Univ Access Inf Soc (2011) 10:69–80
DOI 10.1007/s10209-010-0188-6

LONG PAPER

BlinkWrite: efficient text entry using eye blinks

I. Scott MacKenzie · Behrooz Ashtiani

Published online: 13 May 2010

Abstract In this paper, a new text entry system is proposed, implemented, and evaluated. *BlinkWrite* provides a communication gateway for cognitively able motor-impaired individuals who cannot use a traditional eye-tracking system. In contrast to most hands-free systems, *BlinkWrite* allows text to be entered and corrected using a single input modality: blinks. The system was implemented using a scanning ambiguous keyboard, a new form of scanning keyboard that allows English text to be entered in less than two scanning intervals per character. In a user study, 12 participants entered text using the system with three settings for scanning interval: 1,000, 850, and 700 ms. An average text entry rate of 4.8 wpm was observed with accuracy >97%. The highest average text entry rate was achieved with the scanning interval of 850 ms.

Keywords Blink typing · Hands free text-entry · Eye typing · Scanning ambiguous keyboard · Assistive technologies

1 Introduction

From composing an email message to writing a novel, communication is central to the human experience. To most people, text entry is as simple as typing on a keyboard. To those suffering from physical disabilities, that same routine task may present a significant challenge. Severe disabilities such as amyotrophic lateral sclerosis (ALS), cerebral palsy (CP), or locked-in syndrome (LIS) often lead to complete loss of control over voluntary muscles, except the eye muscles, rendering the individual paralyzed and mute. Conventional physical interfaces, specialized switches, and voice recognition systems are not viable interaction solutions in these cases. The eyes, therefore, become an important input modality to connect persons with a severe motor impairment to the digital world, and through the digital world to the friends, colleagues, and loved ones with whom they wish to communicate.

1.1 Eye typing

Eye tracking is a technology that holds great promise for people with physical disabilities. Most eye-tracking interfaces, when used for computer input, are designed to simulate a standard pointing device such as a mouse. Based on this concept, controlled eye motion is an important part of these eye-tracking systems. There are two categories of such systems, based on how they utilize the eye movement as motion input: direct-selection systems and gesture-based systems.

Direct-selection systems translate the motion of the eyes directly to that of a mouse, facilitating movement of the cursor from point to point according to the user's shifting point of gaze. An on-screen keyboard is a typical example. "Eye typing" involves looking at the desired letter or key to highlight it, and dwelling on it for selection [21, 22]. Notable examples of such systems are *ERICA* [1], *GazeTalk* [6], *AUK* [24], *WiViK* [26], *SoftType* [11], and *HandsOFF*! [22]. Grouping the letters together in a

I. Scott MacKenzie (✉) · B. Ashtiani
Department of Computer Science and Engineering,
York University, 4700 Keele Street,
Toronto, ON M3J 1P3, Canada
e-mail: mack@cse.yorku.ca

B. Ashtiani
e-mail: abehrooz@cse.yorku.ca

hierarchical pie structure is studied in *pEYEwrite* [8]. The *GazeTalk* and *pEYE* interfaces are shown in Fig. 1. The *GazeTalk* interface in (a) uses a small number of large buttons. This configuration allows input in situations where there is imprecise eye control or imprecise tracking of eye movements. The system uses linguistic context to present the six most likely next letters, while using an additional key to access the entire alphabet. In the example, the user is dwelling on "I" with the dwell time shown in a progress bar. Hansen et al. [6] report a novice text rate of 4 wpm (words per minute) with a dwell time of 750 ms.

Huckauf and Urbina's *pEYEwrite* [8] uses a two-tier selection scheme with pie menus. To enter "A", for example, the user first dwells on the pie slice containing the desired group of letters, then dwells on the desired slice in a popup pie menu (Fig. 1b). Novice entry rates of 7.9 wpm were reported with a dwell time of 400 ms.

Gesture-based systems take a different approach. Entering a letter involves completing a path rather than a selecting a point. Perhaps such a process is best described as drawing a symbol using linear or non-linear movement of the eyes. The Minimal Device Independent Text Input Method (MDITIM) [9] adopts a region-based technique, with gestures emanating from the center of the screen to the north, south, east, or west edges of the screen. *Eye-S* [25] increased the number of regions, called hotspots, from five in MDITIM to nine, adding north-east, north-west, south-east, and south-west regions. Figure 2a shows Wobbrock et al.'s [31] *EyeWrite*, an eye typing system with gestures resembling hand-printed letters. Gestures are drawn by moving the eye cursor from one corner to another of a rectangular window. Ward et al.'s *Dasher* [30] utilizes a novel approach, called continuous gestures (see Fig. 2b). Letters of the alphabet are placed on the right side of the screen in distinct regions. Selection is performed by gazing in the direction of the desired letter, which causes a horizontal movement toward that region. A letter is typed when it crosses a vertical line on the screen. As letters get closer to the vertical selection line, dynamic zooming is performed for more accurate selection. A new alphabetic list appears on the right of the screen once a letter is typed. Corrections are possible by looking toward the center of the screen. This has the effect of reversing the previous movement by crossing the previous letter back over the vertical selection line.

While existing eye typing systems present a practical method of communication to some members of the disabled community, they fail to include the most severely motor impaired. Especially in the case of locked-in syndrome, movement of the eyes is limited and often unreliable for use with eye trackers. As a result, the previous eye typing systems are not a reliable alternative for persons of

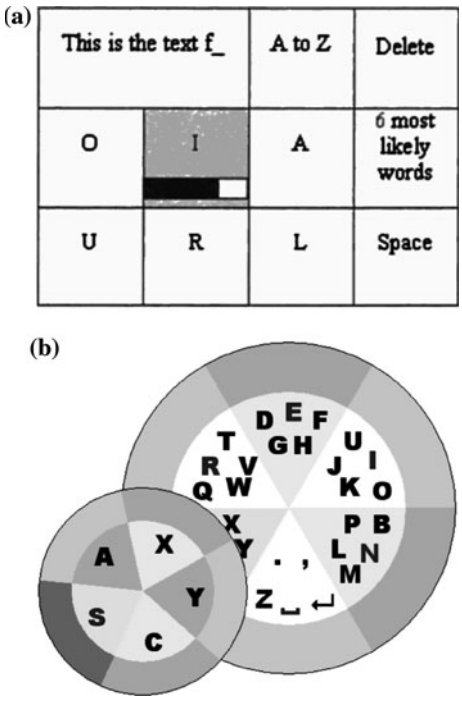

Fig. 1 Direct selection eye typing techniques **a** *GazeTalk* [6], **b** *pEyewrite* [8]

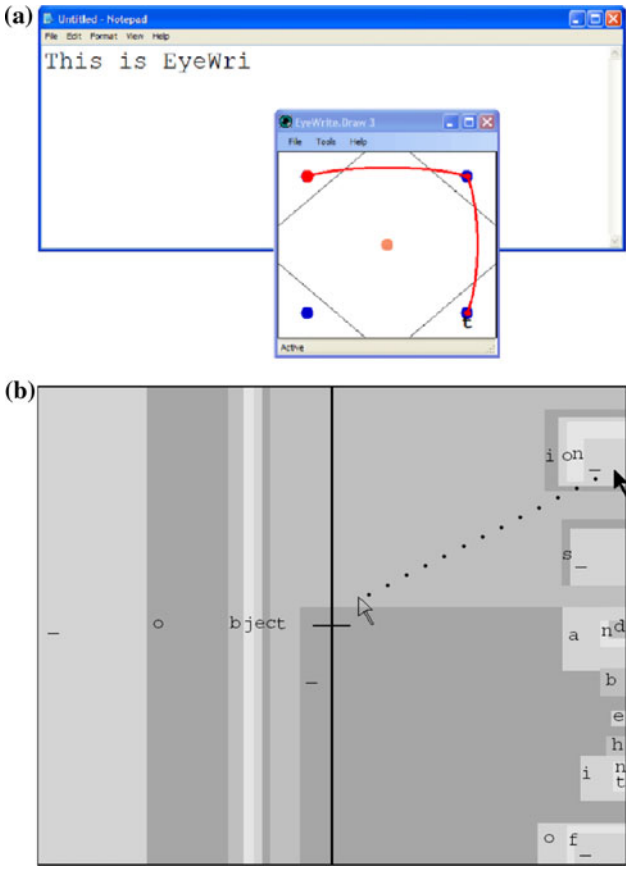

Fig. 2 Gesture-based systems: **a** *EyeWrite* [31], **b** *Dasher* [30]

Fig. 3 With the help of a transcriber, Bauby composes text by blinking his left eye (see text for discussion)

these disabilities, where blinking may be the only viable method of eye-based communication. With regards to interface design, the challenge is to accommodate text entry using only a single binary input: blinking.

1.2 Blink: an eye switch

The Diving Bell and the Butterfly is the memoir of Jean-Dominique Bauby, the editor in chief of *Elle* magazine, who entered a state of locked-in syndrome in 1995 after suffering a massive stroke [3]. Under complete physical paralysis, his only method of communication was through the blink of his left eye. Text entry involved a transcriber reciting a frequency-ordered alphabet to him, until he chose a letter by blinking, as shown in Fig. 3.[1] The entire book took some 200,000 blinks to compose using this method. A typical word took two minutes to write (i.e., text entry speed = 0.5 wpm). Not only is this prohibitively slow, the composition of text required two people.

It has been claimed that blinking is inappropriate as an input signal for eye-tracking systems, because it is unnatural and involves thinking about when to blink [10]. While that might hold true for users capable of alternative methods of interaction, blinking is an input method worth considering for users affected by a severe motor impairment.

Grauman et al. [5] used blinks as an input modality in a gaming system. They reported a blinking accuracy of about 96% with their *BlinkLink* system. This value was calculated based on the total number of hits and misses in a reflex and coordination game involving voluntary long blinks for target selection. To the authors' knowledge, there are no examples in the literature of text entry using blinks as the sole input modality; i.e., without tracking eye movement.

[1] http://en.wikipedia.org/wiki/Diving_bell_and_the_butterfly.

2 Scanning keyboards

Applications developed for single-key input typically employ switch-based scanning techniques, where the movement between options occurs automatically in timed intervals. Implementation of this technique for text entry results in a *scanning keyboard*.

Given the large number of letters and symbols in the English alphabet (and alphabets of other languages), there are significant challenges in implementing a scanning keyboard that supports a reasonable rate of text composition. Typical text entry rates are in the range of 3–5 wpm [2, 28]. Strategies for improving the performance of scanning keyboards are numerous and date to the 1970s. The most obvious strategy is to group letters together for multilevel selection. Variants of so-called "row-column scanning" are universally used in all scanning keyboards today. And there is little doubt they yield an increase in the rate of selecting characters for input (e.g., [2, 14, 29]).

The Microsoft *Windows XP* implementation of a row-column scanning keyboard is shown in Fig. 4. Scanning begins with the entire top row in focus. Focus advances row-by-row according to a preset scanning interval. The user presses SPACE during the scanning period in which the row containing the desired key has focus. In (a) scanning has reached the third row, the top line containing letters. To enter "T", the user presses SPACE then waits six more scan steps and then presses SPACE again (Fig. 4b). Clearly, this scenario is a significant improvement over a simple key-by-key scan.

Fig. 4 Scanning keyboard in Microsoft *Windows XP* **a** scanning is at the third row, **b** scanning is at the sixth key ("t") in the third row

Still, row-column scanning is slow. The most obvious improvement for row-column scanning is to rearrange letters with frequent letters near the beginning of the scan sequence, such as in the first row or in the first position in a column. There are dozens of research papers investigating this idea (e.g., [2, 11, 12, 14, 17, 29]). Example letter arrangements, as described by Koester and Levine [12] and Jones [11], are shown in Fig. 5. As seen, common letters in English are either in the top row or near the beginning of a subsequent row.

Researchers have investigated additional methods to speed up text entry with scanning keyboards. Dynamic techniques have been tried, whereby the position of letters varies depending on previous letters entered and the statistical properties of the language and previous text [14, 23].

Performance may also improve using a three-level selection scheme, known as block, group, or quadrant scanning [4, 16]. The idea is to scan through a block of items (perhaps a group of rows). The first selection enters a block. Scanning then proceeds among smaller blocks within the selected block. The second selection enters one of the smaller blocks and the third selection chooses an item within that block. Block scanning is most effective if there are a large number of selectable items.

Reducing the scanning interval will increase the text entry rate, but there is a cost: more errors or missed opportunities for selection. One approach is to dynamically adjust the system's scanning interval. Decisions to increase or decrease the scanning interval can be based on previous user performance, including text entry throughput, error rate, and reaction time [13, 15, 28].

Word or phrase prediction or completion is also widely used with scanning keyboards [11, 23, 26]. As a word is entered, the current word stem is expanded to form a list of matching complete words. The list is displayed in a dedicated region with a mechanism for the user to select the word early.

Overall, all scanning keyboards in use are variations on "divide and conquer". For any given entry, the first selection chooses a group of items and the next selection chooses within the group. The next section explores the prospect of combining the idea of a scanning keyboard with a text entry method commonly used on mobile phones.

(a)

_	E	A	R	D	U
T	N	S	F	W	B
O	H	C	P	V	J
I	M	Y	K	Q	
L	G	X	Z		

(b)

	_	T	I	H	C	B
E	A	N	D	F		
O	S	L	Y	V		
R	M	W	K	Q		
U	P	X	Z			
G	J					

Fig. 5 Examples of optimized letter arrangements for row–column scanning keyboards **a** Koester and Levine [12], **b** Jones [11] ("_" represents the SPACE character)

3 Scanning ambiguous keyboards (SAK)

The letter arrangement in the ubiquitous mobile phone keypad provides an interesting point of leverage for scanning keyboards. The phone keypad is *ambiguous* for text entry because each key bears three or four letters. With the help of a built-in dictionary, text entry is possible using one key press per letter as a word is entered. At the end of a word, occasionally there is more than one match ("word collisions") and, if so, extra key presses are required to retrieve the desired word from the alternatives. Although this sounds awkward, analyses show that about 95% of words in English can be entered unambiguously using a mobile phone keypad [27].

Recently, a new one-key text entry method has been proposed and demonstrated that combines the most demanding constraint of scanning keyboards—single-switch input—with the most appealing feature of ambiguous keyboards—one key press per character [18]. The result is a *scanning ambiguous keyboard* (SAK). The general idea is shown in Fig. 6. Letters are assigned over a small number of virtual keys in a letter-selection region. Scanning proceeds left to right and when the key bearing the desired letter is highlighted, it is selected. There is only one selection per letter and only one physical key or switch (not shown). With each selection, an ordered list of candidate words is presented based on the current keycodes. When the full word is entered (or earlier, see below), SPACE is selected whereupon scanning switches to the word-selection region. When the desired word is highlighted, it is selected and added to the text message with a terminating space. Scanning then returns to the letter-selection region for entry of the next word.

SAK designs include a few interesting interaction features. One is "cyclic scanning". Conventional scanning

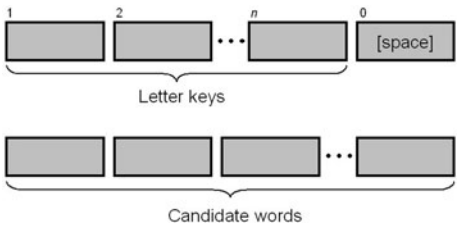

Fig. 6 A scanning ambiguous keyboard (*SAK*) combines a letter-selection region (*top* with a word-selection region (*bottom*)

keyboards restart scanning to a home position with each letter selection. With a SAK design, scanning is cyclic. After a letter selection, scanning continues to the next letter key and continues to cycle through the letter keys until SPACE is selected. When SPACE is selected, scanning switches to the word-selection region.

Another interaction feature is "early selection". If the desired word appears in the candidate list before a word is fully entered, then SPACE may be selected early to enter the word-selection region and select the word. Sometimes there is a performance benefit (i.e., fewer total scan steps), sometimes there is not. Clearly, issues such as a user's strategy and skill acquisition play a role.

SAK designs support "multiple selection". Multiple selections can occur in the same scanning interval if the highlighted key bears both the desired letter and one or more subsequent letters. For this feature to work effectively, SAK designs also use "timer restart on selection". When a selection is made, the scanning timer is restarted, thus making the entire scanning interval available again for a subsequent selection on the same key. The extra time is also useful even without multiple selections—to give the user a little extra time to think through the interactions necessary to enter a word.

Of course, there are many design issues. The first is simple: How many letter keys should be used and how should letters be assigned to keys? Clearly, SAK designs are compromises. Fewer letter keys reduce the scan steps in the letter selection region but increase ambiguity and, therefore, increase the scan steps in the word selection region. With more letter keys, the opposite effect occurs. As for ordering letters, there are two choices. One is to maximize the linguistic distance between keys by assigning letters in a way that minimizes word collisions. The other choice is to use an alphabetic ordering. An alphabetic ordering is easier for novice users, as it minimizes the cognitive load in finding the letters during entry. For the initial implementation of *BlinkWrite,* it was decided to use an alphabetic ordering.

In searching for an optimal design, *scan steps per character* (*SPC*) is a useful characteristic metric. *SPC* is the number of scan steps on average to enter each character of text in a given language. *SPC* includes both passive scan steps (no selection) and active scan steps (one or more selections) and includes the transition from the letter-selection region to the word-selection region (SPACE) and the scan steps to reach and select the desired word. Designs with lower *SPCs* are better, since they support a higher text entry throughput.

An algorithm to compute *SPC* requires a language corpus in the form of a word-frequency list. For this, Silfverberg et al.'s list of 9,022 unique words and frequencies drawn from the British National Corpus was used

Fig. 7 SAK alphabetically ordered letter arrangement yielding the lowest scan steps per character (*SPC* = 1.713)

[27]. An exhaustive run of the algorithm computed *SPC* for all designs from 1 to 6 virtual letter keys with all possible alphabetic letter assignments. The letter assignment with the lowest scan steps per character, *SPC* = 1.713, is shown in Fig. 7.[2]

SPC < 2 is remarkably low. As an example, the word "character" can be entered in just 11 scan steps:

```
c.r.a.tS..W
```

In the sequence above, a lowercase letter is a selection on the key bearing the letter. A period (".") is a passive scan. If multiple selection occurs, only the first letter is shown. In the example, "c" = "cha" and "a" = "ac". "S" is a selection on the virtual SPACE key. "W" is a word selection. Early selection is shown in the example, since not all letters were entered. Evidently, "character" appeared as the third word in the candidate list after "charact" (1113113) was entered ("..W"). For the example word, *SPC* = 1.10, computed as 11 scan steps divided by 10 characters ("character_").

The *SPC* calculation, whether for an example word or English in general, is a best case and assumes that users take all opportunities to optimize. Of course, missed opportunities can and will occur, depending on a variety of factors, such as the user's skill, motor ability, or the scanning interval. But missed opportunities are not errors; they simply slow the entry of words and phrases.

Two designs in the literature are similar to the SAK concept presented here; however, both involve more than one physical key. Harbusch and Kühn describe an ambiguous keyboard using "step scanning": one key advances the focus point while another makes selections [7]. Venkatagiri describes a scanning keyboard with three or four letters per virtual key [29]. Separate physical keys choose the desired letter once a key receives focus. Both papers just cited present models only. No user studies were performed, nor were the designs actually implemented.

The initial evaluation of SAK yielded text entry rates up to 9 wpm [18]. User input involved pressing a single physical key on the system's keyboard. Based on this promising result, it is only natural to adopt the SAK design concept for blink input.

[2] The design in Fig. 7 was reported with *SPC* = 1.834 in the original publication [18]; however, only "double selection" was used in the calculation. SAK designs supporting "multiple selections" yield a slightly lower *SPC*.

3.1 *BlinkWrite*: a blinking extension to SAK

BlinkWrite is an implementation of SAK that works with an eye-tracking apparatus using blinks for input. The *TM3* eye tracker from EyeTech Digital Systems was used (http://www.eyetechds.com/). The device is bundled with the *QuickGlance* software that provides control over eye tracking, data capture, system setup, and calibration. An additional API, *QuickLink*, allows customization of the interface and direct access to the control variables of *QuickGlance*. A background application was developed called *TrackerSetup* to enable blink clicking and disable eye-movement tracking. *TrackerSetup* also translates the *primary* and *secondary* blink signals, captured by the tracker, into simulated keyboard inputs "p" and "s", respectively. A primary signal is produced for blink durations between 200 and 500 ms. A secondary signal is produced for blinks longer than 500 ms. Blinks of lower than 200 ms are ignored and produce no input signal. This threshold is consistent with recommendations in previous research [5, 10] and helps to mitigate the "Midas touch problem", whereby unintended actions cause inadvertent selections in an interface.

3.1.1 *Error correction*

Error correction was implemented using the secondary signal (blink duration > 500 ms) as a delete operation. Both single-letter delete and word delete are supported. The algorithm, illustrated in Fig. 8, also includes a mechanism to avoid lengthy and frustrating sequences of single-letter deletes. Briefly, a delete operation removes either the last keystroke (letter), if input is in the middle of a word, or the last word, if input is between words. If a single-letter delete is followed immediately by another delete, the effect is to remove the rest of the keystrokes for the current word. The next delete operation would remove the preceding word, and so on.

3.1.2 *Auditory feedback*

In the absence of feedback, users of eye-tracking systems are left guessing whether the system captured their intended input correctly. Since input with *BlinkWrite* involves keeping the eyes closed for a period of time, visual feedback can be missed. Another problem arises when system commands are performed based on how long the eyes are closed. It is difficult for users to adapt to and learn the timing by practice. Auditory feedback is used in *BlinkWrite* in a manner that informs and attunes users of the exact time an operation is committed. For blinks generating a primary signal (letter selection), a "click" sound was used. For blinks generating a secondary signal (delete), "click" was

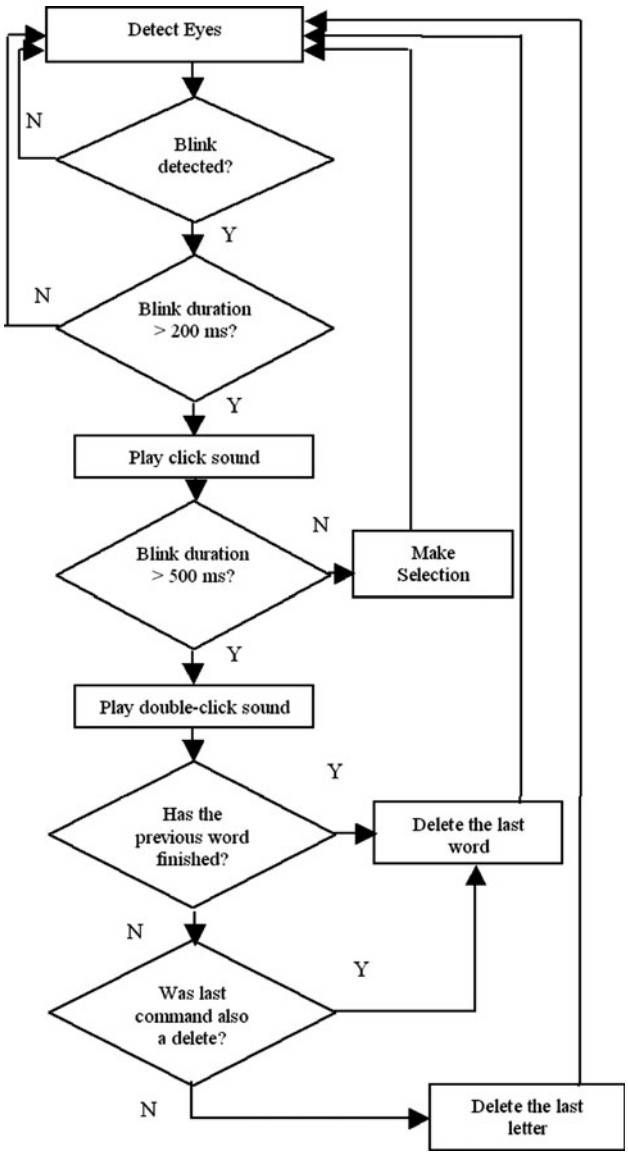

Fig. 8 Error correction mechanism in *BlinkWrite*

followed with "double-click". The timing of the auditory feedback with respect to the blink dynamics is illustrated in Fig. 8. For completeness, an involuntary blink of <200 ms duration is included in Fig. 9a.

A blink longer than 200 ms is either a letter-selection blink or a delete blink. The primary and secondary auditory stimuli are generated at the 200 and 500 ms points, respectively, rather than at the end of their respective blinks. These are the points in time where a blink is committed as being either a primary or a secondary signal. For example, if a letter-select blink was 350 ms in duration, the click sound is heard at the 200 ms point (see Fig. 8).

Of course, the system's internal primary signal is not generated until the blink ends at 350 ms, because it is not

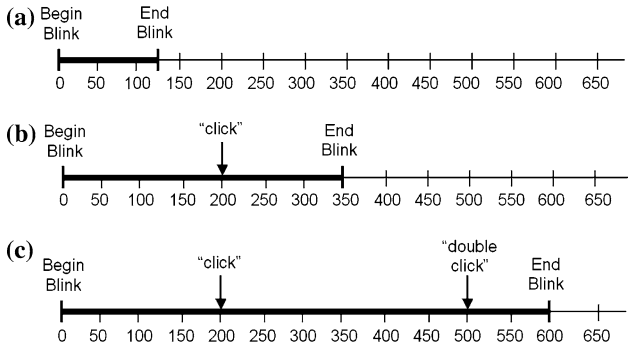

Fig. 9 Auditory feedback and blink durations. **a** Involuntary blink <200 ms (no feedback), **b** letter-select blink of duration 350 ms with auditory feedback at 200 ms, **c** delete blink of duration 600 ms with auditory feedback at 200 and 500 ms

yet known whether the current blink is for letter selection or delete. However, generating the click sound at 200 ms helps inform the user and build expectations on the operation of *BlinkWrite* (Fig. 8).

Figure 9c gives an example of a delete blink—a blink exceeding 500 ms in duration. Here, the blink is 600 ms in duration and there is both a click at 200 ms and a double click at 500 ms. Of course, the blink at 200 ms cannot be avoided, because the type of blink is not yet known. Informal testing suggests that this poses no problem for the user. The consistency of having distinct auditory signals at the 200 and 500 ms points is effective in allowing users to accurately learn and perform both letter-selection blinks (200–500 ms) and delete blinks (>500 ms).

3.1.3 BlinkWrite prototype software

The *BlinkWrite* experimental interface is shown in Fig. 10. At the top is the "presented text" field in which a phrase of text is presented for entry. Below that is the "transcribed text" field showing the participant's progress. The letter-selection region is next, followed by a field showing the current keystroke sequence. Below that is the word-selection region. In the example, the user is in the midst of entering the fifth word in the phrase "rent is paid at the beginning of the month". The first four words are in the transcribed text field. The keystrokes show 111222, the first six keystrokes for "beginning". The user has selected SPACE and is about to select "beginning", which is highlighted as the third word in the candidate list.

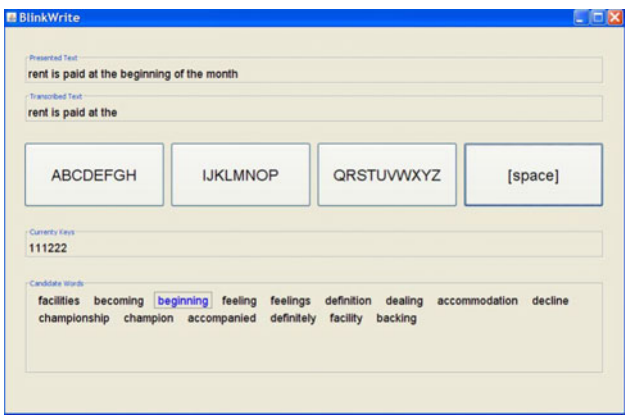

Fig. 10 *BlinkWrite's* graphical user interface

3.1.4 Candidate list size

The prospect of long candidate lists is clearly apparent in Fig. 10. Although the *SPC* calculation accounts for this, it does so only as an overall average for text entry in a given language. Although there are 15 words in the candidate list in Fig. 10, it is important to bear in mind that the list changes with every letter key selected. As selections are made, words either move off the list or move up in the list. For example, the last entry in the candidate list in Fig. 10 is the seven-letter word "backing". If the user wished to enter this word, it would be unwise to select SPACE after only six keystrokes (111222), as shown in the figure. With the final keystroke added (1112221; note that "g" = 1), the candidate list is recast as "feeling", "dealing", "decline", "backing", and a few other words. So, "backing" moves from position 15 to position 4 with one additional letter key selection. The nuances of "working the candidate list" are likely to evolve as a user becomes familiar with *BlinkWrite*. For high-frequency words, users will quickly learn the opportunities to minimize entry time. For less frequent words, more experience is necessary.

Despite the above, the questions arise of how long are the candidate lists, and how frequent are long candidate lists. The answers to these questions are determined by two factors: the letter-key assignments and the dictionary. For the 9,025-word dictionary used with the prototype SAK, 55.7% of the words are at the front of the candidate list when word selection begins. Furthermore, 82.2% of the words appear in the candidate list at position 4 or better; 94.8% are at position 10 or better, and 99.6% are at position 20 or better. So, very long candidate lists are rare. But, still, they occasionally occur. As an example of position 20, if a user wished to enter "alas" (1213), it would be at the end of a long list of candidates: {does, body, goes, boat, dogs, diet, flat, coat, andy, gift, flew, ends, aids, alex, clay, bias, blew, gods, dies, alas}.

3.1.5 Entering non-dictionary words

As with all ambiguous keyboards, SAK designs embed a frequency-ordered dictionary of words. Text entry is constrained to these words. Of course, a mechanism is needed

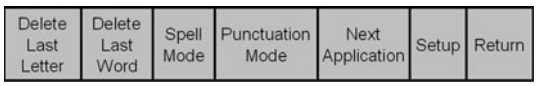

Fig. 11 Conduit for options accessed through a long press

to enter non-dictionary words and add these to the dictionary for subsequent input. There are a variety of ways to extend *BlinkWrite* to add additional features such as this—features needed for a full and proper implementation. Such features would include a "spell mode", as mechanisms to add punctuation symbols, changing configuration parameters, or switch to another application. Adding these features and maintaining the one-key constraint is a challenge, to say the least. One possibility is to use the long blink as a conduit. A long blink could still be used for error correction, as in *BlinkWrite*, but the design could be re-worked somewhat, such as popping up a small set of scanned virtual "option keys". An example (not implemented) is shown in Fig. 11.

3.1.6 Upper limit text entry speed

With a traditional Qwerty keyboard, touch-typing speed increases with practice. Furthermore, the increase continues indefinitely and is impeded only by the motor and cognitive abilities of the typist. This is not the case with scanning keyboards. The timing interval inherent with scanning keyboards places a strict upper limit on entry speed—a limit that is governed by the design of the keyboard, rather than a person's proficiency in using it. Given this, the issue arises of what is the upper limit for text entry speed using a scanning ambiguous keyboard such as *BlinkWrite*. Once the letter assignments and key layout are established, the answer to this question hinges primarily on two parameters: the scanning interval (*SI,* in milliseconds) and the scan steps per character (*SPC*) metric for the design. Using these parameters, the upper limit for text entry speed (*S*) is,

$$S = \left(\frac{1}{SPC \times SI}\right) \times \left(\frac{60000}{5}\right) \quad (1)$$

The first term yields a result in "characters per millisecond". The second term converts this to "words per minute", using the conventional definition of a word as "five characters".[3] As an example, if $SPC = 1.713$ (see Fig. 7) and $SI = 750$ ms,

$$S = \left(\frac{1}{1.713 \times 750}\right) \times \left(\frac{60000}{5}\right) = 9.34\,\text{wpm} \quad (2)$$

[3] It has been a convention dating back to the 1920s to define "word" in "words per minute" as five characters, including letters, punctuation, etc. [32].

In fact, this estimate is too generous. An additional adjustment is required to account for "timer restart on selection". With each selection in *BlinkWrite*, the scanning timer is restarted to facilitate multiple selections, as described earlier. The effect is to lengthen the average scanning interval for each character entered, and thereby lower the upper limit in text entry speed. To account for this in the calculation, two additional parameters are needed. One is the minimum blink time (*BT*) for selection, which is 200 ms for *BlinkWrite*. The other is the average number of *selections per scan step* (*SPS*) required for text entry in the language of interest (e.g., English). *SPS* is closely related to *SPC*. For the design in Fig. 7, $SPS = 0.627$. With these parameters added to Eq. 1, a more realistic upper limit for text entry speed is obtained:

$$S = \left(\frac{1}{SPC \times (SI + SPS \times BT)}\right) \times \left(\frac{60000}{5}\right) \quad (3)$$

Here, the scanning interval is adjusted upward to include the additional time due to selections. As an example, if $SPC = 1.713$, $SI = 750$ ms, $SPS = 0.627$, and $BT = 200$ ms,

$$S = \left(\frac{1}{1.713 \times (750 + 0.627 \times 200)}\right) \times \left(\frac{60000}{5}\right) = 8.00\,\text{wpm} \quad (4)$$

This calculation assumes that each selection (blink) is of the minimum duration allowable (i.e., 200 ms) and that it occurs at the earliest opportunity (e.g., at the beginning of scanning interval). These behaviors are a "best case" scenario and are unlikely to occur in practice. However, they are reasonable behaviors if the goal is to calculate an upper limit for text entry speed. In summary, users of *BlinkWrite* entering English text with the design described herein and a scanning interval of 750 ms will have an upper limit text entry speed of about 8 words per minute. Of course, it remains to be seen what text entry speed will be attained in practice. To determine this, an evaluation with users is necessary.

4 Methodology

4.1 Participants

Twelve volunteer participants ranging in age from 23 to 31 (*mean* = 26, SD = 2.4) were recruited. Four were female, eight male. All were daily users of computers, reporting 4–10 h of usage per day (*mean* = 6.8, SD = 1.8). Four used corrective lenses (3 male, 1 female). Four (males) had prior experience using an eye tracker.

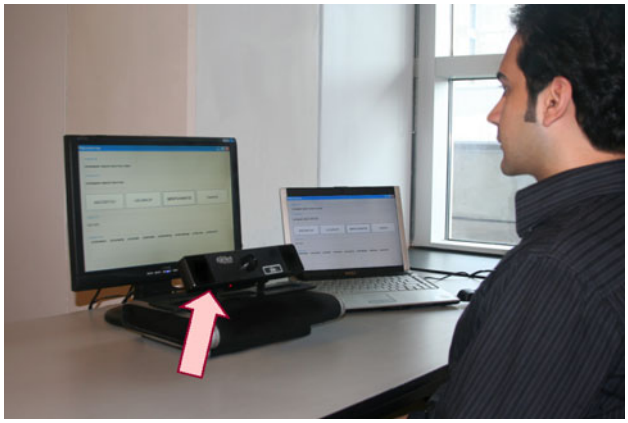

Fig. 12 Experimental apparatus used for *BlinkWrite* evaluation. The EyeTech Digital Systems *TM3* is identified by the *block arrow*

4.2 Apparatus

The EyeTech Digital Systems *TM3*[4] eye tracker (described above) was connected to a standard Microsoft *Windows XP* notebook with a 2.0 GHz Intel Core2 duo CPU and 4 GB of memory. The *BlinkWrite* application was displayed on an external 15 in. LCD monitor. The eye tracker and monitor were placed on a white box to achieve an optimal angle for eye detection. The setup is shown in Fig. 12.

4.3 Procedure

Prior to collecting data, the experimenter explained the task and demonstrated the software. Participants were instructed on the method of text entry, early word selection, error correction, and the audio feedback. They were instructed to enter the given phrases as quickly and accurately as possible and make corrections only if an error is detected in the current or previous word.

Participants were asked to adjust the height of the chair and their distance from the display to positions deemed comfortable to them. The angle and distance of the eye tracker were adjusted accordingly, to achieve an optimal lock on their eyes, evident by a crosshair displayed inside the image of the eyes in *QuickGlance* software.

Participants were allowed to enter a few trial phrases to become familiar with the selection and correction methods and the corresponding audio feedback. For the experiment, data collection began with the first selection blink for each phrase and ended upon completion of that phrase. The phrases were chosen randomly from a standard set of 500 phrases used for evaluating text entry systems [20]. Phrase lengths ranged from 16 characters to 43 characters (*mean* = 28.6). A separate scan steps per character analysis was done on the phrase set. The result was *SPC* = 1.850. So, on average, the phrases were slightly more complicated than English in general (as already mentioned, the *BlinkWrite* System has *SPC* = 1.713; see Fig. 7).

Although this point might seem minor, the phrases used in evaluating systems like *BlinkWrite* play a critical role. For example, it would be relatively easy to concoct phrases using words with low *SPC* values, and thereby obtain artificially inflated results. This was not done. If anything, using a phrase set with a higher *SPC* than English will yield results on the conservative side.

Participants were allowed to rest at their discretion between phrases. The *QuickGlance* software was checked after each break to ensure proper blink detection; adjustments were applied if necessary. After completing the experiment, participants were given a questionnaire soliciting demographic information (results cited above) and their impressions of the most suitable scanning interval setting (discussed later).

4.4 Design

The experiment included a single independent variable, scanning interval, with three levels: 700, 850, and 1,000 ms. As it was important to determine an optimal scanning interval for initial use of *BlinkWrite*, counterbalancing was used in assigning the three levels to participants. To offset learning effects, two participants were assigned to each of the six possible orders for the three conditions. Values for the scanning interval were chosen based on pilot testing and on the results in the initial SAK evaluation [18]. For each scanning interval condition, participants entered five phrases of text, yielding a total of 12 participants × 3 scanning intervals × 5 phrases = 180 phrases of input.

5 Results and discussion

5.1 Speed

The grand mean for entry speed was 4.37 wpm. This alone is a promising result, given that entry involves only eye blinks. Of the three scanning intervals tested, the nominal setting of 850 ms was fastest at 4.58 wpm. The other two scanning intervals were slightly slower: 4.37 wpm at 700 ms and 4.17 wpm at 1,000 ms (see Fig. 13). The effect of scanning interval on entry speed was not significant, however ($F_{2,22} = 2.05$, $p > 0.05$).

Figure 13 also includes the upper limit for text entry speed at each scanning interval. The values were calculated using Eq. 3 with *SPC* = 1.850, which corresponds to the phrases used in the evaluation (noted above). Although the upper limit increases with a decreasing scanning interval,

[4] http://www.eyetechds.com/.

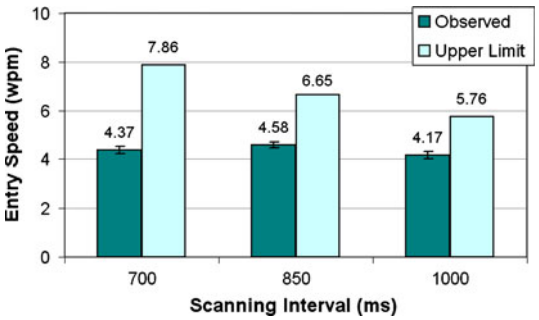

Fig. 13 Text entry speed by scanning interval. *Error bars* show ±1 SE

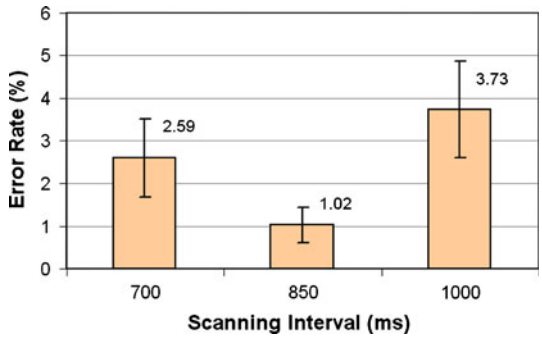

Fig. 14 Error rate by scanning interval. *Error bars* show ±1 SE

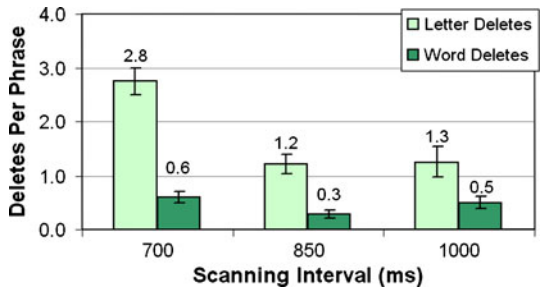

Fig. 15 Letter and word deletions per phrase by scanning interval. *Error bars* show ±1 SE

participants failed to produce a corresponding increase in text entry speed. However, with a scanning interval of 850 ms, participants' text entry speed, at 4.58 wpm, was 68.9% of the upper limit speed of 6.65 wpm. This is considered as a promising result, particularly in view of the fact the each participant entered only five phrases for each scanning interval. No doubt, with continued practice, entry speeds would increase toward the upper limit.

5.2 Accuracy

The accuracy in participants' responses is revealed in two ways: (1) the errors in the transcribed text and (2) the errors committed but corrected. Figure 14 shows the errors rates in the transcribed text. The measures were computed using the minimum string distance (MSD) method of comparing the presented text string with the transcribed text string [19]. The error rates are quite low overall, as participants generally chose to correct errors during entry. In fact, 81.7% of all phrases had an error rate of 0%. Typically, if any errors were left in the transcribed text, it was because an incorrect word was selected early in the phrase or the participant simply did not look for or did not notice the error. Consequently, those transcribed text phrases that contained errors, tended to have a large error rate (e.g., 10–20%). This also explains the relatively large error bars in Fig. 14 compared to Fig. 13. It is no surprise, then, that the effect of scanning interval on error was not statistically significant ($F_{2,22} = 1.89$, $p > 0.05$).

Errors committed but corrected are reflected in the incidence of elongated blinks (>500 ms) for letter delete or word delete. The results are shown in Fig. 15. As with entry speed and error rates, the best performance occurred with a scanning interval of 850 ms, with 1.2 letter deletes and 0.3 word deletes per phase (keeping in mind that the mean phrase length was 28.6 characters, equivalent to 5.7 words). Delete blinks were twice as prevalent with a scanning interval of 700 ms.

5.3 Multiple selections

The ability to select a key multiple times in a single scanning interval is an important interaction feature of *BlinkWrite*. An analysis of the raw data revealed that approximately 3.5 multiple selections were attempted per phrase. The highest number of multiple selection attempts was made with the low scanning interval at approximately 3.7 attempts per phrase. The number of attempts was reduced by 5% for the medium and 11% for the fast scanning intervals.

5.4 Optimal scanning interval

It seems the three levels chosen for the scanning intervals were appropriate in the sense that they bracketed user behavior. Performance overall was best with a scanning interval of 850 ms: entry speed was fastest, error rate was lowest, and letter deletes and word deletes were lowest. Participants' performance suffered at the faster (700 ms) and slower (1,000 ms) scanning intervals. The performed observations clearly point to a nominal value of about 850 ms as being preferred for initial use of *BlinkWrite*.

Of course with practice, the situation would change. Simpson and Koester, for example, used an adaptive scanning interval [28]. As expertise developed, the scanning interval was incrementally reduced according to user performance. Optimal performance was achieved at 525 ms. With practice, users of *BlinkWrite* could, no doubt, work

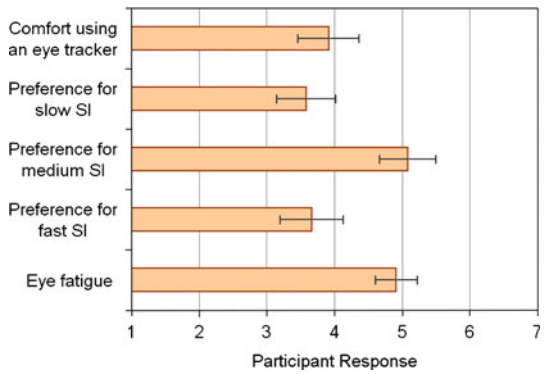

Fig. 16 Results of the questionnaire for comfort and preference. A response of 7 is most favorable, and 1 least favorable. *Error bars* ±1 SE

effectively with a faster scanning interval and thereby achieve higher text entry throughput. Of course, the scanning interval with *BlinkWrite* is slightly different than with typical scanning keyboards. *BlinkWrite* selections occur through blinks of duration 200–500 ms. With each blink, the scanning timer is restarted, making the full duration of the scanning interval available again for a subsequent selection in the same scan if the next letter is on the same key. So, faster scanning intervals do not bring the same sense of "urgency" with *BlinkWrite* as they do with traditional scanning keyboards. Nevertheless, while faster scanning intervals increase text entry throughput with traditional scanning keyboards, the connection is less direct with *BlinkWrite*.

5.5 Questionnaire

The questionnaire solicited responses on participants' level of comfort in using the eye tracker, eye fatigue, and preference for the scanning interval settings. Responses were rated on a 7-point Likert scale, where 1 is the least favorable response and 7 is the most favorable. The results are shown in Fig. 16.

A total of 92% of participants rated the medium scanning interval of 700 ms higher than the other two settings. Users mentioned that the slow scanning interval setting was fatiguing, especially when the word to be selected was significantly far along on the candidate list. They felt this forced them to hold off on blinking until the target was reached.

Overall, 75% of the users described a higher than average eye fatigue during the experiment. The comfort level in using the eye tracker was rated slightly below the mid-point score of 4.

6 Conclusions

This paper has presented a new hands-free text entry system using blinks as the only source of input. *BlinkWrite* was implemented using the core concepts of a scanning ambiguous keyboard [18]. Error correction schemes such as letter deletion and word deletion were included, as was auditory feedback to inform users of the type of command issued by their blinks. A user study was conducted to assess the text entry speed of the system based on the scanning interval. The best performance was with a scanning interval of 850 ms, which yielded a text entry speed of 4.58 wpm and error rates of 1.02%.

BlinkWrite is designed to assist the severely motor impaired individuals who are unable to create motion input, but are able to voluntarily blink their eyes. It is planned to test *BlinkWrite* with target users in a future study. Further study is needed to more thoroughly assess user fatigue with eye blinks in prolonged sessions, as well as to clinically assess the ability to control eye blinks for people with, for example, late stage ALS. For the *BlinkWrite* implementation, work is planned to improve the method of selecting candidates, to substitute a web cam for the eye tracker, and to implement new input methods using, for example, longer blinks or a re-organized word list.

Acknowledgments The authors wish to thank the reviewers for thoughtful and helpful comments on an earlier draft of this manuscript. We thank EyeTech Digital Systems for the loan of a *TM3* eye-tracking system. This research is sponsored by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Ashmore, M., Duchowski, A. T., Shoemaker, G.: Efficient eye pointing with a fisheye lens. Proceedings of Graphics Interface 2005, pp. 203–210. Canadian Information Processing Society, Toronto (2005)
2. Baljko, M., Tam, A.: Indirect text entry using one or two keys. Proceedings of the ACM SIGACCESS Conference on Assistive Technologies—ASSETS 2006, pp. 18–25. ACM, New York (2006)
3. Bauby, J.-D.: The diving bell and the butterfly. Knopf, New York (1997)
4. Bhattacharya, S., Samanta, D., Basu, A.: User errors on scanning keyboards: empirical study, model and design principles. Interact. Comput. **20**, 406–418 (2008)
5. Grauman, K., Betke, M., Lombardi, J., Gips, J., Bradski, G.R.: Communication via eye blinks and eyebrow raises: video-based human-computer interfaces. Univ Access Inform Soc **2**, 359–373 (2003)
6. Hansen, J. P., Hansen, D. W., Johansen, A. S.: Bringing gaze-based interaction back to basics. Proceedings of HCI International 2001, pp. 325–328. Erlbaum, Mahwah, NJ (2001)
7. Harbusch, K., Kühn, M.: An evaluation study of two-button scanning with ambiguous keyboards. Proceedings of the 7th European Conference for the Advancement of Assistive Technology—AAATE 2003, pp. 954–958. AAATE c/o Danish Centre for Assistive Technology, Taastrup, Denmark (2003)
8. Huckauf, A., Urbina, M. H.: Gazing with pEYEs: towards a universal input for various applications. Proceedings of the ACM Symposium on Eye Tracking Research and Applications—ETRA 2008, pp. 51–54. ACM, New York (2008)

9. Isokoski, P., Raisamo, R.: Device independent text input: a rationale and an example. Proceedings of the Working Conference on Advanced Visual Interfaces—AVI 2000, pp. 76–83. ACM, New York (2000)
10. Jacob, R. J. K.: What you look at is what you get: Eye movement-based interaction techniques. Proceedings of the ACM Conference on Human Factors in Computing Systems—CHI '90, pp. 11–18. ACM, New York (1990)
11. Jones, P. E.: Virtual keyboard with scanning and augmented by prediction. Proceedings of the 2nd European Conference on Disability, Virtual Reality and Associated Technologies, pp. 45–51. University of Reading, UK (1998)
12. Koester, H.H., Levine, S.P.: Learning and performance of able-bodied individuals using scanning systems with and without word prediction. Assist. Technol. **6**, 42–53 (1994)
13. Lesher, G., Higginbotham, D. J., Moulton, B. J.: Techniques for automatically updating scanning delays, Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America—RESNA 2000, pp. 85–87. RESNA, Arlingtion, VA (2000)
14. Lesher, G., Moulton, B., Higginbotham, D.J.: Techniques for augmenting scanning communication. Augment. Altern. Commun. **14**, 81–101 (1998)
15. Lesher, G., Moulton, B., Higginbotham, J., Brenna, A.: Acquisition of scanning skills: the use of an adaptive scanning delay algorithm across four scanning displays. Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America—RESNA 2002, pp. 75–77. RESNA, Arlington, VA (2002)
16. Lin, Y.-L., Chen, M.-C., Wu, Y.-P., Yeh, Y.-M., Wang, H.-P.: A flexible on-screen keyboard: Dynamically adapting for individuals needs. Universal Access in Human-Computer Interaction. Applications and Services, pp. 371–379. Springer, Berlin (2007)
17. Lin, Y.-L., Wu, T.-F., Chen, M.-C., Yeh, Y.-M., Wang, H.-P.: Designing a scanning on-screen keyboard for people with severe motor disabilities. Computers Helping People With Special Needs, pp. 1184–1187. Springer, Berlin (2008)
18. MacKenzie, I. S.: The one-key challenge: Searching for an efficient one-key text entry method, Proceedings of the ACM Conference on Computers and Accessibility—ASSETS 2009, pp. 91–98. ACM, New York (2009)
19. MacKenzie, I. S., Soukoreff, R. W.: A character-level error analysis technique for evaluating text entry methods, Proceedings of the Second Nordic Conference on Human-Computer Interaction—NordiCHI 2002, pp. 241–244. ACM, New York (2002)
20. MacKenzie, I. S., Soukoreff, R. W.: Phrase sets for evaluating text entry techniques. Extended Abstracts of the ACM Conference on Human Factors in Computing Systems—CHI 2003, pp. 754–755. ACM, New York (2003)
21. MacKenzie, I. S., Zhang, X.: Eye typing using word and letter prediction and a fixation algorithm. Proceedings of the ACM Symposium on Eye Tracking Research and Applications—ETRA 2008, pp. 55–58. ACM, New York (2008)
22. Majaranta, P., Räiha, K.-J.: Twenty years of eye typing: systems and design issues. Proceedings of the ACM Symposium on Eye Tracking Research and Applications—ETRA 2002, pp. 15–22. ACM, New York (2002)
23. Miró, J. and Bernabeu, P. A.: Text entry system based on a minimal scan matrix for severely physically handicapped people. Proceedings of the 11th Conference on Computers Helping People with Special Needs—ICCHP 2008, pp. 1216–1219. Springer, Berlin (2008)
24. Mourouzis, A., Boutsakis, E., Ntoa, S., Antona, M., Stephanidis, C.: An accessible and usable soft keyboard. Proceedings of HCI International 2007, pp. 961–970. Berlin: Springer (2007)
25. Porta, M., Turina, M., Eye, S.: A full-screen input modality for pure eye-based communication. Proceedings of the ACM Symposium on Eye Tracking Research and Applications—ETRA 2008, pp. 27–34. ACM, New York (2008)
26. Shein, F., Hamann, G., Brownlow, N., Treviranus, J., Milner, D., Parnes, P.: WiVik: A visual keyboard for Windows 3.0. Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America—RESNA 1991, pp. 160–162. RESNA, Arlington, VA (1991)
27. Silfverberg, M., MacKenzie, I. S., Korhonen, P.: Predicting text entry speed on mobile phones. Proceedings of the ACM Conference on Human Factors in Computing Systems—CHI 2000, pp. 9–16. ACM, New York (2000)
28. Simpson, R.C., Koester, H.H.: Adaptive one-switch row-column scanning. IEEE Trans. Rehab. Eng. **7**, 464–473 (1999)
29. Venkatagiri, H.S.: Efficient keyboard layouts for sequential access in augmentative and alternative communication. Augment. Altern. Commun. **15**, 126–134 (1999)
30. Ward, D. J., Blackwell, A. F., and MacKay, D. J. C.: Dasher: a data entry interface using continuous gestures and language models. Proceedings of the ACM Symposium on User Interface Software and Technology—UIST 2000, pp. 129–137. ACM, New York (2000)
31. Wobbrock, J. O., Rubinstein, J., Sawyer, M. W., Duchowski, A. T.: Longitudinal evaluation of discrete consecutive gaze gestures for text entry. Proceedings of the ACM Symposium on Eye Tracking Research and Applications—ETRA 2008, pp. 11–19. ACM, New York (2008)
32. Yamada, H.: A historical study of typewriters and typing methods: from the position of planning Japanese parallels. J. Inform. Process. **2**, 175–202 (1980)