SAK: Scanning Ambiguous Keyboard for Efficient One-Key Text Entry

11

I. SCOTT MACKENZIE
York University
and
TORSTEN FELZER
Darmstadt University of Technology

The design and evaluation of a scanning ambiguous keyboard (SAK) is presented. SAK combines the most demanding requirement of a scanning keyboard—input using one key or switch—with the most appealing feature of an ambiguous keyboard—one key press per letter. The optimal design requires just 1.713 scan steps per character for English text entry. In a provisional evaluation, 12 able-bodied participants each entered 5 blocks of text with the scanning interval decreasing from 1100 ms initially to 700 ms at the end. The average text entry rate in the 5th block was 5.11 wpm with 99% accuracy. One participant performed an additional five blocks of trials and reached an average speed of 9.28 wpm on the 10th block. Afterwards, the usefulness of the approach for persons with severe physical disabilities was shown in a case study with a software implementation of the idea explicitly adapted for that target community.

Categories and Subject Descriptors: H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies (e.g., mouse, touchscreen)*; K.4.2 [**Computer and Society**]: Social Issues—*Assistive technologies for persons with disabilities*; K.8.1 [**Personal Computing**]: Application Packages—*Word processing*

General Terms: Design, Experimentation, Human Factors

Additional Key Words and Phrases: Text entry, keyboards, ambiguous keyboards, scanning keyboards, mobile computing, intentional muscle contractions, assistive technologies

ACM Reference Format:

MacKenzie, I. S. and Felzer, T. 2010. SAK: Scanning ambiguous keyboard for efficient one-key text entry. ACM Trans. Comput.-Hum. Interact. 17, 3, Article 11 (July 2010), 39 pages.
DOI = 10.1145/1806923.1806925 http://doi.acm.org/10.1145/1806923.1806925

This research is sponsored by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Deutsche Forschungsgemeinschaft (DFG grant FE 936/3-2).
Author's address: I. S. MacKenzie, York University; email: mack@cse.yorku.ca.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2010 ACM 1073-0516/2010/07-ART11 $10.00
DOI 10.1145/1806923.1806925 http://doi.acm.org/10.1145/1806923.1806925

1. INTRODUCTION

A core human need is to communicate. Whether through speech, text, sound, or even body movement, our ability to communicate by transmitting and receiving information is highly linked to our quality of life, social fabric, and other facets of the human condition. Text, written text, is one of the most common forms of communication. Today the Internet, mobile phones, computers, and other electronic communicating devices provide the means for hundreds of millions of people to communicate daily using written text. The text in their correspondences is typically created using a keyboard and viewed on a screen.

While the ubiquitous Qwerty keyboard is the standard for desktop and laptop computers, many people use other devices for text entry. People "on the move" are bound to their mobile phones and use them frequently to transmit and receive SMS text messages, or even email. Of the estimated three billion or more text messages sent each day,[1] most are entered using the conventional mobile phone keypad. The mobile phone keypad is an example of an "ambiguous keyboard" because multiple letters are grouped on each key. Nevertheless, with the help of some sophisticated software and a built-in dictionary, entry is possible using one keystroke per character.

People with severe disabilities are often impaired in their ability to communicate, for example, using speech. For these individuals, there is a long history of supplanting speech communication with augmentative and alternative communication (AAC) aids. Depending on the disability and the form of communication desired, the aids often involve a sophisticated combining of symbols, phonemes, and other communication elements to form words or phrases [Brandenberg and Vanderheiden 1988; Hochstein and McDaniel 2004]. Computer access for users with a motor impairment may preclude use of a physical keyboard, such as a Qwerty keyboard or a mobile phone keypad, or operation of a computer mouse. In such cases, the user may be constrained to providing computer input using a single button, key, or switch. To the extent there is a keyboard, it is likely a visual representation on a display, rather than a set of physical keys. Letters, or groups of letters, on the visual keyboard are highlighted sequentially ("scanned") with entry using a series of selections—all using just one switch—to narrow in on and select the desired letter. This is the essence of a "scanning keyboard."

In this article, we present SAK, a scanning ambiguous keyboard. SAK combines the best of scanning keyboards and ambiguous keyboards. SAK designs use just a single key or switch (like scanning keyboards) and allow text to be entered using a single selection per character (like ambiguous keyboards). The result is an efficient one-key text entry method.

The rest of this article is organized as follows. First, we summarize the operation of scanning keyboards and ambiguous keyboards. Then, we detail the SAK concept, giving the central operating details. Following this, a model is developed to explore the SAK design space. The model allows alternative designs

[1]According to http://gsmworld.com/, there was a worldwide total of one trillion (10^{12}) SMS messages sent in 2005.

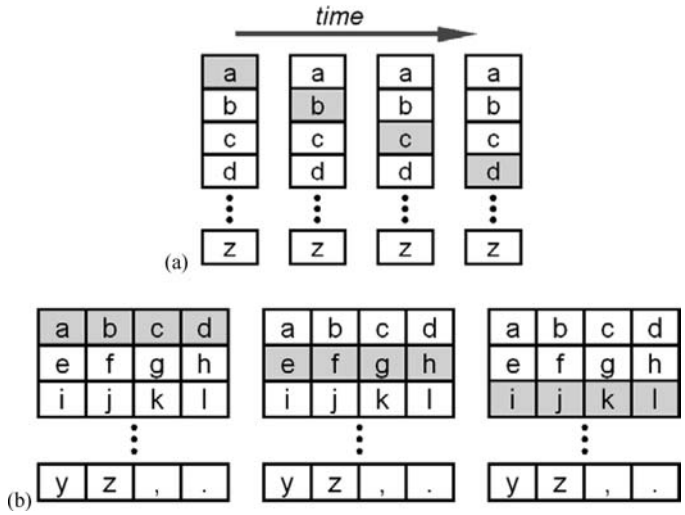

Fig. 1. Scanning keyboard concept. (a) linear scanning (b) row-column scanning.

to be quantified and compared. With this information we present the initial SAK design chosen for "test of concept" evaluation. Our methodology is described with results given and discussed. Following this, a full implementation is presented along with a case study of its use with a member of the target community. Concluding remarks summarize the contribution and identify issues for further research.

1.1 Scanning Keyboards

The idea behind scanning keyboards is to combine a visual keyboard with a single key, button, or switch for input. The keyboard is divided into selectable regions which are sequentially highlighted, or scanned. Scanning is typically automatic, controlled by a software timer, but self-paced scanning is also possible. In this case, the highlighted region is advanced as triggered by an explicit user action [Felzer et al. 2008].

When the region containing the desired character is highlighted, the user selects it by activating the input switch. The general idea is shown in Figure 1(a). Obviously, the rate of scanning is important, since the user must press and release the switch within the scanning interval for a highlighted region. Scan step intervals in the literature range from 0.3 seconds [Miró and Bernabeu 2008] to about 5 seconds.

Regardless of the scanning interval, the linear scanning sequence in Figure 1(a) is slow. For example, it takes 26 scan steps to reach "z". To address this, scanning keyboards typically employ some form of multilevel, or divide-and-conquer, scanning. Figure 1(b) shows row-column scanning. Scanning proceeds row-to-row. When the row containing the desired letter is highlighted, it is selected. Scanning then enters the row and proceeds left to right within the row. When the desired letter is highlighted, it is selected. Clearly, this is an improvement. The letter "j," for example, is selected in 5 scan steps in

Figure 1(b): 3 row scans + 2 column scans. Bear in mind that while row-column scanning reduces the scan steps, it increases the number of motor responses. The latter may be a limiting factor for some users.

When a letter is selected, scanning reverts to the home position to begin the next scan sequence, row to row, and so on. This behavior is a necessary by-product of row-column selection, since continuing the scan at the next letter negates the performance advantage of two-tier selection.

Row-column scanning is also slow. To speed up entry, numerous techniques have been investigated. These fall into several categories, including the use of different letter or row-column arrangements, word or phrase prediction, and adjusting the scanning interval.[2]

The most obvious improvement for row-column scanning is to rearrange letters by placing 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, many dating to the 1970s or 1980s [Damper 1984; Doubler et al. 1978; Heckathorne and Childress 1983, June; Treviranus and Tannock 1987; Vanderheiden et al. 1974]. Some of the more recent efforts are hereby cited [Baljko and Tam 2006; Jones 1998; Lesher et al. 1998; Lin et al. 2008; Steriadis and Constantinou 2003; Venkatagiri 1999]. Dynamic techniques have also been tried, whereby the position of letters varies depending on previous letters entered and the statistical properties of the language [Lesher et al. 1998; Miró and Bernabeu 2008; Wandmacher et al. 2008].

A performance improvement may also emerge using a 3-level or higher selection scheme, also known as block, group, or quadrant scanning [Bhattacharya et al. 2008a; Bhattacharya et al. 2008b; Felzer and Rinderknecht 2009; Lin et al. 2007]. The general 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. Much like nested menus in graphical user interfaces, there is a trade-off between the number of levels to traverse and the number of items to traverse in each level. Block scanning is most useful to provide access to a large number of items [Bhattacharya et al. 2008a; Shein 1997]. Regardless of the scanning organization, the goal is usually to reduce the total number of scan steps to reach the desired character.

Reducing the scanning interval is another way to increase the text entry rate, but this comes at the cost of higher error rates or missed opportunities for selection. Furthermore, users with reduced motor facility are often simply not able to work with a short scanning interval. One possibility 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, or reaction time [Lesher et al. 2000; Lesher et al. 2002; Simpson and Koester 1999].

---

[2]There are also many variations on scanning keyboards that use multiple physical keys for input. As the essence of the research presented here is one-key text entry, these variations are not included in the review.

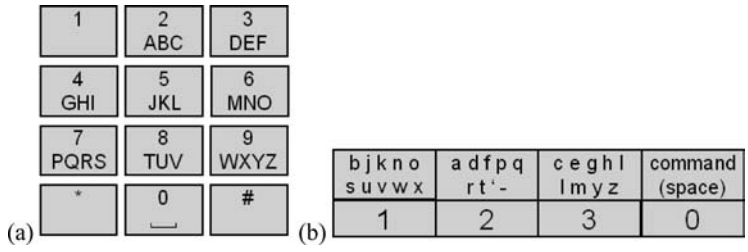

Fig. 2. Examples of ambiguous keyboards (a) phone keypad; (b) research prototype with 26 letters on 3 keys [from Harbusch and Kühn 2003a].

Word or phrase prediction or completion is also widely used to improve text entry throughput [Jones 1998; Miró and Bernabeu 2008; Shein et al. 1991; Trnka et al. 2009]. As a word is entered, the current word stem is used to build a list of matching complete words. The list is displayed in a dedicated region of the keyboard with a mechanism provided to select the word early.

We should be clear that all the techniques just described 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. SAK is different. Although SAK designs use scanning, each letter is chosen with a single selection. In this sense, SAK designs are more like ambiguous keyboards.

1.2 Ambiguous Keyboards

Ambiguous keyboards are arrangements of keys, whether physical or virtual, with more than one letter per key. The most obvious example is the phone keypad which positions the 26 letters of the English alphabet across eight keys (see Figure 2(a)).

From as early as the 1970s it was recognized that the phone keypad could be used to enter text with one key press per letter [Smith and Goodwin 1971]. Today, the idea is most closely associated with "T9," the "text on 9 keys" technology developed and licensed by Tegic Communications and widely used on mobile phones.[3] Due to the inherent ambiguity of a key press, a built-in dictionary is required to map key sequences to matching words. If multiple words match the key sequence, they are offered to the user as a list, ordered by decreasing likelihood. Interestingly enough, for English text entry on a phone keypad, there is very little overhead in accessing ambiguous words. One estimate is that 95% of words in English can be entered unambiguously using a phone keypad [Silfverberg et al. 2000].

There is a substantial body of research on ambiguous keyboards. As noted in a recent survey [MacKenzie and Tanaka-Ishii 2007], the goal is usually to employ fewer keys or to redistribute letters on the keys to reduce the ambiguity. One such example is shown in Figure 2(b), where 26 letters are positioned on just three keys [Harbusch and Kühn 2003b]. Clearly, this design yields greater

[3]Tegic was acquired in 1999 by America On Line (AOL) and in 2007 by Nuance Communications (www.nuance.com).

ambiguity. The unusual letter groupings are an effort to reduce the ambiguity by exploiting the statistical properties of the language.

All ambiguous keyboards in use involve either multiple physical keys or multiple virtual buttons that are randomly accessed ("pressed") by a finger or stylus, or clicked on using the mouse pointer. In 1998, Tegic Communications' Kushler noted the possibility of combining scanning with a phone-like ambiguous keyboard [Kushler 1998]. However, the idea was not developed with reduced-key configurations, nor was a system implemented or tested. There are at least two examples in the literature that come close to the SAK designs discussed here, but both involve more than one physical button. Harbusch and Kühn [2003a] describe a method using two physical buttons to "step scan" through an ambiguous keyboard. One button advances the focus point while a separate physical button selects virtual letter keys. Venkatagiri [1999] proposed two virtual ambiguous keyboards with scanning using either three or four letters per key. Separate physical buttons were proposed to explicitly choose the intended letter on a selected key. Both papers just cited present models only. No user studies were performed, nor were the designs actually implemented.

By adding scanning to an ambiguous keyboard with the one-switch constraint, we arrive at an interesting juncture in the design space. A "scanning ambiguous keyboard" (SAK) is a keyboard with scanned but static keys that combines the most demanding requirement of a scanning keyboard—input using one key or switch—with the most appealing feature of an ambiguous keyboard—one key press per letter. However, putting the pieces together to arrive at a viable, perhaps optimal, design requires further analysis. We need a way to characterize and quantify design alternatives to tease out strengths and weaknesses. For this, a model is required.

2. MODEL FOR SCANNING AMBIGUOUS KEYBOARDS

A model is a simplification of reality. Just as architects build models to explore design issues in advance of constructing a building, HCI researchers build interaction models to explore design scenarios in advance of prototyping and testing. In this section, we develop a model for scanning ambiguous keyboards. The model includes the following components:

—Keyboard layout and scanning pattern
—Dictionary (words and word frequencies)
—Interaction methods
—Measures to characterize scanning keyboards

2.1 Keyboard Layout and Scanning Pattern

Figure 3 shows the general idea for a scanning ambiguous keyboard. Of course, the keyboard is virtual in that it is displayed on a screen, rather than implemented as physical keys. Physical input uses a single key, button, or switch. The keyboard includes two regions: a letter-selection region (top) and a

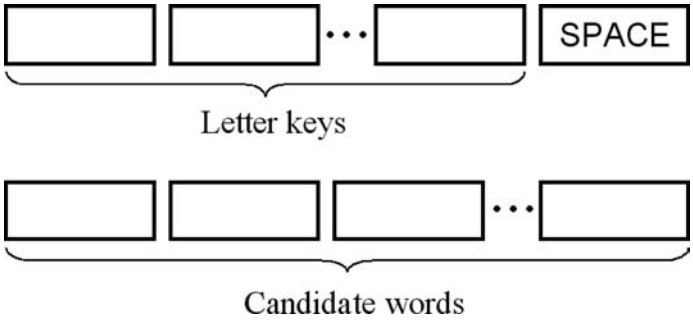

Fig. 3. Scanning ambiguous keyboard concept. There is a letter-selection region (top) and a word-selection region (bottom).

word-selection region (bottom). The design is ambiguous, meaning letters are distributed over a small number of keys, with multiple letters per key.

Scanning begins in the letter-selection region, proceeding left to right, repeating. Activating the physical key when the desired letter key is highlighted makes a selection. There is only one selection per letter.

One novel feature of SAK designs is that focus does not "snap to home" upon selecting a letter. Rather, focus proceeds in a cyclic pattern, advancing to the next key with each selection. Since SAK designs do not use multitier selection, there is no need to revert to a higher tier (e.g., the first row) after a selection within a row. Arguably, the cyclic scanning pattern used in SAK designs is intuitive and natural, since it does not involve transitions between, for example, a top tier and a bottom tier. However, cyclic scanning is predicated on using an alphabetic letter ordering, where there is no advantage in returning to a home position. So there are design issues to consider in choosing a cyclic vs. snap-to-home scanning pattern. The SAK designs considered here assume a cyclic scanning pattern.

After a word is fully entered (or partly entered, see below), the SPACE key is selected. Scanning switches to the word-selection region, whereupon the desired word is selected when highlighted. A SPACE character is automatically appended. After word selection, scanning reverts to the letter-selection region for input of the next word.

The word-selection region contains a list of candidate words, drawn from the system's dictionary, and ordered by their frequency in a language corpus. The list is updated with each selection in the letter-selection region based on the current stem. The candidate list is organized in two parts. The first presents words exactly matching the current key sequence. The second presents extended words, where the current key sequence is treated as the word stem.

The inherent ambiguity of letter selection means the list size is often >1, even if the full word is entered. The words are ordered by decreasing probability in the language; so, hopefully, the desired word is at or near the front of the list. An example is given shortly.

Fig. 4. Example letter-selection region with letters on four virtual keys.

2.2 Dictionary

Systems with ambiguous keyboards require a built-in dictionary to disambiguate key presses. The dictionary is a list of words and word frequencies derived from a corpus. The present research used a version of the British National Corpus[4] containing about 68,000,000 total words. From this, a list of unique words and their frequencies was compiled and from this list the top 9,022 words were used for our dictionary.

2.2.1 *Nondictionary Words.* The core design of SAK only works with dictionary words. Of course, a mechanism must be available to enter nondictionary words (which, presumably, are then added to the dictionary). Most scanning keyboards operate with a basic text entry mode, but implement some form of "escape mechanism" or "mode switch" to enter other modes. Additional modes are necessary for a variety of purposes, such as correcting errors, adding punctuation characters or special symbols, selecting predicted words or phrases, changing the system's configuration parameters, or switching to other applications. There are at least three commonly used mechanisms for mode switching with scanning keyboards: (i) using an additional input switch [Baljko and Tam 2006; Shein et al. 1994], (ii) using a dedicated "mode" key on the virtual keyboard [Bhattacharya et al. 2008b; Jones 1998], or (iii) pressing and holding the primary input switch for an extended period of time, say, 2–3 seconds [Jones 1998; Miró and Bernabeu 2008]. A desirable feature of SAK is to maintain a short scanning sequence in the letter-selection region; so adding even one additional virtual key is not considered a viable option, because of the impact on text entry throughput. An example method for mode switching with SAK designs is given in Section 5 (see "Escape Mode").

Returning to the central idea of SAK, the system works with words and word frequencies, but also requires a keystroke rendering of each word to determine the candidate words for each key sequence. An example will help.

2.2.2 *Example.* Let's assume—arbitrarily for the moment—that we're interested in a SAK design using four letter keys with letters distributed as in Figure 4.

If the user wished to enter "sword," the key selections are 34331. As each letter is (ambiguously) selected in the letter-selection region, a candidate list is produced in the word-selection region. Figure 5 gives the general idea of how the list progresses as entry proceeds. (The example shows only the first five words in the candidate list. In practice, a larger candidate list is needed for highly ambiguous keyboards.) The desired word (underlined) appears in the

[4]ftp://ftp.itri.bton.ac.uk/.

Keys	Candidate Words
3	the of to on that
34	tv oz ux two own
343	two system systems type types
3433	system systems systematic sword systematically
34331	sword system systems systematic systematically

Fig. 5. Keystroke sequence and candidate list for entry of "sword" using letter-key assignment in Figure 4.

fourth position with the 4th keystroke and in the first position with the 5th keystroke.

One interaction strategy to consider is selecting a word early, after the 4th keystroke in the example. There may be a performance benefit, but this will depend on the keyboard layout and scanning pattern, which in turn determines the number of scan steps required for alternative strategies. Note that in Figure 5, that "the" appears at the front of the candidate list after the first keystroke. So, "the" can be entered with two selections in the letter-selection region (3, SPACE) followed immediately by selecting the first candidate in the word-selection region. Here, there is clearly a performance benefit.

2.3 Interaction Methods

The discussion above suggests that SAK designs offer users' choices in the way they enter text. To facilitate our goal of deriving a model, four interaction methods are proposed.

2.3.1 *OLPS—One-Letter-Per-Scan Method.* With the OLPS method, users select one letter per scan sequence. The example layout in Figure 4 has five keys (4 letter keys, plus SPACE) therefore five scan steps are required for each letter in a word. Four scan steps are passive, while one is active—a user selection on the key bearing the desired letter.

2.3.2 *MLPS—Multiple-Letter-Per-Scan Method.* With the MLPS method, users take the opportunity to select multiple letters per scan sequence, depending on the word. For the layout in Figure 4, the word "city" has keystrokes 1234—no need to wait for passive scan steps between the letters. Four successive selections, followed by a fifth selection on SPACE, followed immediately by selecting "city" in the candidate list (it will be the first entry) and the word is entered. For other words, such as "zone" (4321), the MLPS opportunity does not arise.

2.3.3 *DLPK—Double-Letter-Per-Key Method.* With the DLPK method, users may make double selections in a single scan step interval if two letters are on the same key. Using our example layout, "into" has keystrokes 2233. Considering the five scan steps traversing the keys in Figure 4, the DLPK

method for "into" requires just a single pass through the scanning sequence: pause, select-select (22), select-select (33), pause, select (SPACE), select ("into"). The utility of the DLPK method will depend on the scanning interval and on the user's ability to make two selections within the scanning interval.

2.3.4 *OW—Optimized-Word Method.* With the OW method, users seize opportunities to optimize by making an early selection if the desired word appears in the candidate list before all letters are entered *and if there is a performance benefit over the alternative strategy of simply continuing to select keys until all letters are entered*. An extreme example was noted above for "the"—the most common word in English. As it turns out, the OW method offers a performance benefit for many common words in English. Referring again to Figure 5, "of," "to," "on," and "that"—all very common words—appear with the first key selection. Bear in mind that the cost of not performing early selection is at least one more pass through the letter-selection scanning sequence.

The interaction methods just described are progressive in that the first method is easiest to use, but slow. The last method is hardest to use, but fast. Users need not commit to any one method. The methods are simply ways to characterize the sort of interaction behaviors users are likely to exhibit. Most likely, users will mix the methods, but, with experience, will migrate to behaviors producing performance benefits.

We are in a position now to consider design alternatives to the "arbitrary" letter assignment in Figure 4. There is clearly a trade-off between having fewer keys with more letters/key and having more keys with fewer letters/key. With fewer keys, the number of scan steps is reduced but longer candidate lists are produced. With more keys, the number of scan steps is increased while shortening the candidate lists.

However, one component in our model is still missing. We need a way to characterize and quantify scanning ambiguous keyboards of the sort described here. We need a measure that can be calculated for a variety of design and interaction scenarios and that can be used to make informed comparisons and choices between alternative designs.

2.4 Characteristic and Performance Measures

Measures for text entry are of two types: characteristic measures and performance measures. Characteristic measures describe a text entry method without measuring users' actual use of the method. They tend to be theoretical—describing and characterizing a method under circumstances defined a priori. Performance measures capture and quantify users' proficiency in using a method. Often, a measure can be both. For example, text entry speed, in words per minute, may be calculated based on a defined model of interaction and then measured later with users. In this section we present three new measures for text entry with scanning keyboards. Two are characteristic measures, but can be measured as well; one is a performance measure.

First we mention *KSPC*, for "keystrokes per character," as a characteristic measure applicable to numerous text entry methods. *KSPC* is the number of

keystrokes required, on average, to generate a character of text for a given text entry technique in a given language [MacKenzie 2002a; Rau and Skiena 1994]. For conventional text entry using a Qwerty keyboard, $KSPC = 1$ since each keystroke generates a character. However, other keyboards and methods yield $KSPC < 1$ or $KSPC > 1$. Word completion or word prediction techniques tend to push *KSPC* down, because they allow entry of words or phrases using fewer keystrokes than characters. On the other hand, ambiguous keyboards, such as a mobile phone keypad, tend to push *KSPC* up. For example, a phone keypad used for English text entry has $KSPC \approx 2.03$ when using the multitap input method or $KSPC \approx 1.01$ when using dictionary-based disambiguation [MacKenzie 2002a].

2.4.1 *Scan Steps per Character (SPC).* "Scan steps per character" (*SPC*) is proposed here as a characteristic measure for scanning keyboards. *SPC* is similar to *KSPC*. *SPC* is the number of scan steps, on average, to enter a character of text using a given scanning keyboard in a given language. *SPC* includes both passive scan steps (no user action) and active scan steps (user selection).

A feature of *SPC* is that it directly maps to text entry throughput, T, in words per minute, given a scanning interval, SI, in milliseconds:

$$T = \left(\frac{1}{SPC}\right) \times \left(\frac{1000}{SI}\right) \times \left(\frac{60}{5}\right). \tag{1}$$

The first term converts "scan steps per character" into "characters per scan step". Multiplying by the second term yields "characters per second" and by the third term "words per minute."[5] For example, if the scanning interval is, say, 800 ms, and $SPC = 4.0$, then

$$T = \left(\frac{1}{4.0}\right) \times \left(\frac{1000}{800}\right) \times \left(\frac{60}{5}\right) = 3.75 \text{ wpm}. \tag{2}$$

Of course, this assumes the user "keeps up"—performs selections according to the interaction method used in the *SPC* calculation.

Figure 6 demonstrates the scan step sequences for entering "computer" (13233313) using each of the four interaction methods described above. Again, Figure 4 serves as the example design. The scan step sequence (2nd column) shows a lowercase letter for selecting a key bearing the indicated letter (or sometimes a double selection for the DLPK and OW methods). A period (".") is a passive scan step, or a scan step interval without a selection. "S" is a selection on the SPACE key and "W" is a word-selection. The scan count is simply the number of scan steps. *SPC* is the scan count divided by nine, the number of characters in "computer" +1 for a terminating SPACE. (*SPS* is discussed in the next section.)

The figure demonstrates a progressive reduction in *SPC* as the interaction method becomes more sophisticated. With the OLPS method, only one letter is selected per scan step. Since "computer" has eight letters, $8 \times 5 = 40$ scan steps

[5]It is a long-standing convention to define "word" in "words per minute" as five characters [Yamada 1980, p. 182]. This includes letters, spaces, punctuation, and so on.

Method	Scan Step Sequence	Scan Count	*SPC*	*SPS*
OLPS	c.....o...m.....p....u....t..e.....r.SW	41	4.56	0.238
MLPS	c.o...mp....u....t..e.r.SW	26	2.89	0.385
DLPK	c.o...mp....t..e.r.SW	21	2.33	0.476
OW	c.o...mp.S...W	14	1.56	0.500

Fig. 6. Example calculation of *SPC* (scan steps per character) and *SPS* (selection per scan step) for entry of "computer" using the scanning keyboard in Figure 4. See text for discussion.

are required, plus a final active scan step to select the word in the candidate list. As word selection ("W") immediately follows SPACE ("S"), evidently the word was at the front of the candidate list. For the MLPS method, "o," "p," and "r" are entered in the same scan sequence as the preceding letter, thus saving $3 \times 5 = 15$ scan steps. The DLPK method improves on this by combining "p" and "u" in a double-selection, since they are on the same key. (Only the first letter, "p," is shown in the figure.) Five scan steps are saved.

A further improvement is afforded by the OW method. After double selecting "pu," "computer" appears in the candidate list in the fourth position. The opportunity is taken, producing a further savings of 7 scan steps. As an aside, the words preceding "computer" in the candidate list at this point were "bonus," "donor," and "control" (not shown). Because "bonus" and "donor" are exact matches with the key sequence 13233, they are at the front of the list. "control," "computer," and a few other words, follow as possible extended words matching the current numeric word stem. They are ordered by their frequencies in the dictionary.

2.4.2 *Selections per Scan Step (SPS).* Although reducing scan steps per character (*SPC*) is an admirable pursuit, there is a downside. To capture this, we introduce *SPS*, for "selections per scan step". *SPS* is the number of selections divided by the total number of scan steps. It can be computed for individual words or phrases, or as an overall weighted average for a given scanning keyboard, interaction method, and language.

As seen in the two right-hand columns of Figure 6, as *SPC* decreases, *SPS* increases. *SPS* captures, to some extent, the cognitive or motor demand on users. While passive scan steps may seem like a waste, they offer users valuable time to rest or to think through the spelling and interactions necessary to convey their message. Although this is a moot point for highly inefficient scanning methods, as depicted in Figure 1, it becomes relevant as more ambitious designs are considered—designs that push *SPC* down. With the OW method in Figure 6, $SPS = 0.500$. One selection for every two scan steps may not seem like much; however, if taking opportunities to quicken interaction involves, for example, viewing a list of candidate words, then cognitive demand may be substantial.

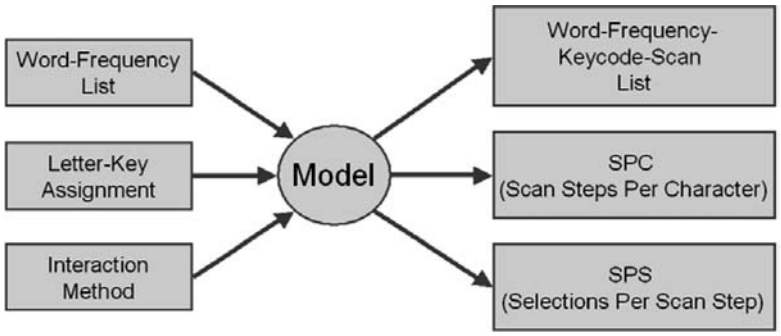

Fig. 7. Modeling tool for scanning ambiguous keyboards.

2.4.3 *Scanning Efficiency (SE).* The OW scan sequence for "computer" in Figure 6 represents the absolute minimum number of scan steps for which this word can be entered. This is true according to the embedded dictionary and the defined layout and behavior of the example SAK. The other scan sequences represent less efficient entry. The distinction is important when considering overall performance or when analyzing user behavior and learning patterns. For example, the phrase "the quick brown fox jumps over the lazy dog" requires only 92 scan steps if all opportunities to optimize are taken. Will users actually demonstrate this behaviour? Probably not. Will they eventually, or occasionally, demonstrate this behavior as expertise evolves? Perhaps.

To capture user performance while using scanning keyboards, we introduce Scanning Efficiency (*SE*) as a human performance measure:

$$SE = \frac{scan_{MIN}}{scan_{USER}} \times 100\%. \tag{3}$$

For example, if a user was observed to enter the quick-brown-fox phrase in, say, 108 scan steps, then

$$SE = (92/108) \times 100 = 85.2\% \tag{4}$$

As expertise develops and performance improves toward optimal behavior, *SE* will increase toward 100%. *SE* can be computed for the entry of single words, entire phrases, or as an overall human performance measure of user efficiency while using a scanning keyboard for text entry.

2.5 Searching for an Optimal Scanning Ambiguous Keyboard

Given the components of the model described above, software tools were built to search for an optimal scanning ambiguous keyboard. We define *Optimal* as a design that minimizes scan steps per character (*SPC*). Whether the scanning interval is 1 second or 5 seconds, a design with a lower *SPC* will produce a higher text entry throughput than a design with a higher *SPC*, all else being equal.

Figure 7 illustrates the general operation of the model. Three inputs are required: a dictionary in the form of a word-frequency list, a letter-key assignment (e.g., as per Figure 4), and a specification of the interaction method.

The interaction method is OLPS (one letter per scan), MLPS (multiple letters per scan), DLPK (double letters per key), or OW (optimized word), as discussed earlier. For the dictionary, the 9,022-word list noted earlier was used. The letter-key assignment is used, among other things, to develop numeric keycodes for words in the dictionary. These are added to the word-frequency list to form a word-frequency-keycode list.

Given the word-frequency-keycode list, a scan step sequence is built for every word in the dictionary for the specified input method. The result is a word-frequency-keycode-scan list. The scan steps for "computer" were given earlier for all four methods (see Figure 6). Generating the candidate list for the OW method is somewhat involved, due to ambiguity in the keycodes. The list must be built after every key selection. If the word appears in the list, a decision is required on whether to choose early selection, if there is a performance benefit, or to continue with the next selection. From the word-frequency-keycode-scan list, *SPC* and *SPS* are then computed as a weighted average over the entire dictionary. The process is then repeated for other letter-key assignments.

The calculations we have described are but one part of a complex design space. Although the model is highly flexible, the search was constrained to designs placing letters on 2, 3, 4, 5, or 6 keys. These designs span a range that avoids absurdly long candidate lists (e.g., a 1-key design) while maintaining a reasonably small number of scan steps across the letter-selection region.

The search was further constrained to alphabetic letter arrangements only. Relaxing this constraint not only causes an explosion in the number of alternative designs, it also produces designs that increase the cognitive demand on users who must confront an unfamiliar letter arrangement. While designs with optimized letter arrangements often yield good predictions "on paper," they typically fail to yield performance benefits for users [Baljko and Tam 2006; Bellman and MacKenzie 1998; MacKenzie 2002b; Miró and Bernabeu 2008; Pavlovych and Stuerzlinger 2003; Ryu and Cruz 2005].

Even with the constraints described above, the search space is substantial because of the number of ways letters may be assigned across keys. In particular, if n letters are assigned in alphabetic order across m keys, the number of assignments (N) is

$$N = \frac{(n-1)\,!}{(m-1)\,!\ \times\ (n-m)\,!}. \tag{5}$$

For 26 letters assigned across 2, 3, 4, 5, or 6 keys, the number of letter-key assignments is 25, 300, 2300, 12650, and 53130, respectively.

An exhaustive search was undertaken to find the letter-key assignment generating the lowest *SPC* for each interaction method (OLPS, MLPS, DLPK, OW) for alphabetic assignments over 2, 3, 4, 5, and 6 keys. The results are shown in Figure 8. The range is from $SPC = 5.91$, using the OLPS method with six letter keys, down to $SPC = 1.834$, using the OW method with three letter keys.

Note that for each column in Figure 8, the letter-key assignment yielding the lowest *SPC* was slightly different among the input methods. Nevertheless,

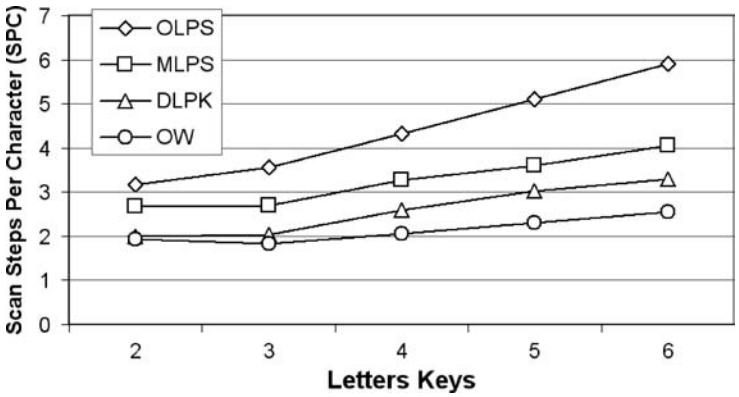

Fig. 8. Minimized scan steps per character (*SPC*) for the OLPS, MLPS, DLPK, and OW interaction methods for designs with 2 through 6 letter keys.

Letter Keys	Letter-Key Assignment	*SPC*	*SPS*
6	abcdefgh-ijkl-mno-pqr-stu-vwxyz	2.551	0.3521
5	abcdefgh-ijklm-nopqr-stu-vwxyz	2.299	0.4072
4	abcdefgh-ijklm-nopqr-stuvwxyz	2.066	0.4771
3	abcdefgh-ijklmnop-qrstuvwxyz	1.834	0.5765
2	abcdefghijklm-nopqrstuvwxyz	1.927	0.5830

Fig. 9. Letter-key assignments producing the optimal scan steps per character (*SPC*) for the optimized-word (OW) input method. The *SPS* (selections per scan step) statistics are given as well.

there was a coalescing of results: Designs generating the lowest, or near lowest, *SPC* for one interaction method faired similarly for the other methods. Figure 9 gives the chosen optimal design for each size. The assignment chosen in each case was based on the OW method. The *SPC* and *SPS* statistics are also provided.

SPC decreases reading down the table, but experiences an increase at two letter keys. This is due to the increased ambiguity of placing 26 letters on two keys. The result is longer candidate lists with more passive scan steps required to reach the desired words. The similarity in the designs is interesting. For designs with 3 through 6 letter keys, the first break is after "h," and for designs 4 and 5 the next two breaks are after "m" and "r." This is likely due to the performance benefits in selecting or double selecting letters early in the scanning sequence, to avoid an additional pass if possible.

2.5.1 *Ambiguity Analysis.* The specter of highly ambiguous words and long candidate lists is unavoidable with ambiguous keyboards, particularly if letters are positioned on just a few keys. The *SPC* calculation described above accounts for this, but does so only as an overall average for text

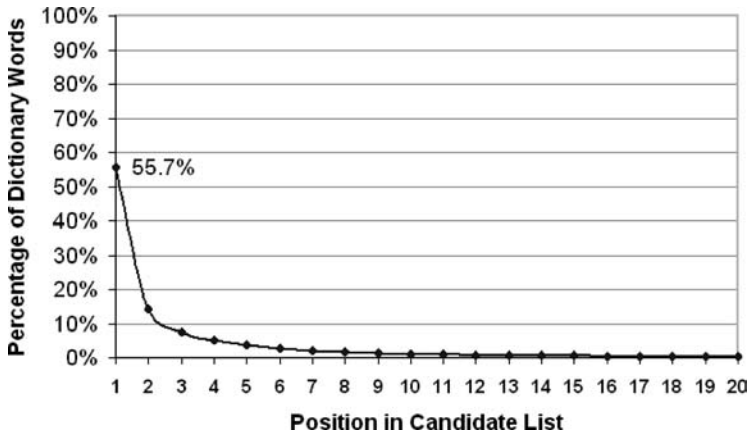

Fig. 10. Percentage of dictionary words at each position in the candidate list. The analysis is for the design in Figure 9 with three letter keys using a 9022-word dictionary.

entry using a particular letter-key assignment and the built-in dictionary. While $SPC < 2$ is remarkably low, we are still left wondering: How long are the candidate lists? What is the typical length? What is the worst case? Figure 10 offers some insight for the design in Figure 9 using three letter keys ($SPC = 1.834$). The analysis uses the 9,022 unique words from the corpus noted earlier. Evidently, 55.7% of the words are at the front of the candidate list when word selection begins. This figure includes both unambiguous words (41.5%) and those ambiguous words that are the most frequent of the alternatives (14.2%). The pattern follows a well known relationship in linguistics known as Zipf's law, where a small number of frequently used language elements cover a high percentage of all elements in use [Wobbrock et al. 2006; Zhai and Kristensson 2003].

Considering that the three-letter-key design requires four scan steps for a single pass through the letter-selection region, the cost of having words in positions 1 through 4 in the candidate list is minor. As a cumulative figure, 82.2% of the words appear in the candidate list at position 4 or better. The same figures are 94.8% at position 10 and 99.6% at position 20. So, very long candidate lists are rare. But, still, they will occasionally occur. As an example of position 20, if a user wished to enter "Alas, I am dying beyond my means,"[6] after entering 1213 for "alas," the candidate list would be quite tedious: {does, body, goes, boat, dogs, diet, flat, coat, andy, gift, flew, ends, aids, alex, clay, bias, blew, gods, dies, alas}. This is an extreme example. In this case, the poor position of "alas" is compensated for, at least in part, by the good position of the other words: "I" (1), "am" (4), "dying" (6), "beyond" (2), "my" (4), "means" (1). The worst case for the test dictionary is "dine" (1221) at position 30. Matching moreprobable candidates include "file," "gold," "cope," "golf," and so on.

[6]Allegedly uttered by Oscar Wilde, as he sipped champagne on his deathbed.

Corpus	Unique Words	Letter Keys				
		2	3	4	5	6
BNC-1	9,022	1.927	1.834	2.066	2.299	2.551
BNC-2	64,588	2.160	1.905	2.136	2.383	2.611
Brown	41,532	2.373	2.027	2.298	2.600	2.848

Fig. 11. Scan steps per character (*SPC*) calculated for the BNC-1, BNC-2, and Brown corpora for letter keys 2 through 6 using the designs in Figure 9.

2.5.2 *Corpus Effect.* It is worth considering the effect of the dictionary on the *SPC* calculation. If the dictionary had substantially more words or if it were derived from a different corpus, what is the impact on *SPC*? To investigate this, the *SPC* values in Figure 9 were re-calculated using two additional word-frequency lists. One was a much larger version of the British National Corpus containing 64,588 unique words (BNC-2). Another was the well-known Brown corpus of American English [Kucera and Francis 1967] containing about 41,000 unique words from a sample of about one million words. The results are shown in Figure 11. For comparison, the top row (BNC-1) gives the *SPC* values from Figure 9.

The most reassuring observation in Figure 11 is that substantially larger dictionaries do not cause an untoward degradation in performance. For the three-letter-key design, increasing the dictionary from 9,022 words (BNC-1) to 64,588 words (BNC-2) produced only a 3.9% increase in *SPC*, from $SPC = 1.834$ to $SPC = 1.905$. That the increase is small is likely because the additional words are, for the most part, larger and more obscure than the core 9,022 words. Ambiguity tends to arise with shorter words. Even though the Brown corpus has fewer words than BNC-2, there is evidently more ambiguity. For the three-letter-key design, the degradation compared to BNC-1 is 10.5%, from $SPC = 1.834$ to $SPC = 2.027$. The explanation here is simple. The letter-key assignment used in the calculation (see Figure 9) was based on optimization for the BNC-1 dictionary. If one wished to design a scanning ambiguous keyboard using a dictionary of words from the Brown corpus, or any other corpus or source, then the modeling process should use that dictionary.

Overall, the results above are promising. The optimal design for three letter keys ($SPC < 2$) suggests that English text can be entered in less than two scan steps per character on average. Of course, with $SPS > 0.5$, the cognitive or motor demand may pose a barrier to attaining the maximum possible text entry throughput. "Maximum possible" is the correct term here. Assuming users take all opportunities to optimize, they will indeed attain the maximum text entry rate. Of course, the actual rate depends on the scanning interval, which must be tailored to the abilities of the user. Just as an example, if the scanning interval was, say, 700 ms, the maximum text entry throughput (T) for the design in Figure 9 with three letter keys is given by Equation (1) as

$$T = \left(\frac{1}{1.834}\right) \times \left(\frac{1000}{700}\right) \times \left(\frac{60}{5}\right) = 9.35 \text{ wpm}. \quad (6)$$

This is an average rate for English, assuming the use of the BNC-1 dictionary. The rate for individual words or phrases may differ depending on their linguistic structure.

The text entry rate just cited (9.35 wpm) is quite good for one-key input with a scanning keyboard. Miró and Bernabeu [2008] cite a predicted text entry rate of 10.1 wpm with a one-key scanning keyboard using two-tier selection. However, their rate was computed using a scanning interval of 500 ms. Using a scanning interval of 500 ms in Equation (6) yields a predicted text entry rate of 13.1 wpm. Predictions are one thing; tests with users are quite another. Rates reported in the literature that were measured with users are lower (although they are difficult to assess and compare due to variation in the methodologies). Baljko and Tam [2006] researched text entry with a scanning keyboard with letter positions optimized using a Huffman coding tree. They reported entry rates ranging from 1.4 wpm to 3.1 wpm in a study with twelve able-bodied participants. The faster rate was achieved with a scanning interval of 750 ms. In Simpson and Koester's [1999] study with eight able-bodied participants, entry rates ranged from 3.0 wpm to 4.6 wpm using a scanning keyboard with an adaptive scanning interval. The faster rate was achieved with a scanning interval just under 600 ms.

It remains to be seen whether users can actually achieve the respectable entry rates conjectured for a scanning ambiguous keyboard of the type described here. In the next section, a "test of concept" evaluation is presented. For this, the three-letter-key design yielding the lowest *SPC* (see Figure 9) was chosen for evaluation.

3. EXPERIMENT 1: TEST OF CONCEPT

3.1 Participants

Twelve able-bodied participants were recruited from the local university campus. The mean age was 25.3 years ($SD = 3.2$). Five were female, seven male. All participants were regular users of computers, reporting an average daily usage of 7.6 hours. Testing took approximately one hour, for which participants were paid ten dollars.

3.2 Apparatus

A prototype scanning ambiguous keyboard (SAK) was created in Java. The application included a letter-selection region and a word-selection region, as described earlier. For experimental testing, regions were also included to present text phrases for input and to show the transcribed text and keycodes during entry. Input was performed using any key on the system's keyboard.

Several parameters configured the application upon launching, including the scanning interval, the letter-key assignment, a dictionary in the form of a word-frequency list, and the name of a file containing test phrases. A screen snap of the application, called *OneKey*, is shown in Figure 12.

In the screen snap, focus is on the first letter key. Focus advances in the expected manner, according to the scanning interval. As selections are made,

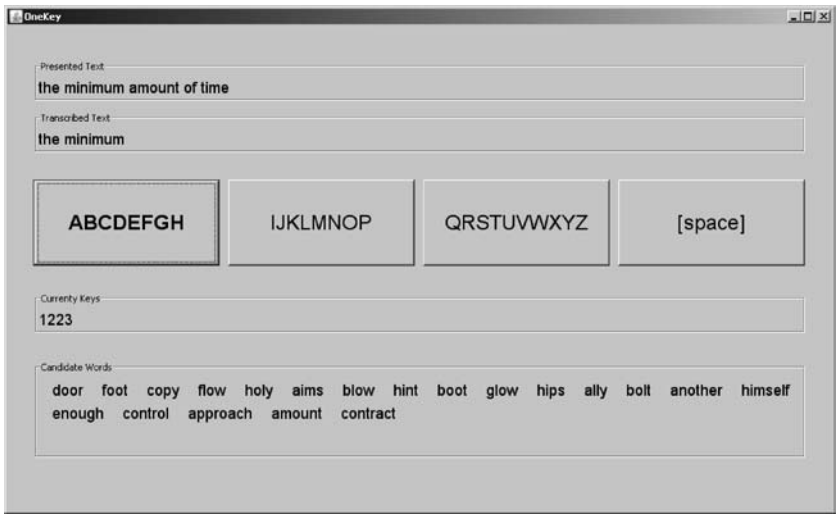

Fig. 12. Screen snap of the *OneKey* application. See text for discussion.

focus advances to the next key, rather than reverting to the first key (as noted earlier). In the image, the user has entered the first two words in the phrase "the minimum amount of time." The first four keys (1223) of the next word ("amount") have been entered. The candidate list shows words exactly matching the key sequence followed by extended matching words. The desired word is at position 18 in the list. The correct strategy here is to continue selecting letter keys to further spell out the word. When the user finishes selecting letter keys, a selection on space ("[space]") transfers focus to the word selection region. Words are highlighted in sequence by displaying them in blue with a focus border. When the desired word is highlighted, it is selected and added to the transcribed text.

Timing begins on the first key press for a phrase and ends when the last word in a phrase is selected. At the end of a phrase, a popup window shows summary statistics such as entry speed (wpm), error rate (%), and scanning efficiency (%). Pressing a key closes the popup window and brings up the next phrase for input.

3.2.1 *Phrase Set.* The outcome of text entry experiments may be affected by the text users enter. Depending on the entry method, text may be chosen specifically to elicit a favorable outcome. In the present experiment, for example, phrases could be concocted from words with low *SPC* values, thus creating an artificially inflated text entry throughput. This was not done. Instead, a generic set of 500 phrases was used [MacKenzie and Soukoreff 2003]. Some examples are given in Figure 13. For each trial, a phrase was selected at random and presented to the user for input.

The phrase set was designed to be representative of English. Phrase lengths ranged from 16 to 43 characters ($mean = 28.6$). In analyzing the phrases, it was

```
please follow the guidelines
an airport is a very busy place
mystery of the lost lagoon
is there any indication of this
are you sure you want this
the fourth edition was better
```

Fig. 13. Example phrases used in the experiment (from MacKenzie and Soukoreff [2003].

determined that about 10% of the 1164 unique words were not in the dictionary. These words were added to the dictionary (with frequency = 1); thus, insuring that all phrases could be entered using the SAK under test.

To gain a sense of the linguistic structure of the phrases, a small utility program was written to compute *SPC* for every phrase. For the phrase set overall, $SPC = 1.980$, with a best case of $SPC = 1.226$ ("great disturbance in the force") and a worst case of $SPC = 3.833$ ("my bike has a flat tire"). Given that $SPC = 1.834$ for the SAK used in the experiment, the test phrases were, on average, slightly more difficult than English as represented in the embedded dictionary.

3.2.2 *Error Correction.* Error correction was implemented using a "long press"—pressing and holding the input key for two or more scan step intervals. While the input key was held, scanning was suspended. Scanning resumed when the key was released. If a long press occurred during entry of a word, the effect was to clear the current key sequence. If a long press occurred between words, the effect was to delete the last word in the transcribed text region.

3.3 Procedure

After signing an informed consent form, participants were told the general idea of text entry using scanning keyboards, ambiguous keyboards, and a scanning ambiguous keyboard. The text entry method was demonstrated using an initial scanning interval of 1100 ms. The operation of the letter-selection and word-selection regions was explained. Participants were allowed to enter a few practice phrases and ask questions while further instructions were given on the different ways to select letters and words (as per the OLPK, MLPK, DLPK, and OW methods discussed earlier) and the way to correct errors. They were told to position their hand comfortably in front of the keyboard and to make selections with any key. Selection using the index finger on the home row (F-key using the right hand) was recommended.

Participants were asked to study each phrase carefully—including the spelling of each word—before beginning entry of a phrase. They were reminded that timing did not begin until the first key press. Entry was to proceed as quickly and accurately as possible. They were encouraged to correct errors, if noticed, but were also told that perfect entry of each phrase was not a requirement.

At the end of the experiment, participants completed a brief questionnaire.

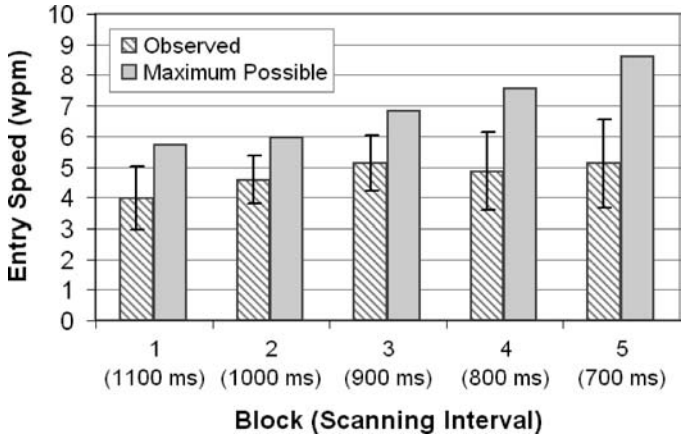

Fig. 14. Observed entry speed (wpm) by block and scanning interval. For each block, the maximum possible entry speed and the scanning interval are also shown. Error bars show ± 1 *SD*.

3.4 Design

The experiment included "block" as a within-subjects factor with five levels (1, 2, 3, 4, 5). Each block of input consisted of five phrases of text entry. The total amount of input was 12 participants $\times$ 5 blocks $\times$ 5 phrases/block = 300 phrases.

The scanning interval was set to 1100 ms for the first block of testing. It was decreased by 100 ms per block, with a final setting of 700 ms in block 5.

The dependent variables were text entry speed (wpm), scanning efficiency (%; Equation (3)), error rate (%), and word corrections (the number of "long presses" per phrase).

4. RESULTS AND DISCUSSION

4.1 Text Entry Speed and Efficiency

The grand mean for text entry speed was 4.73 wpm. There was a 28.3% improvement over the five blocks of input with observed rates of 3.98 wpm for the 1st block and 5.11 wpm for the 5th block. The trend is shown in Figure 14.

While the trend was statistically significant ($F_{4,44} = 7.58, p < .0001$), this must be considered in light of the confounding influence of scanning interval, which inherently increases the text entry rate, all else being equal. So an increase in text entry speed was fully expected. In fact, the experiment, as a test of concept, was designed to ease participants into the operation of the SAK, and to gradually elicit an increase in text entry throughput by gradually decreasing the scanning interval.

Figure 14 also shows the maximum possible entry speed for each block. The value for each block was computed according to Equation (1) using the scanning interval and the average *SPC* for the 60 phrases randomly selected for that block (12 participants $\times$ 5 phrases/block). Each *SPC* used in the calculation was computed using the optimized word (OW) entry method. As noted

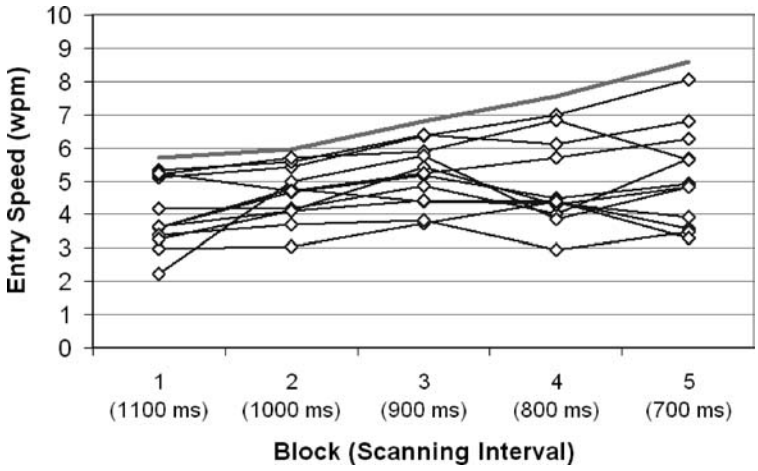

Fig. 15. Per-participant text entry speed by block and scanning interval. The line at the top shows the maximum possible text entry speed.

earlier, this value is the minimum scan steps for which a phrase can be entered, assuming users take all opportunities to optimize. For the 5th block, the maximum possible speed was 8.60 wpm.

The pattern in Figure 14 suggests that participants were not able to improve their text entry throughput after the 3rd block. This is particularly true if comparing the observed speed to the maximum possible speed. Admittedly, decreasing the scanning interval from block to block was a bit of a gamble: Would participants' emerging expertise, combined with a decreasing scanning interval, yield an increase in their text entry speed, proportional to the maximum possible rate? Clearly, the answer was "no." There are a number of possible reasons for this. One is simply that the experiment afforded too little time for participants to learn and develop entry strategies before reducing the scanning interval. Each block involved only about 8–10 minutes of text entry. Another possible reason is that the cognitive or motor demand was simply too high, when considering the need to make frequent selections with the three letter-key design under test.

However, further insight lies in the large standard deviation bars for the 4th and 5th blocks in Figure 14. These suggest substantial individual differences in the responses across participants. Figure 15 shows this by plotting the trend for each participant over the five blocks. Evidently, some participants continued to improve in the 4th and 5th blocks. Three participants achieved mean text entry speeds above 6 wpm in the 5th block, with one achieving a mean of 8.05 wpm—very close to the maximum possible speed.

While the inability of participants—overall or individually—to attain the maximum possible entry rate is worthy of analysis and speculation, the mean entry speed of 5.11 wpm in the 5th block is still quite good for one-key text entry. Of the studies noted earlier, the closest is Simpson and Koester's [1999] system with an adaptive scanning interval. Their maximum reported entry

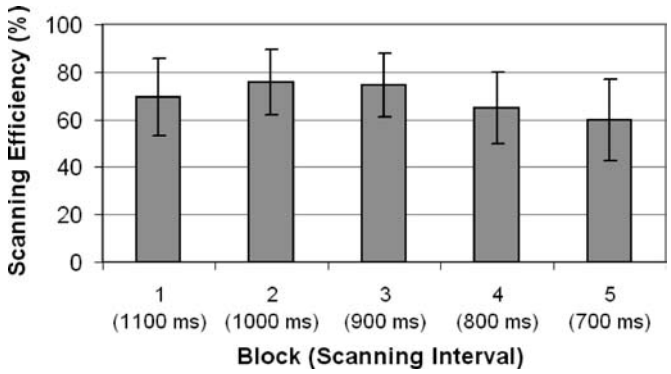

Fig. 16. Scanning efficiency by block and scanning interval. Scanning efficiency is the ratio of the minimum number of scan steps to the observed number of scan steps. The error bars show ± 1 *SD*.

speed was 22.4 characters per minute, or 4.6 wpm. However, they excluded trials where participants missed selection opportunities. Furthermore, their methodology excluded error correction. By contrast, the figure reported here of 5.11 wpm includes all trials in the 5th block (12 participants $\times$ 5 phrases each). The measurements include the time lost through missed selections as well as the time in correcting errors. So, overall, the results reported here for a scanning ambiguous keyboard are very promising.

Participants with lower text entry speeds made more selection errors and had more difficulty seizing opportunities to optimize their entry, particularly with shorter scanning intervals. Figure 16 shows the overall trend of participants in terms of scanning efficiency—the ratio of the minimum number of scan steps to the observed number of scan steps (Equation (3)).

Scanning efficiency (*SE*), as a human performance measure, is intended to reflect participants' ability to make selections at the earliest opportunity. *SE* improved after the 1st block, reaching about 75% in the 2nd and 3rd blocks. After that, *SE* dropped—to 64.8% in the 4th block and to 58.8% in the 5th block. Although the most germane explanation is that participants missed opportunities to optimize (and this is certainly true), another explanation seems more likely. In perusing the raw data, evidently many additional scan steps were present due to selection errors and corrections.

The large standard deviation bars in Figure 16 suggest that some participants fared better than others. It is felt that the predominant underlying cause for the drop in *SE* is insufficient practice. Participants were by no means experts at the end of the experiment. With sufficient practice and a more gradual shortening of the scanning interval, scanning efficiency would likely edge upward, as would text entry speed.

4.2 Accuracy

Accuracy was represented in the experiment in two ways: uncorrected errors and corrected errors. Errors remained in the transcribed text if a participant selected a wrong word and did not correct it (perhaps because it was not

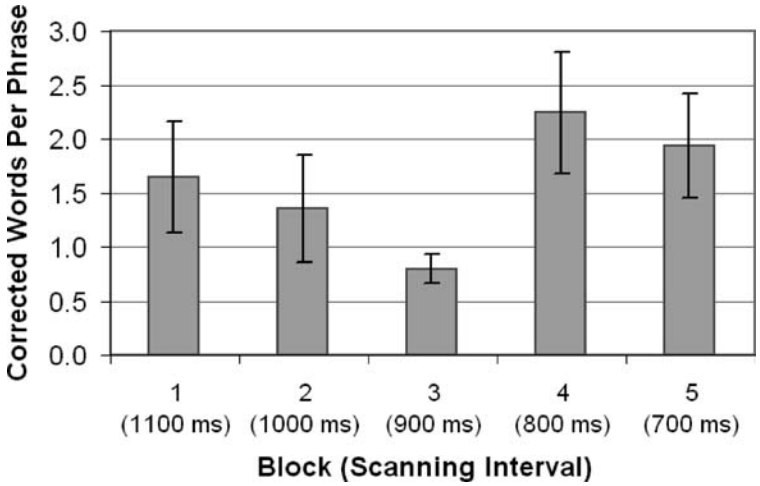

Fig. 17. Corrected words per phrase by block and scanning interval. Error bars show ±1 *SD*.

noticed). These are uncorrected errors. Such errors were measured by computing the minimum string distance between the presented and transcribed text [MacKenzie and Soukoreff 2002; Soukoreff and MacKenzie 2001]. The result is the character-level error rate as a percent. As it turns out, the error rate in the experiment was very low. The overall mean error rate was just 0.96%, equivalent to 99.04% accuracy. In fact, of the 300 total phrases, 280 were error free. Evidently, participants tended to notice when they committed errors and chose to correct them.

The second form of accuracy is corrected errors, or, more precisely, "corrected words." If a participant committed an error and noticed it, he or she could correct the error using a "long press," defined earlier. The effect was to erase either the current word (i.e., current key selection sequence) or the last word entered. The participant would then reenter the word. The results for corrected words are shown in Figure 17.

Clearly, the 4th and 5th blocks were problematic. Approximately two word corrections per phrase may not seem like much; however, a substantial number of scan steps are often involved because corrections tended to occur with longer words. There are two reasons. One is simply that there are more chances to make a selection error if there are many selections. Another is due to the inherent characteristic of ambiguous keyboards that the display is unstable during entry of a word [Silfverberg et al. 2000]. Because of this, it is hard to detect a selection error until the end of a word, wherein one discovers that the desired word is not among the candidates.

The large error bars in the 4th and 5th blocks in Figure 17 signal that something is amiss. A closer look as the data revealed that some participants occasionally had difficulty. In all, there were 32 phrases with 5 or more word corrections. Of these, 8 phrases had 10 or more word corrections. The majority of these were during the 4th or 5th block where the scanning interval was shortest. While a criterion could have been developed to classify some trials as

Participant	What did you think of the scanning speed in the last block?	Could you use the interface if scanning was faster?	Text Entry Speed (wpm) in 5th block
P01	FAST	YES	3.58
P02	FAST	YES	4.93
P03	OK	YES	8.05
P04	FAST	YES	5.67
P05	OK	YES	6.81
P06	VERY FAST	NO	5.66
P07	OK	YES	3.92
P08	OK	YES	4.84
P09	FAST	NO	4.82
P10	OK	YES	6.27
P11	FAST	NO	3.48
P12	FAST	NO	3.28

Fig. 18. Participant responses to two questions along with text entry speed (wpm) in the 5th block.

outliers, this was not done, given that this is the first evaluation of a scanning ambiguous keyboard. All behaviors were considered reasonable, at least for this initial test.

4.3 Participant Questionnaire

As well as soliciting comments in general, participants were asked two questions at the end of the experiment. The questions and responses are shown in Figure 18, along with each participant's text entry speed (wpm) in the 5th block.

Seven participants considered the scanning interval in the 5th block either FAST or VERY FAST; five indicated it was OK. Eight participants felt they could use the interface with an even shorter scanning interval; four felt they could not. Of the four who felt they could not, two (P11, P12) had text entry rates under 4 wpm in the 5th block. The three participants (P03, P05, P10) with text entry speeds above 6 wpm all answered OK/YES, evidently feeling confident they could achieve even better performance.

Participants also shared a variety of observations and comments. Frustration was expressed on waiting for words when they were situated near the end of the candidate list. Evidently, selections that crossed the boundary of a scanning interval were lost and caused a word reset (key selections erased). One participant expressed a desire to allow triple selections in a scanning interval. Others commented on feeling rushed trying to make selections within a brief interval and also in switching attention from the letter-selection region to the word-selection region.

4.4 Timer Restart on Selection

Two participants commented that additional time can be added to the scan step interval by restarting the timer when a selection is made. The effect is to extend the current scan step interval. This gives the user additional time to consider

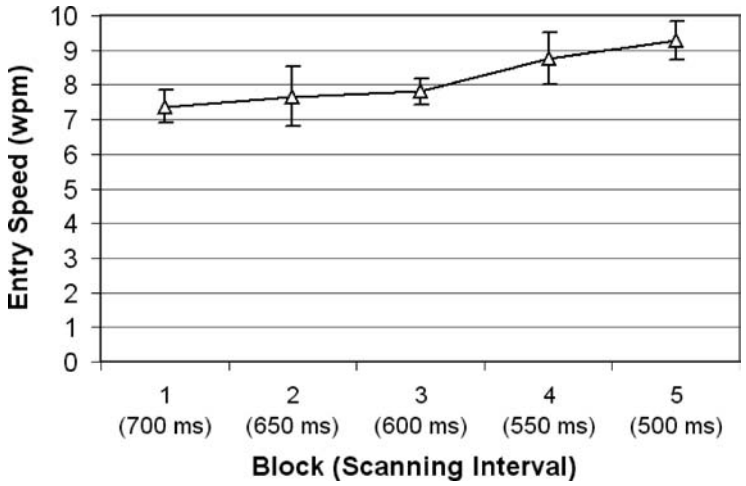

Fig. 19. Extended trials for one participant using the "timer restart on selection" mode. This mode facilitates making multiple selections per scan step. The data are for five blocks with five phrases per block.

the need for a possible selection in the next scan step interval. Importantly, this provides extra time when SPACE is selected to consider the first entry in the candidate list. One participant suggested doubling the scanning interval for the first entry in the candidate list. This seems reasonable considering there is always a selection in the preceding scan step interval ("[space]") and that the cost of missing a word in the candidate list is high.

Restarting the timer also facilitates making a second selection in a scan step interval, because the effect is to make the full scan step duration available again for the second selection. Furthermore, if a second selection is made, the timer is again reset, facilitating a third selection, and so on. As it turns out, with the three letter-key arrangement evaluated here, there are numerous opportunities for more than two selections on a key. Some examples are "had" (111), "each" (1111), "try" (333), "feeling" (1112221), "fact" (1113), "ideal" (21112), "been" (1112), and so on. This modification suggests a new "interaction method" for the model. The DLPK (double letter per key) interaction method extends to an MLPK (multiple letters per key) interaction method. With this possibility included in the model, the letter-key assignments used in the SAK prototype yield $SPC = 1.713$. This is a 6.6% reduction in scan steps per character from the minimum value reported earlier ($SPC = 1.834$; see Figure 9).

This idea was considered so provocative that the SAK application was modified to implement a "timer restart on selection" mode. One participant agreed to do an extra five blocks (five phrases each) to test the mode. For these "extended" trials, the scanning interval started at 700 ms and was reduced by 50 ms per block, finishing at 500 ms. The results are shown in Figure 19.

The entry speeds ranged from 7.38 wpm in the 6^{th} block to 9.28 wpm in the 10^{th} block. The overall error rate for the extended trails was very low, at

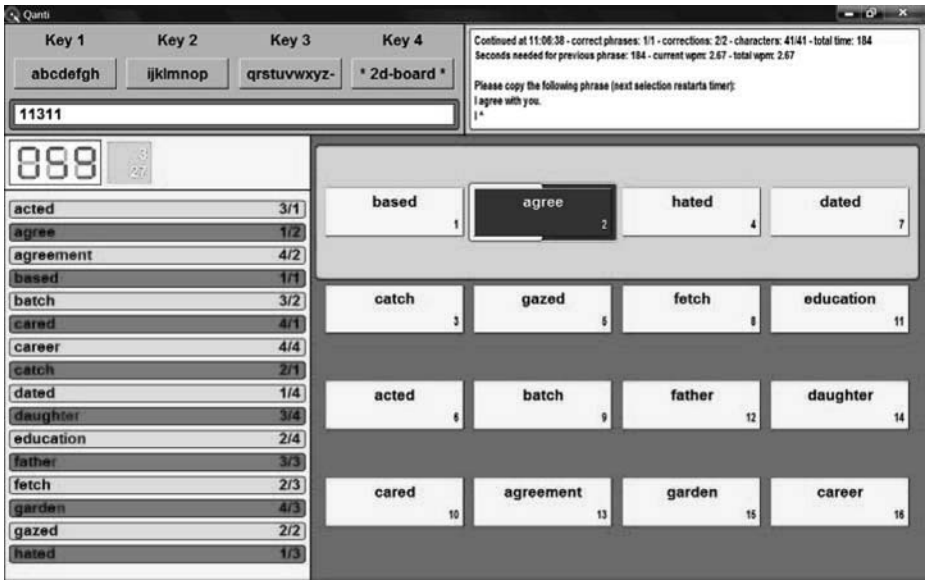

Fig. 20. Application window of the full implementation. See text for discussion.

0.81%. A look at the raw data revealed some use of multiple selections. Of 558 total letter selections, there were 98 (17.5%) double selections, 17 (3.0%) triple selections, and 3 (0.5%) quadruple selections.

5. EXPERIMENT 2: CASE STUDY

The experiment previously detailed confirms the viability of a scanning ambiguous keyboard (SAK). The question remains whether the SAK approach can indeed help the target population in everyday life. To investigate the practical usefulness of SAK, a case study with one 39-year-old male participant was conducted. The participant experiences considerable motor problems and has used a wheelchair for more than 20 years due to Friedreich's Ataxia (FA). Although he normally (still) uses a standard (manual) keyboard when interacting with a computer, the progressive nature of his disease makes use of a manual keyboard more difficult with time. His manual typing speed has progressively slowed. Previously measured speeds were 12 wpm at age 18, diminishing to 6 wpm at age 35.

As this study was intended to assume real-life conditions, it first required a full implementation of the SAK idea, including, for example, the entry of nondictionary words.

5.1 Software—Qanti

"*Qanti*"—a software tool for "Quick, Ambiguous, Non-standard Text Input" is a full implementation of the scanning ambiguous keyboard (SAK) concept described above. The interface is shown in Figure 20. The tool was developed in

C++ under Windows.® The *Qanti* interface divides the screen into four regions: a letter-selection region (top-left), an output region (top-right), an information region (bottom-left), and a multi-purpose, two-dimensional scanning board with 4 × 4 virtual keys (bottom-right).

5.1.1 *Letter-Selection Region.* *Qanti*'s letter-selection region operates similar to that of *OneKey*, the Java prototype SAK discussed earlier (see Figure 12). Initially, the focus is on the letter-selection region, with the four virtual keys linearly scanned as described earlier. The user composes text through keycode sequences by selecting the three leftmost letter keys (*with* timer restart on selection). The current keycode sequence is displayed under the four virtual keys. The fourth key allows switching focus to the two-dimensional scanning board or switching to an escape mode.

5.1.2 *Two-Dimensional Scanning Board.* In the default case, the two-dimensional scanning board receives focus when the user selects the fourth key in the letter-selection region having selected at least one letter key in the current scan cycle. The board presents up to 16 candidate words. In Figure 20, the user is entering "agree." The keycode sequence 11311 has been entered, followed by a selection on the fourth key, labeled "2d-board." A list of 16 candidates is presented and scanned.

The candidate words are drawn from two lists: first, a list containing words exactly matching the current keycode sequence, and second, words extending (completing) the keycode sequence. The order of the candidates is determined by the frequency of words in a dictionary (e.g., the 9022-word dictionary mentioned earlier). The distribution of the frequency-ordered list over the 16 keys takes the row-column scanning pattern into account (note the key indices in Figure 20).

The 4 × 4 board is scanned using traditional row-column scanning. The problem is that there are often more than 16 candidates matching or completing the current keycode sequence (59 for "11311" in Figure 20). To account for this, the scan cycle for the four rows is complemented by a fifth step (not shown) allowing the user to go down in the candidate list and display the words ranked 17–32, and so on.

After the selection of a candidate, the keys of the 2d-board are relabeled with various *modifiers*, which are again scanned in a row-column fashion, as shown in Figure 21. The modifiers allow, for example, appending a space character, a comma, or a period at the end, or capitalizing the first character of the candidate before the word is copied to the output region (with focus reverting to the letter-selection region for the entry of the next word).

5.1.3 *Escape Mode.* To "escape" the default way of entering a word, the user may select the fourth key in the letter-selection region *without* selecting any of the three letter keys in the same scan cycle (which unfortunately means a temporal overhead of at least four scan steps). In this case, the 2d-board presents an auxiliary menu, giving access to various services, for example, to delete the last character or word, to clear the current keycode sequence,

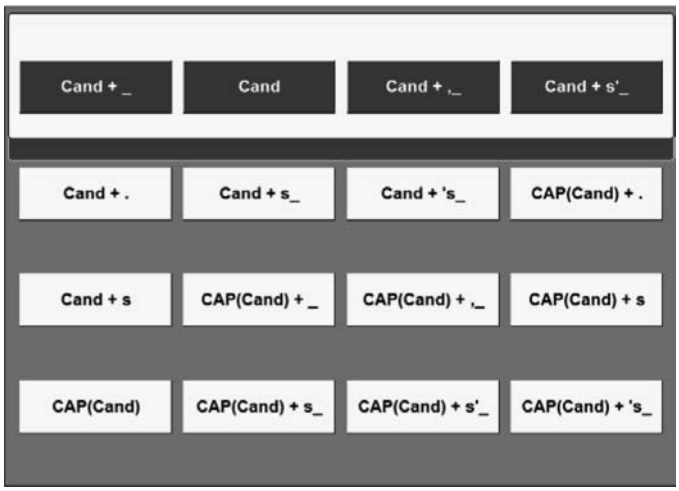

Fig. 21. After selecting a word, the two-dimensional scanning board changes to show modifier and word-ending options.

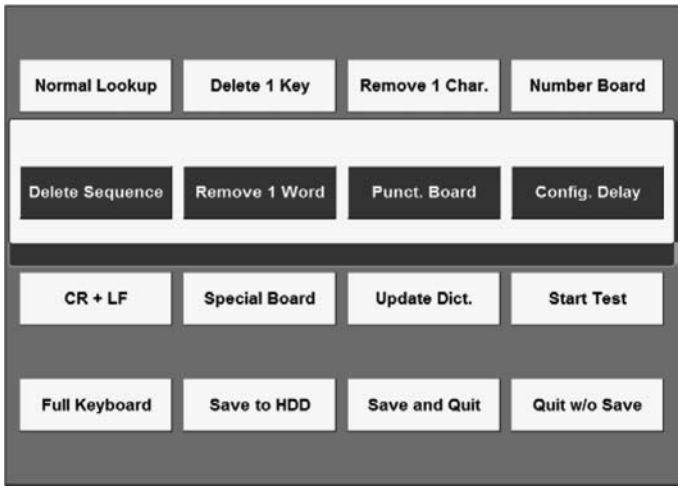

Fig. 22. Escape mode shows a menu providing access to a range of editing and system functions.

to configure the scanning interval, or to update the dictionary, as shown in Figure 22.

One menu function (bottom-left option in Figure 22) is responsible for launching a standard scanning keyboard for entering arbitrary character sequences as non-dictionary words which can subsequently be added to the dictionary. The implementation of the full keyboard (not shown) resembles the 3-level scanning text entry routine described by Felzer and Rinderknecht [2009] which features a 64-key on-screen keyboard. The full keyboard was accessed during the *Qanti* sessions when words containing punctuation were entered (e.g., "I'm," "don't," further explained below).

5.1.4 *Information Region.* The information region in *Qanti* contains all sorts of (sometimes redundant) context-dependent information to help the user in assessing the current program status. For example, in addition to displaying the total number of candidates, it lists the currently selectable ones in alphabetical order (also showing their row-column coordinates among the keys of the 2d-board). The aforementioned auxiliary menu also allows activating a *test mode* (used in the sessions described below). In the test mode, which is activated in Figure 20, the information region displays a timer with one-second resolution.

5.1.5 *Output Region.* The output region is where *Qanti* accumulates the entered text. Furthermore, any messages concerning user interaction in the test mode, such as the phrase to transcribe or the achieved statistical data, are displayed here.

5.2 Hardware

As *Qanti* is intended for users with severe physical disabilities, a specialized hardware configuration is required.

5.2.1 *Input Signals.* An important difference between *OneKey* and *Qanti* is the type of input signals required for issuing selections. *OneKey* is operated by pressing any key on the standard keyboard. This is sufficient for test-of-concept evaluations with able-bodied subjects, yet it still requires use of the hands.

On the contrary, while also reacting to pressing the SPACE key or mouse button clicks, *Qanti* is configured to operate through *intentional muscle contractions* [Felzer and Nordmann 2008; Felzer et al. 2009]. These input signals only require a tiny contraction of a single muscle of choice (which requires a minimum of physical effort) and are more suitable for members of the target population (i.e., persons with severe physical disabilities).

5.2.2 *Setup.* Figure 23 depicts the experimental setup used in the two *Qanti* sessions.

The subject was positioned a comfortable distance from a laptop screen and produced selections in a hands-free manner using a headband sensor. The sensor is based on a piezo element, which actively generates a voltage when deformed by the participant contracting the brow muscle intentionally. This is similar to the common act of frowning. The voltage signal is filtered, amplified, converted from analog to digital in an interface circuit, and sent to the computer via the USB port. Additional processing software "listens" to the USB and notifies *Qanti* when it detects a contraction event.

5.3 Test Sessions

To quantify the usefulness of the SAK approach in general and the usability of the *Qanti* software in particular, the FA patient mentioned above performed several sessions of text entry. The tests were distributed over three consecutive

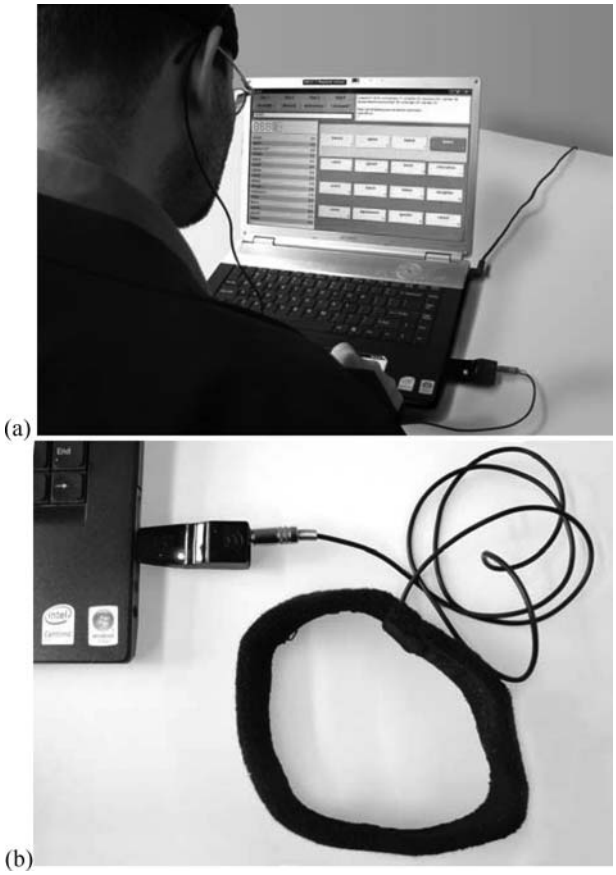

Fig. 23. Case study setup. (a) Participant and apparatus. (b) Headband sensor. See text for discussion.

days, with sessions lasting between ten minutes (four on the third day) and one hour (one on each of the first and second days).

5.3.1 *Day One.* Testing on the first day was fashioned after that with the able-bodied participants in Experiment 1. Using *Qanti*, the participant performed blocks of text entry with five phrases per block. The scanning interval was initially 1100 ms, and decreased by 100 ms per block.

Although the time to enter a phrase—starting with the first selection, ending when the phrase was finalized by entering a newline character—was recorded, a high entry rate was not overly important in the initial session. The main objective was to familiarize the participant with *Qanti* (as well as the expected task) and to identify a favorable scanning interval. A favorable scanning interval is one deemed comfortable (not too slow, not too fast) that would facilitate the production of optimal entry rates for the second day.

5.3.2 *Day Two.* *Qanti* was also used on the second day. The test involved five blocks with five phrases each, all with the preferred scanning interval,

as identified in the initial session. This time, the participant was asked to transcribe the phrases as quickly and accurately as possible, and also to correct any errors noticed. Note that the method of correcting errors was quite different with *Qanti* than with *OneKey*. *OneKey* used a "long press" for error correction (described in Section 3.2.2). While this may work well for able-bodied users, the motor control required to issue long vs. short selection signals may not be possible for many disabled users. Error correction with *Qanti* was implemented with dedicated commands accessible through the escape mode. Examples are seen in Figure 21.

There were sufficient rest intervals between the phrases and between the blocks, and the participant was always free in deciding when to move on.

5.3.3 *Day Three.* The sessions on the third day served comparison purposes. One session was devoted to *OneKey* and was performed using the same software and setup as with Experiment 1. Using manual input (the SPACE key of the physical keyboard), the participant performed one block of five phrases with his preferred scanning interval. This was followed by three sessions, five phrases each, with his usual method: manual, with the standard keyboard. The sessions were over the entire day (allowing for the observation that the participant's typing speed varied considerably depending on the time of day): one in the morning, one at noon (before lunch), and one in the evening.

In all, there were 13 blocks (five phrases each) distributed over three days as follows:

—Day 1—Four familiarization blocks with *Qanti* (plus a fifth attempted but unfinished block; see below)
—Day 2—Five blocks with *Qanti*
—Day 3—One block with *OneKey*, three blocks with a manual keyboard

5.3.4 *Phrase Set and Dictionary.* The phrases used in the test sessions were similar to those in the *OneKey* experiment, albeit modified to reflect "real-life" conditions. The modifications included capitalizing the first character of the first word, inserting grammatically correct commas, etc., and appending the logically intended terminating punctuation (period, exclamation mark, or question mark). In addition, only complete sentences were used, and expressions like "do not" were converted into "don't." The implication of this for *Qanti* is that the participant had to occasionally escape from the default mode of input using the scanning ambiguous keyboard and use the conventional full (row-column scanning) keyboard to build-up non-dictionary words that included letters and punctuation characters.

In all, 100 phrases were selected and modified as described above. For Day 1 and Day 2, phrases were drawn at random from the set. The last five phrases from Day 2 were selected for use on Day 3. Since the *OneKey* prototype software was not designed to handle punctuation or contractions (e.g., "don't"), the five phrases were used in their unmodified form. For example, "I'm allergic to bees and honey!" was reduced to "I am allergic to bees and honey." These same five

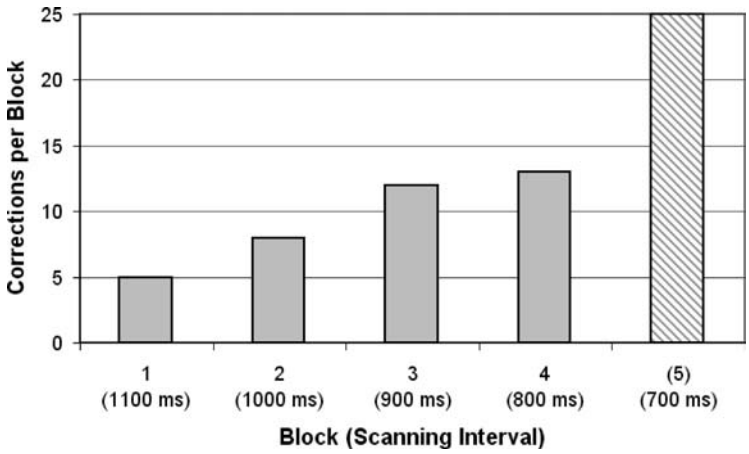

Fig. 24. Required corrections in the *Qanti* familiarization phase. See text for discussion.

phrases, but in their modified form, were used in the three sessions with the manual keyboard.

The same dictionary was used with *Qanti* as was used with *OneKey* in Experiment 1.

5.4 Results and Discussion

Unlike the former *OneKey* experiment, the familiarization session comprised only four complete blocks. This was due to the delayed reaction time of the participant. He was not able to enter more than two consecutive words correctly with a scanning interval of 700 ms, and, consequently, gave up completing the 700 ms block after eight unsuccessful trials (meaning that this block—and *only* this block—was restarted again and again).

The described problem is also seen when looking at the total number of corrections per block. A *correction* in this sense was defined as any operation taking back words or characters either within the entered text or in the current keycode sequence. A correction was only utilized if there had been a preceding mistake (i.e., an erroneous selection). As seen in Figure 24, the total number of required corrections gradually increases with decreasing scanning interval and "explodes" in the block with the 700 ms scanning interval where the participant was unable to complete any phrases.

The participant finally decided on 1000 ms as the most comfortable scanning interval. This speed configuration was used throughout all subsequent scanning sessions.

5.4.1 *Achieved Entry Speed.* When looking at the achieved entry speed in the 9 blocks of testing with *Qanti* (Figure 25), it is immediately apparent that the rate in the familiarization blocks—even in the second one with the same scanning interval (i.e., 1000 ms)—is generally lower than in the main test blocks. This occurred because the participant was told in the familiarization

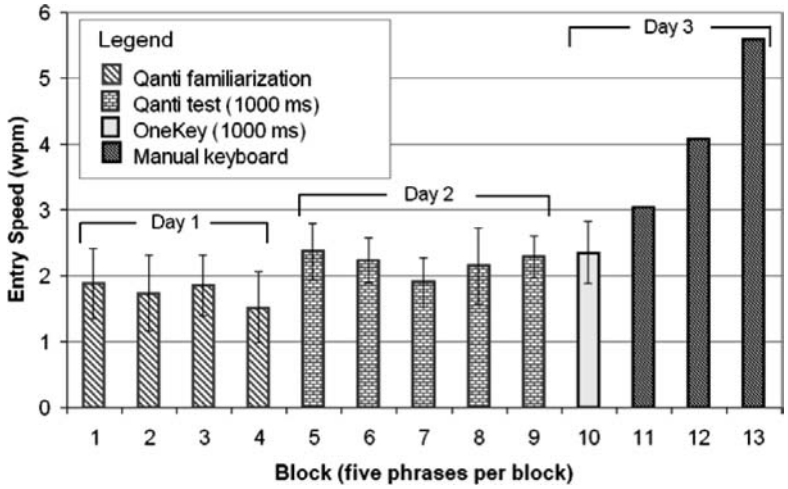

Fig. 25. Achieved entry speed during the case study. See text for discussion.

blocks not to concentrate on being fast, but on becoming acquainted with the software and with the setup.

An interesting result is that the participant was about as fast with *OneKey* as with *Qanti*—a little over 2 wpm. Given that the phrases in the *OneKey* session were simpler (and *never* required a full row-column scanning keyboard), one would probably expect *OneKey* to elicit significantly superior entry rates. The mentioned advantages were leveled out by the more convenient input method in the *Qanti* test (i.e., use of the headband sensor). Especially, the DLPK/MLPK interaction methods caused problems for the participant while using *OneKey*, as these methods involve pressing and releasing a physical key multiple times in quick succession.

When comparing the entry rates achieved with *Qanti* to those achieved with *OneKey* in Experiment 1, one might think the *Qanti* rates were "slow." However, the less favorable results are hardly attributable to *Qanti*. Rather, the main reason is related to the test participant's health condition: Experiment 1 involved able-bodied participants, while the case study results were achieved by a member of the target population (i.e., someone severely disabled). See also the result in block 10, where *Qanti* was *not* used at all. In addition, the practical "real-life" assumptions mentioned above (e.g., grammatically correct punctuation characters or newline characters between sentences) make the two experiments hard to compare, especially in view of the request to correct any noticed errors. Notably, all transcribed phrases in the case study were, in the end, 100% correct, and this naturally reduced the achieved entry rate.

5.4.2 *Subjective Assessment.* Looking at the entry speed in the manual sessions raises the question of whether the participant was interested in using *Qanti* as an alternative, given that he was faster with the keyboard in all three manual sessions. The answer was a definite "yes" for two reasons: the expected

exercising effect and his personal evolution. What he meant is explained as follows.

First, *Qanti* poses a high cognitive load on the user, but this is expected to diminish with practice. For example, after entering "11311," the word "education" appears in the candidate list in the second row and the fourth column (see Figure 20). When entering "education" for the first time, it is almost impossible to take all optimization opportunities; i.e., to observe the appearance of the candidate at the earliest opportunity *and* to consciously issue a selection at the right time. The test participant indicated that, with practice, it would be possible to know and anticipate the coordinates of most candidate words. Although this would take extended training, in the end *Qanti* is expected to beat the manual morning-session (block 11).

Second, because of the participant's diminishing manual typing speed with age (noted earlier), he is likely to become slower than 2 wpm in a few years. Besides, people with other disabilities, who cannot use a manual keyboard at all are likely to benefit by the overall efficiency of the SAK concept, provided a single-switch input signal can be generated.

One additional note: the same participant that participated in the case study has participated in about a dozen evaluations with alternative text entry applications, ranging from a simple on-screen keyboard to a multi-tier scanning application with frequency-based character ordering [Felzer and Nordmann 2006a; Felzer and Nordmann 2006b; Felzer et al. 2008], Most of the evaluations involved the hands-free mouse emulator of Felzer and Nordmann [2008], and all relied on intentional muscle contractions as input signals. Therefore, one could safely say that the participant was quite experienced with a range of alternative and competing text entry methods. However, all "prior-SAK" tests hardly exceeded an entry rate of 1 wpm. Doubling that value "off the top of one's head" (i.e., without extensive training) is a tremendous improvement.

6. CONCLUSIONS

This article introduced SAK, a scanning ambiguous keyboard supporting text entry using a single key, button, or switch for input. SAK involves scanning virtual keys on a display with multiple letters assigned to each key. The design combines the most demanding requirement of a scanning keyboard—input using a single key or switch—with the most appealing feature of an ambiguous keyboard—one switch activation per character. A model was built and run to search for designs that minimize *SPC*—the number of scan steps per character required on average for English text entry. To avoid high cognitive demands, only designs using an alphabetic arrangement of letters were considered. The design yielding the lowest scan steps per character, $SPC = 1.713$, placed letters on three keys arranged as abcdefgh-ijklmnop-qrstuvwxyz. A fourth "SPACE" key is used to terminate letter selection and transfer scanning to a word-selection region.

As well as combining scanning with ambiguous virtual keys, as just described, SAK is novel in at least two other ways. First, the scanning pattern in

the letter-selection region continues is a cyclic pattern, even after character selection. Existing scanning keyboards restart scanning at a home position after a character is selected. This is a necessary by-product of multi-tier selection, which is not used in SAK designs. Second, SAK designs allow two or more key activations in a single scanning interval if multiple consecutive letters are on the same key.

An evaluation of the optimal three-letter-key design was conducted using 12 participants who entered text phrases drawn at random from a set. Entry was organized as five blocks with five phrases per block, and lasted about one hour for each participant. The scanning interval was 1100 ms initially, decreasing by 100 ms per block, finishing at 700 ms. The average text entry rate in the 5^{th} block was 5.11 wpm. Three participants reached average rates above 6 wpm, with one reaching 8.05 wpm. Furthermore, the accuracy in the transcribed text was very high at 99% overall. This was largely because the methodology allowed and encouraged participants to correct selection errors.

Participants' scanning efficiency—the ratio of the minimum number of scan steps possible to the observed number of scan steps—was only 59% in the 5^{th} block. Although this was partly due to missed opportunities to optimize (i.e., failing to make selections at the earliest opportunity), the main cause was the additional scan steps accompanying selection errors (which were subsequently corrected). For the SAK tested, the model predicts an upper bound text entry rate of 9.35 wpm with a 700 ms scanning interval.

After the experiment, the design was modified to use "timer restart on selection," thus facilitating > 2 selections per scan step. With this modification, one participant performed an additional five blocks of input and reached an average entry rate of 9.28 wpm in the last block (SI = 500 ms).

A case study with a disabled user provided further verification of the viability of the SAK concept. The user wore a headband keeping a piezo element in contact with his forehead. Input "keystrokes" were generated using an eyebrow motion similar to a frown. The case study involved text entry using *Qanti*, an implementation of a SAK that included a full range of editing and application commands. Despite the user's substantial motor problems, entry rates a little over 2 wpm were attained. Qualitative results were also promising. The user felt that considerable further improvement in entry speed was likely since, with continued practice, opportunities to optimize would become apparent.

The SAK concept represents a novel and effective means for text entry using a single key or switch. While target applications include mobile computing, ubiquitous or wearable computers, or gaming, the most likely application is accessible interfaces for disabled users who are constrained to single-switch input to computers. Such users can benefit from the efficient text entry possible with a scanning ambiguous keyboard.

ACKNOWLEDGMENTS

The authors wish to thank Foad Hamidi, Farzan Sasangohar, Fraser Shein, Rob Teather, and Hussain Tinwala for helpful discussions on scanning keyboards, the SAK design, and the testing methodology.

REFERENCES

BALJKO, M. AND TAM, A. 2006. Indirect text entry using one or two keys. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'06)*. ACM, New York, 18–25.

BELLMAN, T. AND MACKENZIE, I. S. 1998. A probabilistic character layout strategy for mobile text entry. In *Proceedings of Graphics Interface'98*. Canadian Information Processing Society, 168–176.

BHATTACHARYA, S., SAMANTA, D., AND BASU, A. 2008a. Performance models for automatic evaluation of virtual scanning keyboards. *IEEE Trans. Neu. Syst. Rehab. Engin. 16*, 510–519.

BHATTACHARYA, S., SAMANTA, D., AND BASU, A. 2008b. User errors on scanning keyboards: Empirical study, model and design principles. *Interact. Comput. 20*, 406–418.

BRANDENBERG, S. AND VANDERHEIDEN, G. C. 1988. Communication board design and vocabulary selection. In *The Vocally Impaired: Clinical practice and research*, (3rd Ed.), L. Bernstein, Ed. Allyn & Bacon, Needham Heights, MA, 84–135.

DAMPER, R. I. 1984. Text composition by the physically disabled: A rate prediction model for scanning input. *Appl. Ergon. 15*, 289–296.

DOUBLER, J. A., CHILDRESS, D. S., AND STRYSIK, J. S. 1978. A microcomputer-based control and communication system for the severely disabled. In *Proceedings of the ACM Conference on Computers and the Physically Handicapped*. ACM, New York, 43–46.

FELZER, T. AND NORDMANN, R. 2006a. Alternative text entry using different input methods. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'06)*. ACM, New York, 10–17.

FELZER, T. AND NORDMANN, R. 2006b. Speeding up hands-free text entry. In *Proceedings of the 3rd Cambridge Workshop on Universal Access and Assistive Technology (CWUAAT'06)*. Cambridge University Press, Cambridge, UK, 27–36.

FELZER, T. AND NORDMANN, R. 2008. Evaluating the hands-free mouse control system: An initial case study. In *Proceedings of the 11th Conference on Computers Helping People With Special Needs (ICCHP'08)*. Springer, 1188–1195.

FELZER, T., NORDMANN, R., AND RINDERKNECHT, S. 2009. Scanning-based human-computer interaction using intentional muscle contractions. *Universal Access in Human-Computer Interaction* Part II*, (HCII'09)*. Springer, 309–318.

FELZER, T. AND RINDERKNECHT, S. 2009. 3dScan: An environment control system supporting persons with severe motor impairments. In *Proceedings of the ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'09)*. ACM, New York, 213–214.

FELZER, T., STRAH, B., AND NORDMANN, R. 2008. Automatic and self-paced scanning for alternative text entry. In *Proceedings of the 4th IASTED International Conference on Telehealth and Assisstive Technologies (Telehealth/AT'08)*. IASTED Secretariat, Calgary, Alberta.

HARBUSCH, K. AND KÜHN, M. 2003a. An evaluation study of two-button scanning with ambiguous keyboards. In *Proceedings of the 7th European Conference for the Advancement of Assistive Technology (AAATE'03)*. AAATE c/o Danish Centre for Assistive Technology Taastrup, Denmark, 954–958.

HARBUSCH, K. AND KÜHN, M. 2003b. Towards an adaptive communication aid with text input from ambiguous keyboards. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*. MIT Press, Cambridge, MA, 207–210.

HECKATHORNE, C. W. AND CHILDRESS, D. S. 1983. Applying anticipatory text selection in a writing aid for people with severe motor impairment. In *IEEE Micro*, 17–23.

HOCHSTEIN, D. D. AND MCDANIEL, N. 2004. Recognition of vocabulary in children and adolescents with cerebral palsy. *Augment. Altern. Comm. 20*, 45–62.

JONES, P. E. 1998. Virtual keyboard with scanning and augmented by prediction. In *Proceedings of the 2nd European Conference on Disability, Virtual Reality and Associated Technologies*. University of Reading, UK, 45–51.

KUCERA, H. AND FRANCIS, W. N. 1967. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

KUSHLER, C. 1998. AAC using a reduced keyboard. In *Proceedings of the CSUN Technology and People With Disabilities Conference*. California State University Northridge, CA, 140–145.

LESHER, G., HIGGINBOTHAM, D. J., AND MOULTON, B. J. 2000. Techniques for automatically updating scanning delays. In *Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America (RESNA'00)*. RESNA, Arlingtion, VA, 85–87.

LESHER, G., MOULTON, B., AND HIGGINBOTHAM, D. J. 1998. Techniques for augmenting scanning communication. *Augment. Altern. Comm. 14*, 81–101.

LESHER, G., MOULTON, B., HIGGINBOTHAM, J., AND BRENNA, A. 2002. Acquisition of scanning skills: The use of an adaptive scanning delay algorithm across four scanning displays. In *Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America (RESNA'02)*. RESNA, Arlington, VA, 75–77.

LIN, Y.-L., CHEN, M.-C., WU, Y.-P., YEH, Y.-M., AND WANG, H.-P. 2007. A flexible on-screen keyboard: Dynamically adapting for individuals needs. In *Universal Access in Human-Computer Interaction. Applications and Services*. Springer, Berlin, 371–379.

LIN, Y.-L., WU, T.-F., CHEN, M.-C., YEH, Y.-M., AND WANG, H.-P. 2008. Designing a scanning on-screen keyboard for people with severe motor disabilities. In *Proceedings of the 11th Conference on Computers Helping People With Special Needs (ICCHP'08)*. Springer, Berlin, 1184–1187.

MACKENZIE, I. S. 2002a. KSPC (keystrokes per character) as a characteristic of text entry techniques. In *Proceedings of the 4th International Symposium on Human-Computer Interaction with Mobile Devices (MobileHCI'02)*. Springer, Berlin, 195–210.

MACKENZIE, I. S. 2002b. Mobile text entry using three keys. In *Proceedings of the 2nd Nordic Conference on Human-Computer Interaction (NordiCHI'02)*. ACM, New York, 27–34.

MACKENZIE, I. S. AND SOUKOREFF, R. W. 2002. A character-level error analysis technique for evaluating text entry methods. In *Proceedings of the 2nd Nordic Conference on Human-Computer Interaction (NordiCHI'02)*. ACM, New York, 241–244.

MACKENZIE, I. S. AND SOUKOREFF, R. W. 2003. Phrase sets for evaluating text entry techniques. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems (CHI'03)*. ACM, New York, 754–755.

MACKENZIE, I. S. AND TANAKA-ISHII, K. 2007. Text entry with a small number of buttons. In *Text entry systems: Mobility, accessibility, universality*, I. S. Mackenzie, and Tanaka-Ishii, K., Ed. Morgan Kaufmann, San Francisco, 105–121.

MIRÓ, J. AND BERNABEU, P. A. 2008. Text entry system based on a minimal scan matrix for severely physically handicapped people. In *Proceedings of the 11th Conference on Computers Helping People with Special Needs (ICCHP'08)*. Springer, Berlin, 1216–1219.

PAVLOVYCH, A. AND STUERZLINGER, W. 2003. Less-Tap: A fast and easy-to-learn text input technique for phones. In *Proceedings of Graphics (Interface'03)*. Canadian Information Processing Society, Toronto, 97–104.

RAU, H. AND SKIENA, S. 1994. Dialing for documents: An experiment in information theory. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*. ACM, New York, 147–154.

RYU, H. AND CRUZ, K. 2005. LetterEase: Improving text entry on a handheld device via letter reassignment. In *Proceedings of the 19th Conference of the Computer-Human Interaction Special Interaction Group (CHISIG) of Australia (OZCHI'05)*. ACM, New York, 1–10.

SHEIN, F., GALVIN, R., HAMANN, G., AND TREVIRANUS, J. 1994. Scanning the Windows desktop without mouse emulation. In *Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America (RESNA'94)*. RESNA, Arlingtion, VA, 391–393.

SHEIN, F., HAMANN, G., BROWNLOW, N., TREVIRANUS, J., MILNER, D., AND PARNES, P. 1991. WiVik: A visual keyboard for Windows 3.0. In *Proceedings of the Annual Conference of the Rehabilitation Engineering Society of North America (RESNA'91)*. RESNA, Arlington, VA, 160–162.

SHEIN, G. F. 1997. Towards task transparency in alternative computer access: Selection of text, PhD Dissertation, University of Toronto.

SILFVERBERG, M., MACKENZIE, I. S., AND KORHONEN, P. 2000. Predicting text entry speed on mobile phones. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI'00)*. ACM, New York, 9–16.

SIMPSON, R. C. AND KOESTER, H. H. 1999. Adaptive one-switch row-column scanning. *IEEE Trans. Rehab. Engin. 7*, 464–473.

SMITH, S. L. AND GOODWIN, N. C. 1971. Alphabetic data entry via the touch tone pad: A comment. *Hum. Factors 13*, 189–190.

SOUKOREFF, R. W. AND MACKENZIE, I. S. 2001. Measuring errors in text entry tasks: An application of the Levenshtein string distance statistic. In *Extended Abstracts of the ACM Conference on Human Factors in Computing (CHI'01)*. ACM, New York, 319–320.

STERIADIS, C. E. AND CONSTANTINOU, P. 2003. Designing human-computer interfaces for quadriplegic people. *ACM Trans. Comput.-Hum. Interac. 10*, 87–118.

TREVIRANUS, J. AND TANNOCK, R. 1987. A scanning computer access system for children with severe physical disabilities. *Amer. J. Occup. Ther. 41*, 733–738.

TRNKA, K., MCCAW, J., YARRINGTON, D., MCCOY, K. F., AND PENNINGTON, C. 2009. User interaction with word prediction: The effects of prediction quality. *ACM Trans. Access. Comput. 1*, 1–34.

VANDERHEIDEN, G. C., VOLK, A. M., AND GEISLER, C. D. 1974. An alternative interface to computers for the physically handicapped: The auto-monitoring communication board. In *Proceedings of the AFIPS Joint Computer Conference*. ACM, New York, 115–120.

VENKATAGIRI, H. S. 1999. Efficient keyboard layouts for sequential access in augmentative and alternative communication. *Augment. Altern. Comm. 15*, 126–134.

WANDMACHER, T., ANTOINE, J.-Y., POIRIER, F., AND DEPART, J.-P. 2008. Sibylle, An assistive communication system adapting to the context and its user. *ACM Trans. Access. Comput. 1*, 1–30.

WOBBROCK, J. O., MYERS, B. A., AND CHAU, D. H. 2006. In-stroke word completion. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'06)*. ACM, New York, 333–336.

YAMADA, H. 1980. A historical study of typewriters and typing methods: From the position of planning Japanese parallels. *J. Inform. Process. 2*, 175–202.

ZHAI, S. AND KRISTENSSON, P.-O. 2003. Shorthand writing on a stylus keyboard. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'03)*. ACM, New York, 97–104.

Received December 2008; revised September 2009, March 2010; accepted March 2010 by Shumin Zhai