PREDICTIVE SCANNING KEYBOARD OPERATED BY HISSING

Ondřej Poláček, Zdeněk Míkovec, Pavel Slavík
Faculty of Electrical Engineering, Czech Technical University in Prague,
Karlovo nám. 13, 12135 Praha 2, Czech Republic
{polacond, xmikovec, slavik}@fel.cvut.cz

ABSTRACT
This paper presents a new predictive scanning strategy based on the N-ary search algorithm. This strategy is demonstrated on three virtual keyboards operated by hissing – a novel interaction method suitable for severely motor impaired people with speech impairments. An experiment with 39 users was conducted to compare the keyboards to three existing scanning keyboards. The participants were able to enter a text by hissing at the speed of 10-16 characters per minute. Experiment results have also shown that a scanning strategy must be predictable for the users as they need to plan their actions in advance.

KEY WORDS
Text entry method, Scanning, N-ary search, Non-verbal vocal interaction, Hissing, Predictive input.

1. Introduction

Text entry methods have been studied for quite a long time. The QWERTY keyboard has been the most popular layout since the first typewriters emerged. However, in special circumstances (e.g., impaired users, mobile environment) no dominant text entry method can be identified. This has consequently led to the development of many non-traditional approaches, as well as various methods of interaction.

An experienced typist using ten fingers on the QWERTY keyboard can achieve writing speed as high as 250-400 characters per minute (CPM). With such a high speed there is no space for any enhancements like predictive completion or dynamic layouts. For example, using the Reactive keyboard [1] good typists were slowed down by word prediction, however, poor typists or people with motor disability may gain speed.

Physically disabled people usually cannot achieve such a high speed due to their constraints. Their communication with computers is rather limited to only several distinctive stimuli – a small number of physical buttons, a joystick, eye-tracking, the features of the electroencephalographic (EEG) signal etc. This limitation can be compared to typing with one finger only on a virtual keyboard displayed on a touch screen. There is a research available [2], showing that typing with one finger on a touch screen with virtual QWERTY keyboard results in a speed of 160 CPM for expert users after 30-minute training. If the size of the virtual keyboard is reduced from 25 to 7 cm then the speed drops to 105 CPM. The speed reached by physically disabled people will certainly be lower. This situation opens a space for research of new entry methods which will take into account various limitations of motor impaired users and increase the entry speed.

There is currently a range of assistive tools available to help users with motor impairments. However, each user may have significantly different capabilities and preferences according to the range and degree of their impairment. In case of severe physical impairment, people usually have to use other interaction methods to emulate the keyboard. One of the methods successfully used by people with special needs is the non-verbal vocal interaction (NVVI) [3]. It can be described as an interaction modality, in which sounds other than speech are produced.

The research of the non-verbal vocal interaction can be roughly divided into three areas according to timbre of the voice. In pitch-based input the user can control computer applications by height and/or length of tone. Whistling or humming are common sounds used in this interaction style. Pitch-based NVVI has been successfully used in mouse emulation [4], keyboard emulation [5] or real-time games [6]. Next NVVI area uses vowels to control various applications. This has been used for example for mouse emulation [7] or movement of robotic arm [8].

NVVI based on hissing has been so far used mainly for artistic installations such as Expressmas Tree, sssSnake, or Blowtter [9]. Blowtter is a voice controlled plotter and it has been successfully used by motor-impaired children. However, there is no work on a text input method operated just by hissing. Scanning is a suitable technique for those, who can only produce one or two distinctive stimuli. Hissing is a method of interaction that can be used by paralyzed people [9] or people with upper-limb motor impairments, such as quadriplegia induced by stroke, cerebral palsy, brain injury etc. Users are not required to have healthy vocal folds, as the sound of hissing is generated in buccal cavity. However, they must be able to produce controlled hissing. The advantages of such interaction are language independence and fast and accurate recognition as opposed to speech [3]. Speech recognition software usually works relatively well for native speakers; however, the accuracy is much lower in the case of accented speakers or people with speech impairment.

In this paper we use the *character per minute* (CPM) rate [10] for measuring the speed of the keyboards. The CPM rate is defined by Equation 1, where |T| is the length of the written text in characters and S is the time in seconds. According to ISO 9241-4 the word per minute (WPM) rate can be computed by multiplying CPM by five.

$$CPM = \frac{|T| - 1}{S} \times 60. \quad \textbf{(1)}$$

In order to express the number of hissing commands per written character we use in this paper the *gesture per character* (GPC) rate described in [10]. Gesture is regarded as an atomic operation – in our case a gesture corresponds to a hissing command. Text entry methods with low GPC rate are considered better than those with high rate, however, other parameters must be taken into account, such as the length of hissing or the length of scanning period. The GPC rate is defined by Equation 2, where $|IS_{\emptyset}|$ is an input stream which contains all hissing commands produced by the user and |T| is the length of the written text in characters.

$$GPC = \frac{|IS_{\emptyset}|}{|T|}. \quad \textbf{(2)}$$

2. Related Work

There is a wide range of text entry methods targeting motor-impaired users. The methods described in this section often differ significantly in physical interaction used, which is determined by specific motor impairment. Each method is often unique to concrete impairment conditions and thus it typically makes no sense to compare various methods as they are not in concurrent position. Several major principles can be identified in the literature – *scanning*, *predictive completion*, and *ambiguous keyboards*. Those principles are often combined together to gain better performance.

When the number of stimuli, which can be issued by the user, is limited to only one or two, using scanning technique is inevitable. For example, in the case of two buttons, the first button can be pressed repetitively (scanning) to select a key and second button is used to confirm the selection. When only one button is available, keys are selected automatically during a certain amount of time. When the time expires, next key is selected. The button is used to confirm the selection again. Keys can be selected by linear scanning [11], row-column scanning [12], three-dimensional scanning [13], or containment hierarchical scanning [14]. Layout of a scanning keyboard can be static and characters can be displayed alphabetically or sorted according to its probability. Some scanning keyboards use dynamic layouts, in which keys are reorganized after each entry according to the probability of preceding n-grams [15]. An exhaustive enumeration of scanning techniques is described in work by Lesher et al. [16]. Generally, scanning technique is very slow as it reaches only approximately 5-20 CPM.

A text entry method can be accelerated by prediction, when a list of possible completions is updated with each entered character. This reduces number of keystrokes per character. The Reactive Keyboard [1] predicted possible words according to context that had been already written. An adaptive dictionary-based language model was used. Predicted candidates could be selected by the mouse cursor. Expert users of a QWERTY keyboard would be slowed down, however, such prediction is useful for poor typists or people with limited movement of upper limbs. Another predictive keyboard GazeTalk [17] predicted six most probable letters and six words according to current context. If no prediction was correct, there was full keyboard available. This virtual keyboard was controlled by the eye gaze. The keys were activated by dwell-time selection system [18]. The average typing rate achieved by novice users was 16 CPM.

In ambiguous keyboards the alphabet is divided into several groups of characters and then each group is assigned to one key. The user selects desired characters by selecting the keys and after a sequence of keys is entered the word is disambiguated using a dictionary. Probably the most prevalent ambiguous keyboard is the commercial T9 system by Tegic Communications [19] that is widely used in mobile phones. For its efficiency, similar ambiguous keyboards were designed for physically impaired people. Kushler [20] describes an ambiguous keyboard in which the alphabet was assigned to seven keys and the eighth key was used as a space key that initiated the disambiguation process. Tanaka-Ishii [21] published similar system, in which only four physical keys were used. Besides disambiguation, the text entry method was capable of predicting words. The average speed of this method was 70 CPM, achieved after ten sessions by able-bodied participants. Harbusch [22] presented similar method in which the whole alphabet was assigned to only three keys and one key was used for executing special command in a menu.

Combining linear scanning with an ambiguous keyboard is a common technique. For example, Kühn [23] used four-key scanning ambiguous keyboard and achieved 35 CPM without out-of-vocabulary words. Miro [24] limited the number of keys to only two (keys 'a-m' and 'n-z') and estimated its entry rate to 50 CPM for an expert user. Beltar [25] used three keys and developed a virtual mobile keyboard. In QANTI [26] three keys are mapped to the alphabet. The keyboard is operated by one switch that is triggered by intentional muscle contractions. The typing rate ranges from 12.5 to 33 CPM.

An efficient system is Dasher [27], which is based on a dynamically modified display and adaptive language model [28]. The characters are selected by moving the mouse cursor around the screen. Continuous "one finger" gestures are used as the input method. This is a very suitable input method for motor impaired users who can operate a pointing device. The writing speed achieved is approximately 100 CPM with experienced users reaching up to 170 CPM. For users who have no hand function, a modification of the Dasher system can be made to allow input via eye tracking. A longitudinal study [29] found that an average writing speed of 87 CPM after ten 15-minute sessions could be achieved. This speed was a large increase from the initial speed of just 12.5 CPM. Speech Dasher [30] is another interesting modification of Dasher. It combines speech input with the zooming input of Dasher. The system must first recognize a user's utterance. Errors are then corrected via the zooming input. Expert users reached a writing speed of approximately 200 CPM.

The non-verbal vocal interaction (NVVI) has also been used for keyboard emulation. Sporka et al. [5] describe a humming-based method of keyboard emulation. Each vocal command is assigned to a specific key on the keyboard, when a sound is produced a corresponding key is emulated. The average reported typing rates varied between 12 and 16 CPM, which was measured in a study with able-bodied participants. Different assignments of NVVI sounds to keys were investigated, namely the pitch-to-address, pattern-to-key and Morse code mappings. In the pitch-to-address mapping, the keyboard was mapped onto a 4×4×4 matrix, while a sequence of three tones of specific pitches determined an address in the matrix. In the pattern-to-key mapping each key was assigned to a specific sound.

Another keyboard operated by humming is CHANTI [31]. It is an ambiguous keyboard, where the alphabet is split into only three groups. The keyboard combines philosophy of the ambiguous keyboard QANTI [26] and humming input. Scanning technique is replaced by direct selection of a key by humming. The keyboard was tested with five severely motor-impaired people, the speed ranged from 10 to 15 CPM after 7 sessions.

Humming input and the PPM language model [28] is combined in Humsher [32]. Four keyboard layouts are presented and evaluated with motor impaired users. They were able to type at 14 – 22 characters per minute after seven sessions.

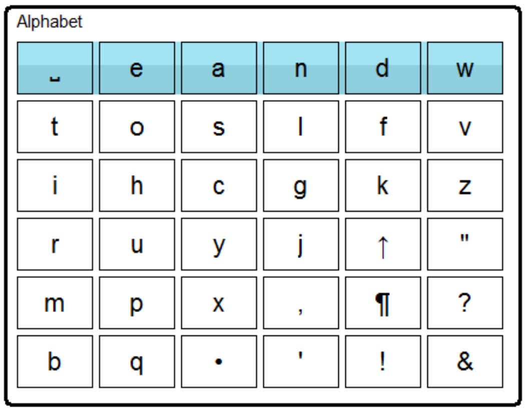
Additionally, the P300 speller [33] is a method that utilizes the electroencephalographic (EEG) signal in the human brain to control a virtual keyboard. The keyboard is a 6x6 matrix containing alphanumeric characters. The user focuses on a character and as the character flashes, the brain produces a stimulus. At least two flashes are needed to input a character. According to Wang et al. [33], the writing speed achieved is approximately 7.5 CPM.

3. Scanning Keyboard Layouts

As described in the previous section, scanning keyboards can use either static or dynamic layouts. Generally, dynamic layouts are rarely appreciated by the users as the position of characters changes and the users often have to search for a character. The solution to this can be a static layout with dynamic selection. For example, in Humsher [32] the static layout of characters was considered better than dynamic layouts by the users. However, in work by Baljko et al [14] the users strongly preferred dynamic layout. One of the objectives of our paper is to find out what causes this contradiction. Another objective of the paper is to evaluate hissing as an input modality for a text entry method.

We developed six scanning keyboards with different layouts. Three of them followed the row-column scanning paradigm and the others used the N-ary search algorithm for selecting a character. Except one keyboard all of them utilized character-level prediction based on PPM (prediction by partial match) language model [28]. The model provides probabilities of characters according to already written context.

All scanning keyboards presented in this paper use two hissing sounds – the short one to confirm selection and the long one for corrections. The threshold between these two tones is initially set to 500 ms, however, it can be adjusted for each user. Hissing sound is recognized when root mean square (RMS) of the filtered input audio signal is higher than a T threshold T_{RMS}. This threshold is also adjusted for each user.



␣	e	a	n	d	w
t	o	s	l	f	v
i	h	c	g	k	z
r	u	y	j	↑	"
m	p	x	,	¶	?
b	q	•	'	!	&

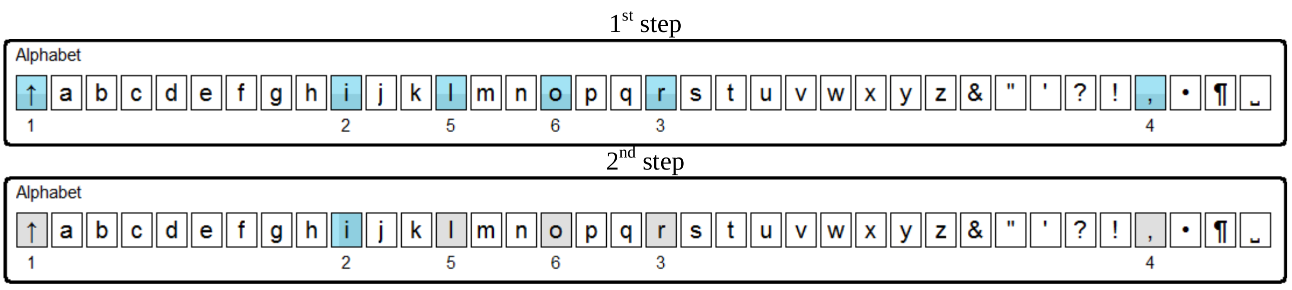
Figure 1. Static RC Scanning Keyboard

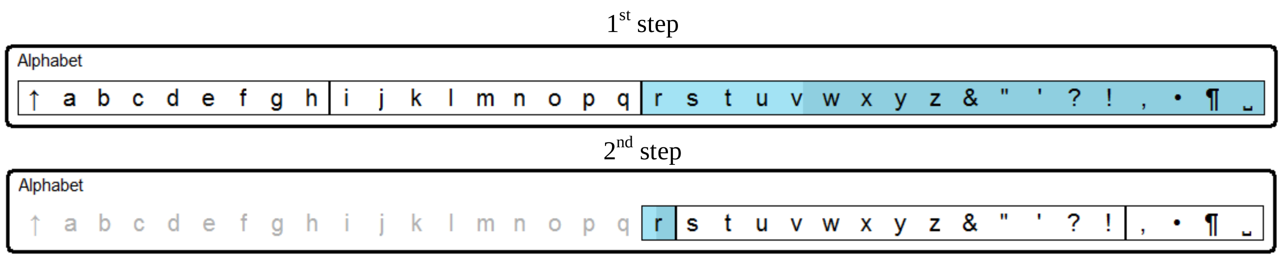
3.1. Row-column scanning

In row-column scanning the user has to select a row in the first step and then a column. The selection appears only for a limited time. The user has to produce a signal within this timeout to confirm the selection. Three layouts that use row-column scanning were developed and tested.

3.1.1. SK1: Static RC scanning

The SK1 keyboard is depicted in Figure 1. The position of characters is static and sorted according to their probability in English. For example, the space as the most probable character is located in the upper left-hand corner and can be selected by only two short gestures with no scans.


1st step
Alphabet
↑ a b c d e f g h i j k l m n o p q r s t u v w x y z & " ' ? ! , • ¶ ␣
1 2 5 6 3 4
2nd step
Alphabet
↑ a b c d e f g h i j k l m n o p q r s t u v w x y z & " ' ? ! , • ¶ ␣
1 2 5 6 3 4
Figure 2. Array arrangement of row-column scanning keyboard.


1st step
Alphabet
↑ a b c d e f g h i j k l m n o p q r s t u v w x y z & " ' ? ! , • ¶ ␣
2nd step
Alphabet
↑ a b c d e f g h i j k l m n o p q r s t u v w x y z & " ' ? ! , • ¶ ␣
Figure 3. Ternary search scanning keyboard, typing "*r*" after "*Text ent*"

3.1.2. SK2: Dynamic RC scanning in matrix

The SK2 keyboard uses the same layout as the SK1 keyboard (depicted in Figure 1). However, the position of characters changes as the user enters text. The characters displayed in the grid are sorted by probabilities that are extracted from the PPM language model. The most probable character appears in the top left-hand corner, while the least probable character in the bottom right-hand corner. Each time the user enters a character, a new arrangement appears in the grid and the user has to locate the desired character over and over.

3.1.3. SK3: Dynamic RC scanning in array

The SK3 keyboard (see Figure 2) works in the same way as SK2 keyboard, however, visual presentation is different. The characters are always sorted alphabetically and displayed as an array. Entering a character is also achieved by two steps. Initially groups of six characters are scanned as depicted in Figure 2. This corresponds to the row scanning in SK2. The numbers below the characters mean the order of selection in the next step. After the selection of a group is made corresponding six characters of this group are highlighted and scanning proceeds among them, which corresponds to column scanning in SK2. Layout of characters in this keyboard is static, characters remain in their positions while entering text.
An example is of user interaction with this keyboard is depicted in Figure 2. Let us assume that the user has already entered the text "Text ent" and wants to continue by entering character "r". In the first step the user has to wait until the group that includes "r" is highlighted and then select it. In our case characters belonging to the group are those with blue background (Figure 2-1st step). In the second step scanning is done only among characters of the group (grey characters in Figure 2-2nd step). Again, the user has to wait until "r" turns blue and then submit the selection.

3.2. N-ary Search Scanning

N-ary search scanning keyboards have the same advantage as the SK3 keyboard – characters are always displayed in an alphabetic order. Such order simplifies the process of visual location of a desired character. In the dynamic layout (SK2) users have to locate a character visually by linear scanning and they cannot rely on the spatial memory. The process of locating a correct character can be tedious for low-probable characters. In the static layout of alphabetically sorted characters the time needed to locate a character is then modeled by Hick-Hyman law [34] and is logarithmically dependent on the length of the alphabet. Locating characters visually in the static layout is faster than in the dynamic layout as logarithmic scanning is used instead of linear one.
With the alphabetical order of characters, the desired character can be entered by binary search algorithm adopted from basic programming techniques as used in Humsher [32]. The algorithm can be however generalized to an N-ary search, where the alphabet is split into N groups. Scanning proceeds among these groups until the selection is confirmed. Then the group is split again and again until the correct character is found. Each character is located in the following number of steps:

$$steps = \lceil \log_N L \rceil \qquad \textbf{(3)}$$

L is the size of the alphabet and N is the number of groups. Originally, the alphabet is split into groups of the

Table 1. Keyboards used in simulation and experiment.

Abbrv.	Name	Character Layout	Prediction	Scanning
SK1	Static RC keyboard	Static	No prediction	Row-column
SK2	Matrix keyboard	Dynamic	PPM	Row-column
SK3	Array keyboard	Static	PPM	Row-column
SK4	Binary keyboard	Static	PPM	Binary search
SK5	Ternary keyboard	Static	PPM	Ternary search
SK6	Quaternary keyboard	Static	PPM	Quaternary search

same size. The number of steps can be, however, improved by incorporating character probabilities. When the size of groups is balanced according to the probability of characters, then a character with high probability could be located in fewer steps.

The N-ary keyboards are based on the modified N-ary search algorithm. In each step the alphabet is split into N groups with balanced probability, i.e. the sum of probabilities of characters in each group is as close to 1/N as possible. The boundary between groups is computed by N-ary Huffman algorithm [35].

Three variations of N-ary search scanning keyboards were developed – Binary (SK4), Ternary (SK5) and Quaternary (SK6) keyboards (see Table 1). An example of user interaction with the Ternary keyboard is depicted in Figure 3. Let us again assume that "Text ent" is the already entered text and "r" is the character to enter. In the first step the alphabet is split into three groups "shift - h", "i-q", and "r-space". The user chooses the third group. In the second step "r" is the only character in the first group because of its high probability. Remaining characters are in the second and third group. The user chooses the first group and character "r" is typed. In this case the character was selected in two steps only.

4. Simulation

Theoretical performance of different keyboards was analyzed in an experiment, where the user actions were simulated by computer. As an input for each keyboard we used Mackenzie's phrase set for evaluating text entry techniques [36]. Following variables were measured:

- C - *Number of characters in the input text*. This number is constant for all keyboards.
- S_i - *Number of uninterrupted initial scan steps*. Initial scan step delay is two times longer than a running scan step delay, therefore this variable was counted separately.
- S_s - *Number of uninterrupted scan steps*. All uninterrupted scan steps except initial ones.
- G - *Number of interrupted scan steps*. All scan steps in which selection is made and hence the timeout is not reached. This number corresponds to number of gestures that has to be produced.

Timeout for initial scan steps was set to 1500 ms and timeout for running scan steps was set to 750 ms. Time needed to produce a gesture was set to 500 ms, which was empirically measured before. Based on these values and the variables listed above we computed an estimation of CPM_{est} and GPC_{est} rates for each keyboard according to Equations 4 and 5.

(4)

$$GPC_{est} = \frac{G}{C}. \quad \textbf{(5)}$$

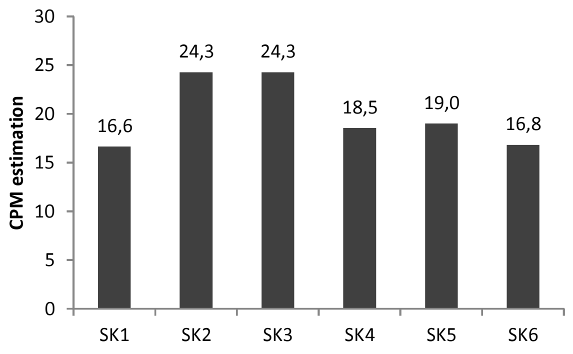


Figure 4. Estimation of CPM for each keyboard.

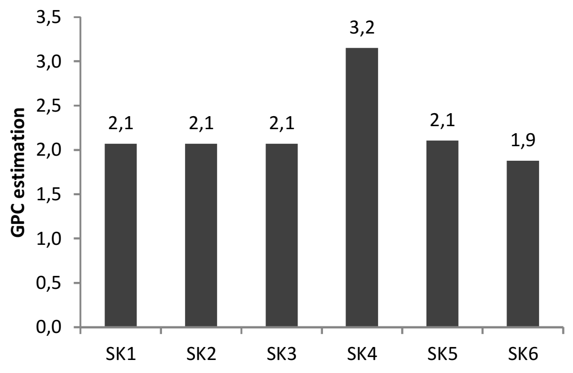


Figure 5. Estimation of GPC for each keyboard.

The estimation of CPM and GPC rates is depicted in Figures 4 and 5 respectively. The rates of dynamic scanning in Matrix (SK2) and Array (SK3) are equal as the scanning algorithm is the same for both keyboards. Among N-ary search keyboards, the Ternary keyboard (SK5) has the highest estimated CPM rate. Estimated CPM of higher-order N-ary keyboards is slow, because a lot of scan steps are needed to select a correct group or character. On the other hand, the Binary keyboard (SK4) is slower than the Ternary keyboard (SK5) because a lot of confirmations have to be made to select a desired character (see its high number of GPC in Figure5). We also run simulation on 5-ary, 6-ary, and 7-ary keyboards, however, these were discarded from further evaluation due to their low estimated performance.

5. Experiment

Despite the results of simulation we expected the N-ary and Array keyboards to perform better than the Matrix keyboard because of the static layout of characters. In order to validate this expectation we conducted an experiment with users. There were established two groups of participants. One group evaluated three keyboards based on row-column scanning and the other group used three N-ary search scanning keyboards. There were total 39 participants (35 men, 4 women, mean age=21.3, SD=1.04) recruited from university students. The keyboards were compared in terms of efficiency and user satisfaction. The first group tested row-column scanning keyboards (SK1, SK2, and SK3) and the other one tested N-ary search scanning keyboards (SK4, SK5, SK6). Each participant completed two 30-minute sessions. During the sessions they were asked to copy phrases taken from the Mackenzie’s phrase set [36]. In order to minimize the learning effect, the sequence of keyboards in each group was counterbalanced. After finishing both sessions the participants were asked to complete a post-test questionnaire to acquire subjective rating of each keyboard. The whole experiment was held remotely. The initial scan delay was set to 1500 ms and a scan delay to 750 ms for each user.

The speed of each keyboard in terms of CPM rate is depicted in Figure 6. The Matrix keyboard (SK2) was significantly the fastest (F(5,102) = 13.0, p < .01) keyboard. The Ternary keyboard (SK5) was significantly faster than the Quaternary one (SK6). Other differences were not statistically significant. Even though the theoretical speed of the Array keyboard (SK3) was the same as of the Matrix keyboard (SK2), the measured speed was much lower. It was quite difficult to keep up with scanning for the participant and they complained that scanning is not predictable. Otherwise the results correspond to simulations.

Average GPC rates are depicted in Figure 7. They roughly correspond to the results of simulation, however, they are affected by misspellings and consequent corrections made in the real experiment by participants. The average number of corrections is depicted in Figure 8. Participants made most corrections when using the Binary keyboard (SK4).

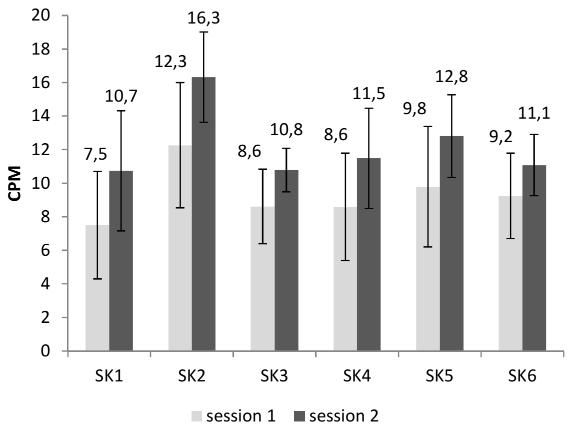


Figure 6. Average CPM rates for each keyboard. Error bars correspond to standard deviations.

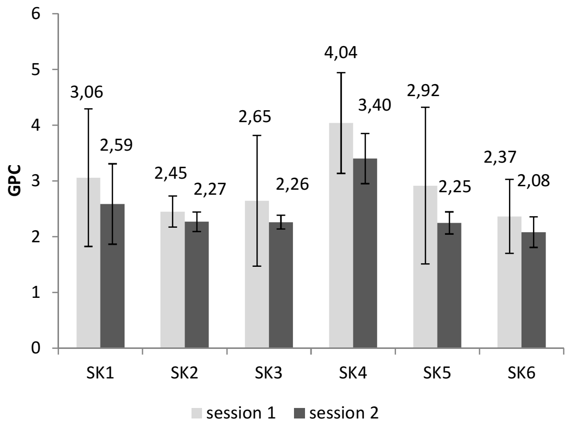


Figure 7. Average GPC rates for each keyboard. Error bars correspond to standard deviations.

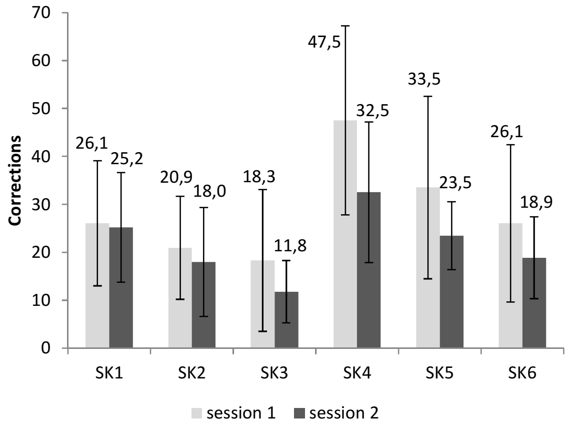


Figure 8. Average number of corrections for each keyboard. Error bars correspond to standard deviations.

In the post-test questionnaire the participants were asked to evaluate each keyboard in terms of perceived accuracy, speed and comfort. Likert scale was used for each statement (e.g. *The Static keyboard is accurate.*). The results are shown in Figure 9. The best rating received the Matrix keyboard (SK2), while the worst one was the Array keyboard (SK3).

The participants mostly complained about the Array keyboard (SK3). They mostly made comments about behavior of the keyboard such as “It was like a lottery”, “It was uncomfortable and hard to follow”. There were also several negative comments about the Binary keyboard (SK2): “The scanning was too fast”. One participant said that the Ternary keyboard was a “reasonable compromise”. Most positive comments were made about the Matrix keyboard (SK2). One participant complained about fatigue and dry mouth after the experiment.

Our expectation that the N-ary and Array keyboards will perform better than the Matrix keyboard because of the static layout of characters turned out to be invalid. The objective results as well as participant comments lead to a conclusion that the correct choice of scanning strategy is crucial for the scanning keyboard and surprisingly the layout of characters does not play such a significant role as expected. The scanning strategy should be easily predictable by the user to allow planning of selections in advance.

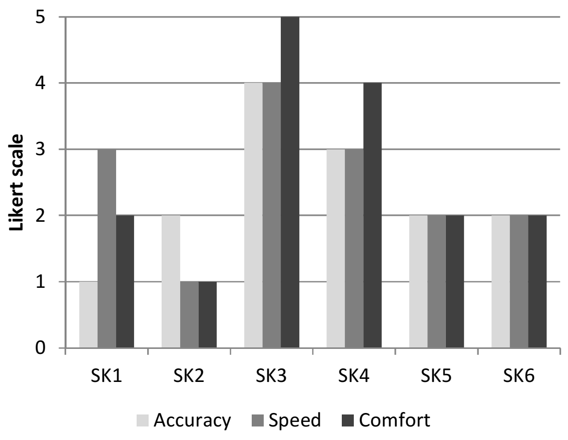


Figure 9. Questionnaire results. Scale 1 (= certainly yes) … 3 (= neither yes nor no) … 5 (= certainly no). Median values displayed for each keyboard.

6. Conclusion

This paper has presented and evaluated six scanning keyboards operated by hissing. The paper has also introduced a new scanning strategy based on the N-ary search algorithm. The ternary search scanning keyboard was the best among N-ary keyabords in terms of speed and user satisfaction. However, it did not outperform the keyboard with dynamic row-column scanning in matrix. The participants were able to enter a text by hissing at the speed of 10-16 CPM depending on the keyboard used.

Dynamic layouts of characters are not commonly used in text entry methods, as the user has to search for next character after each entry. However, the experiment shows that the layout of characters is not as much important as the scanning strategy in terms of user satisfaction. Scanning strategies with an easily predictable sequence by the user are significantly better accepted than strategies that are not easily predictable. The users should not be forced to act immediately after single scan step as planning their actions in advance is important for them. We believe that this is the reason why array arrangement of row-column scanning as well as containment hierarchical scanning proposed by Baljko [14] failed. The N-ary search scanning presented by this paper is easily predictable and therefore accepted by the users, while retaining the static layout of characters.

Acknowledgements

This research has been supported by the project Automatically generated user interfaces in nomadic applications founded by grant no. SGS10/290/ OHK3/3T/13 (FIS 10-802900) and the Veritas project (IST-247765).

References

[1] Darragh, J. J., Witten, I. H., James, M. L. 1990. The Reactive Keyboard: A Predictive Typing Aid. Computer Journal, vol. 23, IEEE Press, 41-49.

[2] Sears, A., Revis D., Swatski, J., Crittenden, R., Shneiderman, B. 1993. Investigating touchscreen typing: the effect of keyboard size on typing speed. In J. Behaviour and Information Technology, vol. 12, 17-22.

[3] Igarashi, T., Hughes, J F. 2001. Voice as sound: using non-verbal voice input for interactive control. In Proceedings of UIST ’01, ACM Press, 155-156.

[4] Sporka, A.J., Kurniawan, S. H., Slavik, P. 2004. Whistling user interface (u3i). In 8th ERCIM International Workshop “User Interfaces For All”, LCNS 3196, Springer, 472-478.

[5] Sporka, A. J., Kurniawan, S. H., Slavík, P. 2006. Non-speech operated emulation of keyboard. In: Clarkson, J., Langdon, P., and Robinson, P. (eds.) Designing Accessible Technology, Springer, London, 145-154.

[6] Sporka, A. J., Kurniawan, S. H., Mahmud, M., Slavik, P. 2006. Non-speech Input vs Speech Recognition: Real-time Control of Computer Games. In Proc. of ASSETS ‘06, ACM Press, 213-220.

[7] Harada, S., Landay, J.A., Malkin, J., Li, X. and Bilmes, J.A. 2006. The Vocal Joystick: Evaluation of voice-based cursor control techniques. Proceedings of ASSETS '06,ACM Press, 197-204.

[8] House, B., Malkin, J., Bilmes, J. 2009. The VoiceBot: a voice controlled robot arm. In Proc. of CHI '09, ACM Press, 183-192.
[9] Al Hashimi, S. 2009. Vocal Telekinesis: towards the development of voice-physical installations. In Universal Access in the Information Society, vol. 8, Springer, 65-75.
[10] Wobbrock, J.O. 2007. Measures of text entry performance. In Text Entry Systems: Mobility, Accessibility, Universality, I. S. MacKenzie and K.Tanaka-Ishii (eds.). Morgan Kaufmann, 47-74.
[11] Schadle, I. 2004. Sibyl: AAC System Using NLP Techniques. In Computers Helping People with Special Needs, LNCS 3118, Springer, Heidelberg, 1009-1015.
[12] Simpson, R., Koester, H. 1999. Adaptive one-switch row-column scanning. In IEEE Transactions on Rehabilitation Engineering, vol. 7, no. 4, IEEE Press, 464-473.
[13] Felzer, T.; Rinderknecht, S. 2009. 3dScan: An Environment Control System Supporting Persons With Severe Motor Impairments. In Proc. of ASSETS '09, ACM Press, 25-28.
[14] Baljko, M., and Tam, A. 2006. Indirect Text Entry Using One or Two Keys. In Proc. of ASSETS '06, ACM Press, 18-25.
[15] Baletsa, G., Foulds, R., & Crochetiere, W. 1976. Design parameters of an intelligent communication device. In Proceedings of the 29th Annual Conference on Engineering in Medicine and Biology, Chevy Chase, MD: Alliance for Engineering in Medicine and Biology.
[16] Lesher, G, Moulton, B., and Higginbotham, D. J. 1998. Techniques for augmenting scanning communication. In Augmentative & Alternative Communication, vol. 14, Informa Healthcare, 81-101.
[17] Hansen, J., Johansen, A., Hansen, D., Itoh, K., Mashino, S. 2003. Language technology in a predictive, restricted on-screen keyboard with ambiguous layout for severely disabled people. In Proceedings of EACL Workshop on Language Modeling for Text Entry Methods.
[18] Jacob, R. J. K. 1990. What you look at is what you get: eye movement-based interaction techniques. In Proceedings of CHI'90, ACM Press, 11-18.
[19] Grover, D. L., King, M. T., Kushler, C. A. 1998. Reduced keyboard disambiguating computer, Technical report, US Patent Publication.
[20] Kushler, C. 1998. AAC: Using a Reduced Keyboard.
[21] Tanaka-Ishii, K., Inutsuka, Y., Takeichi, M. 2002. Entering text with a four-button device. In Proc. of 19th International Conference on Computational Linguistics, Association for Computational Linguistics, 1-7.
[22] Harbusch, K., Kühn, M. 2003. Towards an adaptive communication aid with text input from ambiguous keyboards. In: Proceedings of EACL'03, Association for Computational Linguistics, 207-210.
[23] Kühn, M., Garbe, J. 2001. Predictive and highly ambiguous typing for a severely speech and motion impaired user. In Proceedings of 1st International Universal Access in Human-Computer Interaction Conference, UAHCI '01.
[24] Miro, J., Bernabeu, P. 2008. Text entry system based on a minimal scan matrix for severely physically handicapped people. In Computers Helping People with Special Needs, LNCS 5105, Springer, Heidelberg, 1216-1219.
[25] Belatar, M., Poirier, F. 2008. Text entry for mobile devices and users with severe motor impairments: handiglyph, a primitive shapes based onscreen keyboard. In Proceedings of ASSETS '08, ACM Press, 209-216.
[26] Felzer, T., MacKenzie, I., Beckerle, P., Rinderknecht, S. 2010. Qanti: A software tool for quick ambiguous non-standard text input. In Computers Helping People with Special Needs, LNCS 6180, Springer, 128-135.
[27] Ward, D. J., Blackwell, A. F., MacKay, D. J. C. 2000. Dasher – a data entry interface using continuous gestures and language models. In Proc. of UIST '00, ACM, 129-137.
[28] Teahan, W. 1995. Probability estimation for PPM. In: Proceedings of the New Zealand Computer Science Research Students' Conference.
[29] Tuisku, O., Majaranta, P., Isokoski, P., Räihä, K. J. 2008. Now Dasher! dash away!: Longitudinal study of fast text entry by eye gaze. In Proceedings of the 2008 symposium on Eye tracking research and applications, ETRA '08, ACM Press, 19-26.
[30] Vertanen, K., MacKay, D. J. 2010. Speech dasher: fast writing using speech and gaze. In Proceedings of CHI '10, ACM Press, 595-598.
[31] Sporka, A. J., Felzer, T., Kurniawan, S.H., Polacek, O., Haiduk, P., MacKenzie, I.S. 2011. CHANTI: Predictive Text Entry Using Non-verbal Vocal Input. In proceedings of CHI'11, ACM Press., 2463-2472.
[32] Polacek, O., Mikovec, Z., Sporka, A.J., Slavik, P. 2011. Humsher: A Predictive Keyboard Operated by Humming. To apperar in Proceedings of ASSETS '11, ACM Press.
[33] Wang, C., Guan, C., Zhang, H. 2005. P300 brain-computer interface design for communication and control applications. In: Proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBS'05, 5400-5403.
[34] Hyman, R. 1953. Stimulus Information as a Determinant of Reaction Time. Journal of Experimental Psychology, vol. 45, 188-196.
[35] Huffman, D.A. 1952. A Method for the Construction of Minimum-Redundancy Codes, Proceedings of the I.R.E., vol. 40, 1098–1102.
[36] MacKenzie, I. S., Soukoreff, R. W. 2003. Phrase sets for evaluating text entry techniques. In Extended Abstracts of CHI '03, ACM Press, 754-755.