An Analysis & Design Framework for One Bit Text Entry Systems

Name of First Author and Name of Second Author

Abstract Information and communication devices have become ubiquitous and have impacted our everyday lives. These devices, however, do not sufficiently meet the growing needs of the aging, potentially disabled, population. Thus several novel text entry systems have been designed and developed to meet the growing needs of aged or disabled people. In this paper, we proposed an analysis and design framework for simple text entry systems, which require a minimum input signal, one bit, for text entry. Two optimized reference designs are drawn based on this framework, which brings combinations of high efficiency, low ambiguity and easy learnability into the one-bit text entry. User studies with native English speakers and real patients were conducted. The results proved the system to be both highly efficient and easy to use.

1 Introduction

The proportion of aged people in the total population is increasing over time. Although advances in biomedical sciences are extending chronological life, quality of life tends to decline with advancing age. Disability status, the major determinant of quality of life, is not included in most conventional measures of population aging [1]. Increased old-age dependency ratios (OADR) in recent decades, as explained in [2], suggest an increasing proportion of a person's lifetime is burdened with disabilities or diseases.

Parkinson's disease (PD), for example, occurs in 0.5-5% of the population over the age of 65 [3, 4]. One of the secondary symptoms of PD is low volume speech,

Name of First Author
Name, Address of Institute, e-mail: name@email.address

Name of Second Author
Name, Address of Institute e-mail: name@email.address

and depending on its severity, the loss of the ability to speak, to write or both [15]. Difficulties with communication often leads anxiety, isolation and depression. Many other age-dependent conditions, such as stroke and neurodegenerative diseases, can impair communication and movement. Therefore, to stay in contact with others and to keep an existing relationship or even to build up a new one is difficult for the disabled. The development of information technology with a focus on Augmentative and Alternative Communication using common usability standards and norms such as the Dialogue Principle of ISO 9241-110 will be highly applicable to persons with communication or physical disabilities [7]. Electronic devices or communication paths should compensate for the communication deficits exhibited by a large fraction of the aging population [6, 14].

The One Bit text entry allows the user to type in letters and words by pressing or touching just one button without using a regular keyboard. The goal is to provide an easily usable text entry tool to people who suffer from diseases or are physically handicapped. A user can express his/her idea with a mere finger or thinking via brain-computer interface, which is the smallest piece of information required for a person to communicate with the world outside.

2 Related Work

We have seen many kinds of common keyboards, ranging from the typical 104-key computer keyboard to 12-key cell phone keyboard. With the purpose of reducing the number of keys on a keyboard for the aged population, one bit text entry systems were developed.

Prevalent one bit text entry system can be categorized as two kinds: scanning keyboard (SK) and scanning ambiguous keyboard (SAK). SK is modified from an ordinary keyboard, say an QWERTY keyboard, simply adding a scanning function. Texts are inputted letter by letter, which of course, means very low efficiency. SAK added ambiguity to scanning keyboards by reducing the number of keys and rearranging characters on the virtual keyboard. It increased the speed of the text entry, and allowed users to input text word by word.

2.1 Scanning Keyboard

In Microsoft Windows, there is a hidden “Windows OnScreen-Keyboard” (Fig 1). This OnScreen-Keyboard realizes one-bit text entry by allowing the users to press just one actual key to type sentences on screen. This design of the keyboard is known as a scanning keyboard.

When the SK is loaded, the QWERTY keyboard appears and the rows are highlighted one by one. By only one key pressing, the highlighted row is selected and the SK is switched to column (or columns in the SK of Windows 7) scanning from

left to right. With one more key pressing, the desired character is selected. In this way users can input texts character by character, as using a normal keyboard. It is possible to change the settings, i.e. changing the time of the highlighting intervals or to highlight more keys at the same time in a row in order to shorten the waiting time and reach the required key quicker.

This scanning keyboard demonstrated a way of one bit input keyboard. However, it is inefficient to traverse all the rows and columns over the keyboard and input a word letter by letter. In order to solve this problem, scanning ambiguous keyboards are designed for one bit text entry.

2.2 Scanning Ambiguous Keyboard

Phone-like keyboards (Fig 2), known as ambiguous keyboards, have been developed for the mobile phones with various designs.

The ambiguous keyboards, on one hand, reduce the number of keys for saving the space on the screens of mobile devices; on the other hand, have three to four letters on each key, leading to an ambiguous choosing for each key pressing.

To learn from ambiguous keyboards, various scanning ambiguous keyboards (SAK) have been proposed and developed [8, 9, 10, 11]. SAK has the same feature with SK, which is scanning among the keys on a keyboard, but it also reduces the

Fig. 1 On-Screen scanning keyboard in Microsoft Windows XP

Fig. 2 Ambiguous keyboard of phones

number of keys on a keyboard thus adding ambiguity into the keys. SAK combines the core features of the scanning keyboard and ambiguous keyboards. In order to find out how many keys should be used in SAK, MacKenzie [9] built a model to search for designs that minimize the average number of Scan steps per Character (SPC) required for text entry in a given language. He figured out the letter assignment with the lowest SPC is dividing the characters into three groups, as shown in Fig 3.

This version designed by MacKenzie seems to be perfect except that it overlooked the size of the dictionary, which is really small, with 9025 words in use. As the dictionary size increases, the efficiency of this SAK goes down quickly, since the candidate queue of word list becomes very long, and it will take much more time for users to reach the word they want.

Given a relatively small dictionary, a specific design of SAK might seem efficient, though it may be ambiguous and inefficient when the dictionary is enlarged. So we indicate that a good design of SAK should reach the balance among three essential elements: Efficiency (relative to the number of keys), Ambiguity (relative to the size of dictionary) and Learnability (relative to the usability of users).

3 A Framework of SAK design

To balance the three essential elements, we proposed a unified framework for one bit based SAK design. The scanning of 26 English characters in a SAK can be categorized into three classes: one-dimensional, two-dimensional and three-dimensional scanning keyboard (1D-SAK, 2D-SAK, and 3D-SAK), as shown in Fig 4. 1D-SAK puts all the M keys in a column, and the highlight moves one-by-one from up to down over the keys. This keyboard can also be designed as a rolling wheel with a visible window. The keys move through the visible window one-by-one when the keyboard wheel rolls.

2D-SAK organizes the keys into an $M \times N$ array. The scanning of the keys is first row-by-row then column-by-column or vise versa. The $M \times N$ array of keys can be further sorted into a multi-layered array, which is a three-dimensional scanning keyboard (3D-SAK). Its dimension is $M \times N \times L$. A key selection should have three times of key-press in this case.

In the framework, higher dimensional keyboards have the ability of containing more keys, which lead to less ambiguity, but more times of key-press for selecting a key. A three-dimensional $3 \times 3 \times 3$ scanning keyboard has 27 keys. In this case, there is no ambiguity in letter selection, but it is too complex to learn for the disabled

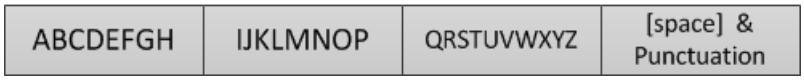

Fig. 3 Scanning ambiguous keyboard with four keys proposed in [9].

and senior users. Relatively, 1D-SAK and 2D-SAK are simple, intuitive and easy to use, where learnability is a key element in SAK systems referring to the fact that the keyboard is designed for disabilities and seniors.

Based on this framework, we proposed two reference SAK designs: one- and two-dimensional scanning ambiguous keyboard. The 3D-SAK is too complex to learn and to use for the target group.

3.1 Reference design 1: 1D-SAK

For one dimensional keyboards, the least number of keys is two with 13 letters on each key. Certainly, this SAK gives great ambiguity to the word list. The other extreme design is 26 keys in a row or column with one letter on each key. The question is: how many keys make the best design for 1D-SAK? The answer is: it depends on the size of the database of the word list. In the case of 9000 words, MacKenzie [8] suggested three keys were good. We also explored the various designs on the number of key and suggested that 5 keys are the minimum for 1D-SAK for balancing the ambiguity and the efficiency when the database contains more words ($>20,000$). Considering the letter frequency in English words and the learnability, we put 5 letters on each keys except the last key, which has 6 letters.

The scanning of 1D-SAK is from top to bottom, and jumps back to the first key when the scanning reaches the last key. This scanning can be intuitively imaged as a rolling wheel with a visible window. The user can use 1D-SAK without any

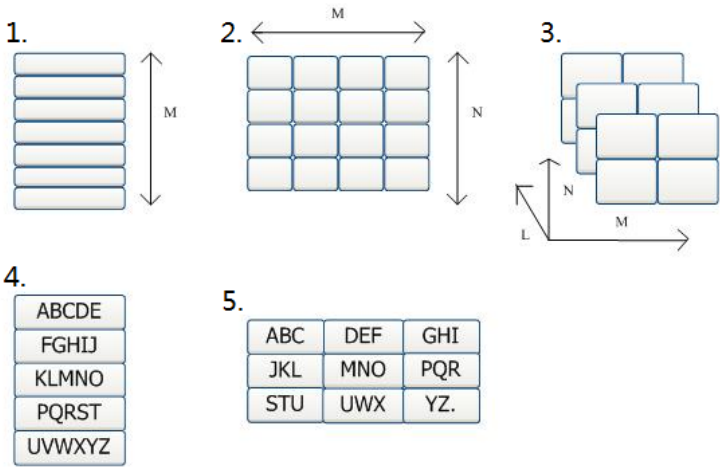

Fig. 4 The framework for scanning ambiguous keyboard. 1. One-dimensional SAK; 2. Two-dimensional SAK; 3. Three-dimensional SAK; 4. A reference design of one-dimensional SAK; 5. A reference design of two-dimensional SAK.

learn, just presses a button when he/she sees the desired letters appear on the visible window.

3.2 Reference design 2: 2D-SAK

2D-SAK, in general, increases the number of keys comparing to 1D-SAK. A key selection in the two-dimensional array of keys needs two times of key pressing. In this case, both 2×2 and 2×3 array 2D-SAK have the similar number of keys with the 1D-SAK, which increases the times of key pressing in key selection but does not decrease the ambiguity comparing to the reference design of 1D-SAK. Therefore, the 3×3 array of keys is the optimized design of 2D-SAK. It allows more keys ($3 \times 3 = 9$) than 1D-SAK, which lowers the ambiguity. And it is the fewest keys in 2D-SAK that guarantees the low ambiguity, since more keys means more time spent for key selection. There is no better choice to balance all three essential elements in 2D-SAK design.

In text entry, users first select one row that contains the desired character, then choose the right column. The selection narrows down to 3 characters, which has a lower ambiguity than 1D-SAK.

The essential aspect of 2D-SAK is one key selection with two key-pressing. In this sense, we further designed a doubled 1D-SAK, which give 2D-SAK a "one dimensional look" (Fig 5), which makes the two dimensional keyboard easier to learn in practical use. The three keys on the left in Fig 5 makes it looks intuitive with simple roll scanning. This facilitates better learnability, as the mere three keys does not look so frightening.

3.3 The Theoretical efficiency of SAK

Based on the framework of SAK design, various SAK could be proposed and designed to meet the special need of users. To compare the efficiency of SAKs theoretically, we launched a method to calculate the average time per letter, which is

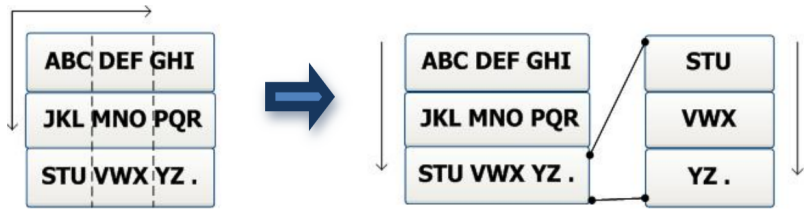

Fig. 5 Doubled 1D-SAK. Arrows in the figure are the directions of scanning of SAKs. Left: a 3×3 2D-SAK; Right: a 3×3 doubled 1D-SAK.

the time needed to input a letter in SAKs. This is a general method for estimating the efficiency of various SAKs, and we presented the examples of calculating the average key selection time of reference designed SAKs.

The mean time per letter t_c can be calculated as in Equation 1:

$$t_c^{1D} = \tau \times \sum_{0}^{N_{key}} P_{key} n_{key} \tag{1}$$

where

$$P_{key} = \sum P_{letter} \tag{2}$$

is the sum of the English letter frequency[1] P_{letter} of the letters in one key (Table 1). n_{key} is the position of key in the virtual keyboard starting at zero ($n_{key} = 0, \cdots, M_{key}$); τ is the scanning interval for the virtual keyboards, which is usually set as one to few seconds. Let t_{click} be the average time of user's making a click on his/her one-bit input device, then

$$0 < t_{click} < \tau \tag{3}$$

Equation 4 is used to estimate the mean input time per letter t_{input},

$$t_{input} = t_c^{1D} + t_{click} \tag{4}$$

in case of 1D-SAK.

The mean time per letter of an $M \times N$ 2D-SAK can be calculated as:

$$t_c^{2D} = \tau \times \sum_{0}^{MN_{key}} (P_{keyM} m_{key} + P_{keyN} n_{key}) \tag{5}$$

where

$$P_{keyM} = \sum P_{letter} \tag{6}$$

$$P_{keyN} = \sum P_{letter} / P_{keyM} \tag{7}$$

$m_{key} = (0, \cdots, M-1)$ and $n_{key} = (0, \cdots, N-1)$ are the coordinates of a key in a 2D-SAK; P_{keyM} is the sum of the English letter frequency P_{letter} of the letters on all the keys in the same row (Fig 5); and P_{keyN} is the sum of the English letter frequency P_{letter} of letters on a key on the same row. Thus the mean input time per letter t_{input} is

$$t_{input} = t_c^{2D} + 2t_{click} \tag{8}$$

Since the user have to make to clicks in one key or letter selection. Therefore, we can calculate the average input time per letter via Equation 4 and 8 in case of 1D- and 2D-SAKs, which can be used to compare their efficiency given the same scanning interval τ and click time t_{click}.

[1] Letter frequency http://en.wikipedia.org/wiki/Letter_frequency/

P_{key}	
ABCD	0.29366
FGHIJ	0.17456
KLMNO	0.21459
PQRST	0.23394
UVWXYZ	0.08294

Table 1 P_{key}

	P_{keyM}	P_{keyN}
ABC	0.4670	0.2665
DEF		0.4107
GHI		0.3228
JKL	0.2962	0.1671
MNO		0.5625
PQR		0.2704
STU	0.2368	0.7662
VWX		0.1473
YZ		0.0865

Table 2 P_{keyM} and P_{keyN}

	Mean sec/char	Mean sec/word	Mean clicks/word	SD
5.69 CPW	3.07	17.46	5	1.67
4.52 CPW	3.61	16.32	4.48	1.39

Table 3 Character Per Word (CPW)

Take the reference designed 5-key 1D-SAK as an example, and set $\tau = 1$ second. The mean time per letter t_c^{1D} is 1.64 seconds given P_{key} in Table 1. Similarly the mean time per letter t_c^{2D} of our reference designed 2D-SAK is 1.67 seconds given P_{key} in Table 2. This analysis proved that both the reference designs have almost the same mean time per letter, but the difference of click times in real input.

Furthermore, two tests were conducted using test phrases from the "English Conducted Book". Both test phrases include 199 characters, one with 35 and the other 44 words (each punctuation was counted as one word), which means the average Character Per Word (CPW) balance are 5.69 for the first passage and 4.52 for the second one. Table 3 shows the average clicks per word and its standard deviation. As can be seen, the average clicks between a word with more characters and one with less differs by 0.52 clicks. That means the program is better in writing words with more characters than with less characters. The reason is that it takes almost as many clicks for words with more characters as it does for ones with fewer characters. For example, "everyone" and "something" need five clicks while "it", "by" and "in" need four clicks.

4 Implementation of SAKs

4.1 Implementation of 1D-SAK

For practical use, obviously there is much more to consider systematically in the SAK framework. We need to type words, letters and symbols (including punctuation) in real world application. Apart from that, function keys should be considered as well, e.g., delete, go back, look for more words, etc.

Based on the reference design of 1D-SAK, we developed a full text entry system by integrating words, letters, symbols, and functions all together into the high efficiency, easy learning 1D-SAK framework. As seen in the implementation schema (Fig 6), the interface consists of two parts: 1) capital letters in alphabetic orders on the left of arrows, 2) letters, words, and punctuation displayed on the right of the arrows. The left part is stable while the right part changes according to user's indication. When a key of 1D-SAK (Fig 6, left) is clicked, the system extends all the letters on that key into six separate keys (The up row in Fig 6). Caps Lock button is for switching between lower and upper letters, and it all happens on the 1^{st} key pressing. In this state, user can select a letter on the right column or continue to select the letter groups on the left column. The system turns to words input if the letter groups are clicked again (Middle row in Fig 6). Accordingly the system shows words on the right, which is a multi-layer menu for word selection. In the case that the required word is not shown on the 1^{st} level, which is likely to happen especially with shorter words, the user should keep on clicking "More Words" button to go to 2^{nd}, 3^{rd}, 4^{th}, $\cdots$ level for word selection. All the words are listed in the levels ac-

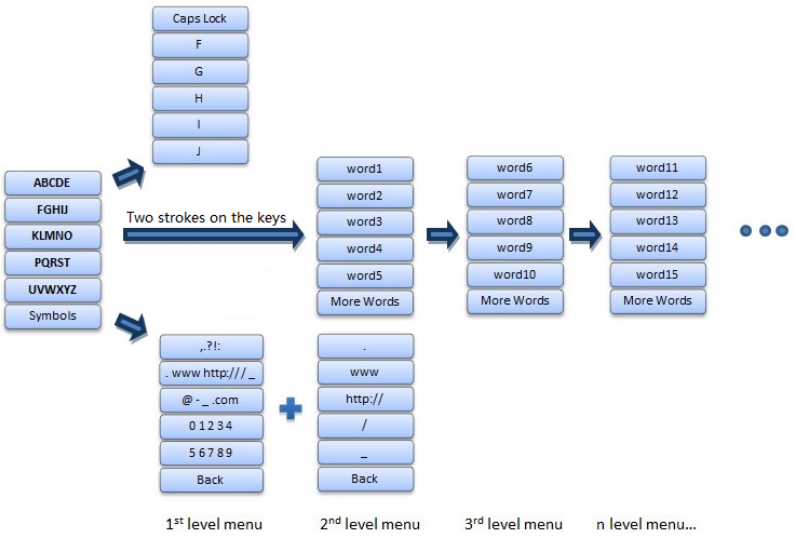

Fig. 6 The implementation schema of 1D-SAK with 6 keys

cording to their given frequency value and a learning algorithm. Besides, the symbol button divided into four categories (punctuation, web address, email address, and digits) provides necessary functionality. When "Symbols" on SAK is clicked, various symbol category keys will be shown on the left. Note that these symbol keys are on the 1^{st} level. If click the key labeled $\{, . ? ! :\}$, for example, it expands the symbols into the 2^{nd} level keys, on which the symbols are shown one by one. In a nut shell, letters input has one level menu, words input can have multi-level menu depending on words frequency, and symbols input has two level menus.

A screen shot of a real 1D-SAK is presented in Fig 7. Highlighting goes through each key alternatively, and the foreground color of a key is changed when it is clicked.

At the start of the program, without indicating specific input, the right column shows the top five frequent words from a 400,000 English corpus. The first time a key of letters from the left column is clicked, letters are expanded on the right column with a "CapsLock" key on the top of the column. Words appear when one of the letter keys on the left column is clicked for the second time. Words can be chosen as highlighting goes to the right column.

There is also an area named "More-Words Window" on the right of 1D-SAK, which shows the words placed in following "More-Words" columns. Below the "More Words Window", the window named What you have typed records the keys users clicked in the input of a word. This clicking-history eases the process in inputting long and complicated words.

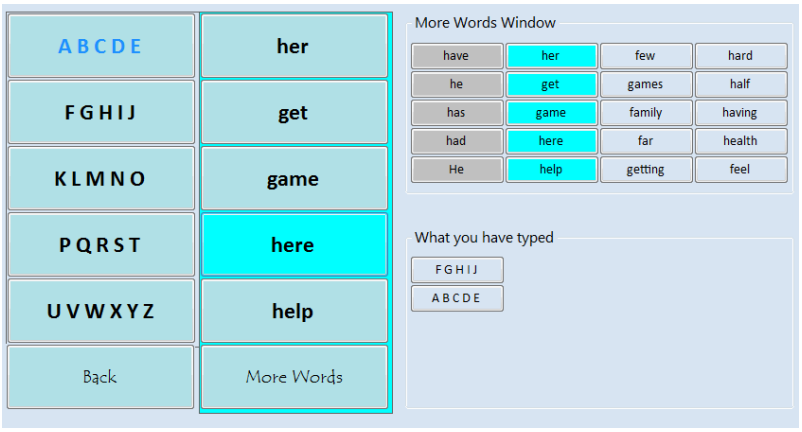

Fig. 7 A screen shot of the real 1D-SAK text entry system (In this case, FGHIJ was clicked and then ABCDE)

4.2 Implementation of 2D-SAK

The doubled 1D-SAK is built in the sense of easy learning for users based on the reference design of 2D-SAK (Figure 8). In this version, users choose one of the three keys at the first step, then the chosen 9 characters is divided into 3 rows as a new column (the second column). Users then select a key again in the three keys with three letters each. After pressing the desired button again, users can see the candidate words on the fourth column to be chosen with ambiguity. At the same time, individual characters on the chosen button is shown in the third column - here users can "type" a single letter or word by repeating the procedures.

Additional functions are also provided just like the 6-key 1D-SAK. For instance, "More Words" provides the next three words from the word list according to their given frequency value; "More Words window" allows users to see in which column the required word is placed. Users can see what they have typed in a click-history window.

4.3 One-Bit input devices

In real world, users have different physical impairments and need special devices for interaction. We have developed prototypes of input devices shown in Figure 9. On the left we show a capacity input device which enables three fingers to place on it (fore-, middle-, and ring-finger). However, only one finger controls the input at

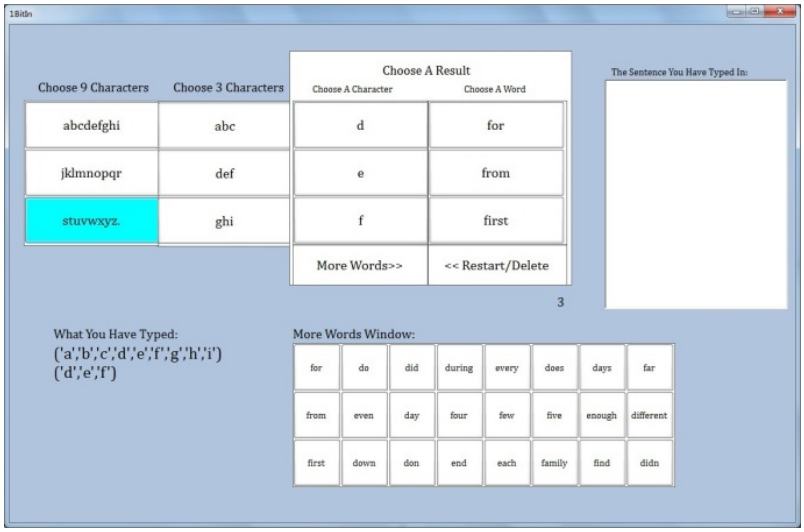

Fig. 8 A screen shot of the double 1D-SAK text entry system (In this case, the keys "abcdefghi" and "def" were clicked)

a time. As long as one finger can be raised and brought down, it can activate and deliver an input. The device's frame was tailored to fit the handicapped people's needs. There is a 1 centimeter gap between the touch buttons in the plastic frame, so that the fingers depress the device lightly. On the right shows a similar device based on detecting the movement of fingers. The hidden infrared sensors between each two stakes capture any movement of a finger for an input. In general, Augmentative and Alternative Communication devices should be ready for a variety of impairment users as described in [13]. The program itself has to provide a user friendly and easy-to-control interface independent from input hardware. For this reason, our program is designed for various input hardware, for instance, a brain computer interaction headset [2].

5 User studies

During the development phase, Value Stream Mapping (VSM) method was used to reduce the necessary steps to type in a word. To guide the users perspective on the program from beginning to end, arrows and boxes are used to symbolize the steps of a task. A simple glass panel and non permanent markers helped to illustrate the work patterns of the program and to get an overview of the users' interaction steps.

Furthermore the design aligns with the current norms and standards for usability in human-computer interaction such as the DIN EN ISO 9241-11 and the dialogue principles DIN EN ISO 9241-110 as well as the ten Nielsen heuristics [16].

The participants of the test consisted of people diagnosed with and without parkinson disease. A quick pretest with 3 participants for the 1D-SAK with non diseased people showed that the interface was more difficult to operate with for people whose first language is not based on the Latin alphabet. To avoid potential language mistakes, English native speaking students were used as test persons. This is to ensure that their knowledge and fluency on diction and the alphabet is not ob-

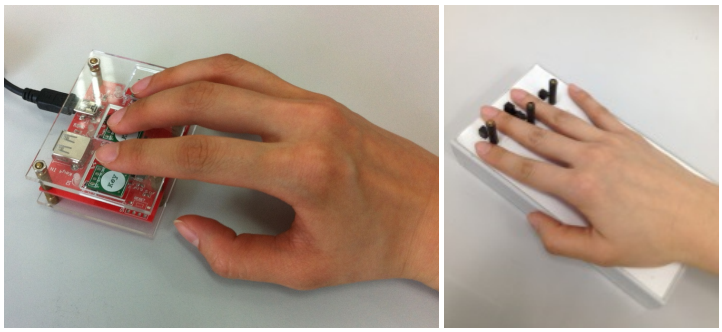

Fig. 9 Prototype of One-Bit input devices

[2] Emotiv Neuroheadset http://www.emotiv.com/store/hardware/epoc-bci/epoc-neuroheadset/

structed. As the program was mainly developed at Tsinghua University in Beijing, China, they were asked randomly in a local caf popular among foreign and exchange students. Both on-screen versions were tested with non parkinson diseased people first to gather shallow usability problems and critical incidents before identifying the ones more specific to the target group. The advantage of testing with non diseased people is that they were easier to recruit in Beijing and the test could last longer since diseased people get tired very easily. 11 participants were recruited for each design of input methods (Fig 7, Fig 8). There were 6 women and 5 men from age 21 to 26.

All the tests were recorded by two video cameras: one was pointed at the display to follow the users navigation, the other one was pointed at the users to record his/her expressions while using the program, which was acknowledged by the participants. After a brief introduction to the program the participants were asked to type in 2 sentences (199 characters) as well as an email address (14 characters). During the experiments, we were using the Thinking Aloud-Method to record participants' thoughts. Before the test, the experimenter explained the procedure and encouraged participants to speak their thoughts out loud while using the program. The required time to fulfill the tasks was not timed because with the Thinking-Aloud method, we wanted to identify critical incidents which could lead to usability problems as well as to gather innovative ideas to improve the program. After the test we discussed their thoughts, feelings and suggestions regarding the program. Each test did not last longer than 45 minutes.

We made usability report for each subject. Based on their feed back, the program in general has high learnability and good satisfaction. Although as healthy people, they sometimes find it slow, but they all believe this could be very beneficial for disabled people and they love this new way of typing. They give some feedback about the program, e.g., words that usually follow "I", say "have" or "am", could show up as predictions. In a nut shell, users are impressed by the idea and find it interesting interacting with only one finger.

The 1D-SAK system was further tested with parkinson patients in the hospital of University of South Florida (USF). In four weeks, five of tens patients of a doctor (neurologist) volunteered to join into the experiment (Fig 10). There were 3 women and 2 men from age 50s to 70s, and two of them in wheelchairs. The patients were trained within three to five minutes via demonstrations, and then asked to text in a sentence they were thinking with one key input device. Four of them have successfully typed in a sentence after few tries, while the other failed after many tries in 20 minutes. In the test, one of them input a name of rare wine which made him proud of it. The patients were especially satisfied with the send mail or SMS option. A patient reported that she needs an hour to input a sentence for SMS message via the keyboard of blackberry phone, but 18 minutes for a similar sentence via 1D-SAK. An impressive feedback from a patient is "Would you please slow down the flashing? I prefer controllable to high efficiency system." This obviously differ from the student subject, who usually complain the system is very slow. The highlighting interval option allows the user to adjust the program to his/her preference.

A PhD student in USF took this 1D-SAK system out of the testing room at night to another hospital in Florida several times. He used it to help his father, who was unable to communicate with his son for the first time after a stroke, to talk with him. This patient cannot speak, and the written words on a paper by his trembling hand was unrecognizable to his relatives. The student said that his father is very satisfied with this new device, and asked him to take away when he left. His father said: ”the other patients will be jealous and steal the device from me”, which made his eyes filled with tears.

These studies proved that the system is helpful for disabled or diseased users, and very easy to learn. The simple structure enables a high learnability curve and a fast interaction with the program. It is not only a tool for allowing handicapped people to text quicker, but also for integrating them in the lives of their friends and family.

6 Conclusion

Everyday life activities such as interacting with a computer can already be a challenge for people suffering from motor impairments and neurological diseases. As already mentioned earlier, to fail using simple applications or programs can lead to sadness or depression. An intuitive and predictable usable interface is highly needed for providing accessibility for all of them [15]. This great potential needs have guided us to the design and implementation of the one-bit text entry system.

The framework of SAK design is focused on integrating efficiency, ambiguity and learnability into a guidance for SAK design and development. The two optimized reference systems are designed via balancing higher efficiency (fewer on-

Fig. 10 User Studies in the hospital of University of South Florida with patients. Left: the testing room in a hospital of USF, Right: a patient in testing with 1D-SAK

screen keys) and lower ambiguity (fewer characters per key). Moreover, an efficiency model is constructed for one-bit text entry systems, which helps the designer to verify the optimal of SAK systems.

Based on the framework and the reference designs, we developed two SAK systems by integrating letters, symbols, auto-space, auto-capitalization, etc. into an intuitive interface. A German version of the 1D-SAK system is built with umlauts like "ä", "ö", "ü", and a corpus of 2 million German words. The SAK systems will also be extended to multilingual versions in near future.

User studies with target groups are conducted, which proved high satisfactions of both patients and their relatives. The system already demonstrated its usefulness and power to elders and disabilities for their better life quality.

Acknowledgement

References

1. Sanderson W. C., Scherbov S.: Remeasuring aging. Science **329**, 1287–1288 (2010)
2. DESA U. N.: World population prospects: the 2008 revision. Department for Economic and Social Affairs, New York (2009)
3. Camicioli R., Fisher N.: Parkinson's disease with dementia and dementia with Lewy bodies. Neurology Asia **10**, 79–98 (2005)
4. Lau, D., Lonneke, M. L., Breteler, M.: Epidemiology of Parkinson's disease. The Lancet Neurology **5**, 525–535 (2006)
5. Pantke Karl-Heinz: Einfuerung in die Unterstuetzte Kommunikation. Unterlagen zur Veranstaltung, Berlin (2008) (Available via WIKIPEDIA.ORG) http://www.locked-in-syndrom.org/unterstuetzte-kommunikation.pdf Cited 15 Nov 2013
6. Ghedira S., Pino P., Bourhis G.: Conception and experimentation of a communication device with adaptive scanning. ACM Transactions on Accessible Computing **1**, 14 (2009)
7. ISO: ISO 9241-110, Dialogue principles. ISO Standard (1996)
8. MacKenzie I. S.: The one-key challenge: searching for a fast one-key text entry method. Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility, pp. 91–98. ACM (2009)
9. MacKenzie I. S., Felzer T.: SAK: Scanning ambiguous keyboard for efficient one-key text entry. ACM Transactions on Computer-Human Interaction **17**, 11 (2010)
10. Miró-Borrás J., Bernabeu-Soler P., Llinares R., Igual J.: Ambiguous Keyboards and Scanning: The Relevance of the Cell Selection Phase. Human-Computer Interaction–INTERACT 2009, pp. 1–4. Springer-Verlag (2009)
11. Miró-Borrás J., Bernabeu-Soler P., Llinares R., Igual J.: A prototype scanning system with an ambiguous keyboard and a predictive disambiguation algorithm. Proceedings of the 12th international conference on Computers helping people with special needs–ICCHP'10, pp. 136–139. Springer-Verlag (2010)
12. Wiki: Letter Frequency (2013) (Available via WIKIPEDIA.ORG) http://en.wikipedia.org/wiki/Letter_frequency Cited 15 Nov 2013
13. Baljko M., Tam A.: Indirect text entry using one or two keys. Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility, pp. 18–25. ACM (2006)
14. Schrag A., Jahanshahi M., Quinn N.: What contributes to quality of life in patients with Parkinson's disease?. Journal of Neurology, Neurosurgery & Psychiatry **69**, 308–312 (2000)

15. Schapira, A. H. V.: Science, medicine, and the future: Parkinsonś disease. British Medical Journal **318**, 311 (1999)
16. Nielsen, J. (1995). Ten Usability Heuristics for User Interface Design. Jakob Nielsens Alertbox. (1995) (Available via Nielsen Norman Group)
http://www.nngroup.com/articles/ten-usability-heuristics Cited 15 Nov 2013