. EyeK: an efficient dwell-free eye gaze-based text entry system. In *APCHI*.

[41] Immo Schuetz, T Scott Murdison, Kevin J MacKenzie, and Marina Zannoli. 2019. An Explanation of Fitts' Law-like Performance in Gaze-Based Selection Tasks Using a Psychophysics Approach. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.

[42] Korok Sengupta, Raphael Menges, Chandan Kumar, and Steffen Staab. 2017. Gazethekey: Interactive keys to integrate word predictions for gaze-based text entry. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion*. 121–124.

[43] Korok Sengupta, Raphael Menges, Chandan Kumar, and Steffen Staab. 2019. Impact of variable positioning of text prediction in gaze-based text entry. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. 1–9.

[44] Ludwig Sidenmark and Hans Gellersen. 2019. Eye&head: Synergetic eye and head movement for gaze pointing and selection. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 1161–1174.

[45] Adalberto L Simeone, Andreas Bulling, Jason Alexander, and Hans Gellersen. 2016. Three-point interaction: Combining bi-manual direct touch with gaze. In *Proceedings of the international working conference on advanced visual interfaces*. 168–175.

[46] Rosella Spataro, Maria Ciriacono, Cecilia Manno, and Vincenzo La Bella. 2014. The eye-tracking computer device for communication in amyotrophic lateral sclerosis. *Acta Neurologica Scandinavica* 130, 1 (2014), 40–45.

[47] Outi Tuisku, Päivi Majaranta, Poika Isokoski, and Kari-Jouko Räihä. 2008. Now Dasher! Dash away! Longitudinal study of fast text entry by eye gaze. In *Proceedings of the 2008 symposium on Eye tracking research & applications*. 19–26.

[48] Wikipedia contributors. 2021. Trie — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Trie&oldid=1038125595 [Online; accessed 31-August-2021].

[49] Erroll Wood, Tadas Baltrusaitis, Xucong Zhang, Yusuke Sugano, Peter Robinson, and Andreas Bulling. 2015. Rendering of eyes for eye-shape registration and

gaze estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 3756–3764.

[50] Erroll Wood and Andreas Bulling. 2014. Eyetab: Model-based gaze estimation on unmodified tablet computers. In *Proceedings of the Symposium on Eye Tracking Research and Applications*. 207–210.

[51] Hui-Shyong Yeo, Xiao-Shen Phang, Steven J Castellucci, Per Ola Kristensson, and Aaron Quigley. 2017. Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4194–4202.

[52] Xin Yi, Chun Yu, Weinan Shi, Xiaojun Bi, and Yuanchun Shi. 2017. Word Clarity As a Metric in Sampling Keyboard Test Sets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. ACM, New York, NY, USA, 4216–4228. https://doi.org/10.1145/3025453.3025701

[53] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. 2017. Tap, dwell or gesture? Exploring head-based text entry techniques for HMDs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 4479–4488.

[54] Shumin Zhai and Per-Ola Kristensson. 2003. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 97–104.

[55] Shumin Zhai and Per Ola Kristensson. 2012. The word-gesture keyboard: reimagining keyboard interaction. *Commun. ACM* 55, 9 (2012), 91–101.

[56] Shumin Zhai, Per Ola Kristensson, Pengjun Gong, Michael Greiner, Shilei Allen Peng, Liang Mico Liu, and Anthony Dunnigan. 2009. Shapewriter on the iPhone: from the laboratory to the real world. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*. 2667–2670.

[57] Xucong Zhang, Michael Xuelin Huang, Yusuke Sugano, and Andreas Bulling. 2018. Training person-specific gaze estimators from user interactions with multiple devices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[58] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 439, 13 pages. https://doi.org/10.1145/3173574.3174013

[59] Suwen Zhu, Jingjie Zheng, Shumin Zhai, and Xiaojun Bi. 2019. i'sFree: Eyes-Free Gesture Typing via a Touch-Enabled Remote Control. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–12.