A Text Input Method for Half-Sized Keyboard using Keying Interval

Takuya Katayama
Kobe University
1-1 Rokkodai-cho Nada-ku
Hyogo-ken, Japan
takuya@stu.kobe-u.ac.jp

Kazuya Murao
Kobe University
1-1 Rokkodai-cho Nada-ku
Hyogo-ken, Japan
murao@eedept.kobe-u.ac.jp

Tsutomu Terada
Kobe University
1-1 Rokkodai-cho Nada-ku
Hyogo-ken, Japan
PREST, JST
tsutomu@eedept.kobe-u.ac.jp

Masahiko Tsukamoto
Kobe University
1-1 Rokkodai-cho Nada-ku
Hyogo-ken, Japan
tuka@kobe-u.ac.jp

ABSTRACT

In various environments, such as mobile and wearable computing, compact I/O devices are desirable from the viewpoint of portability. Now, many users are accustomed to input with a keyboard, however, there is a limitation of miniaturization because it degrades the performance of key touch. Therefore, in this paper, we propose a method to miniaturize a keyboard by excluding the half of it. In using the proposed method, one hand hits keys as usual, and the other hand hits the place outside the keyboard as if the user types with both hands. The user can input words with only one hand because the proposed system estimates the input word using keying interval, which appears also when the user inputs with both hands. From the results of user study, we confirmed that the user can input with only one hand and that it does not decrease input speed drastically.

Categories and Subject Descriptors

H.5.2 [Information Interfaces and Presentation]: User Interfaces—prototyping, input devices and strategies; I.5.5 [Pattern Recognition]: Implementation—interactive systems

General Terms

Design, Experimentation

Keywords

Keyboard, compact I/O device, word estimation

MUM '12, Dec 04-06 2012, Ulm, Germany

Figure 1: The comparison of size

1. INTRODUCTION

With the development of microelectronic technology, computers have been downsized, and then, wearable computing attracts more attention. Wearable computing is a technology that deals with computer systems integrated in clothing. In wearable computing environment, compact I/O devices are desirable from the viewpoint of portability. Today, a keyboard is the most popular text input device in desktop computing environment. However, few applications in wearable computing environments employ a full keyboard because it is too big to carry and to be integrated in clothing. Though there are some small keyboards, they have a limitation of the miniaturization. Too much miniaturization degrades the performance of key touch.

Though numerous studies on compact input devices have been carried out, newly designed devices are hard to be mastered. In contrast, keyboards are widely prevalent, and many users are accustomed to the input method of keyboard. Moreover, almost all novel compact devices cannot achieve the input speed of keyboard even if they were mastered.

As shown in Figure 1, a full keyboard is too big to wear. Therefore, in this paper, we propose a method to miniaturize a keyboard by excluding the half of it. The proposed method leverages one's ability of keyboard input, which is

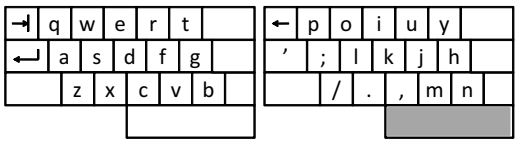

Figure 2: The key arrangement of Wearable Half Keyboard[5]

previously mastered. In using the proposed method, one hand hits keys as usual, and the other hand hits the place outside the keyboard, such as one's thigh or a table, as if the user types with both hands. There are keying intervals caused by the simulant action of normal typing. The proposed system estimates the number of keyings that has been done outside the keyboard using the keying interval obtained from the remaining keys. Moreover, using the estimated number of keyings, the proposed system estimates the input word when the user finish inputting each word. After the estimation, the proposed system displays the candidates for the input word in order of priority. The proposed system makes it possible to reduce the size of keyboard to half without decreasing the performance of key touch and altering the input action. Moreover, the proposed system can make one hand free. The user who get used to the proposed method would become able to input without simulant action of hitting keys. This feature is valuable not only in wearable computing environment. The proposed method is applicable for the environment where only one hand can be used such as input to a car navigation.

The remainder of this paper is structured as follows: The related works are described first. Next, we present the proposed system design including a detailed description of the algorithms. Evaluation results are then presented. We evaluate the input speed and the proficiency of proposed system using a prototype. We provide some idea for improvement in Discussion section. The paper closes with a conclusion.

2. RELATED WORK

A lot of compact text input devices have been proposed and evaluated[1, 2]. We introduce the examples in the following. There are some input methods using body-worn sensors. The gesture based method developed by Liu et al.[3] allows users to input text by tracing letters in mid-air with their fingers that the sensors are attached to. The Chording Glove designed by Rosenberg and Slater[4] is equipped with five sensors located at the tips of each finger that detected when a finger is pressed. Both approaches meet the requirement for downsizing. However, they often lack social acceptability and impose not negligible amounts of training on the user.

For downsizing of key-lined text input devices, it is general that the number of keys is reduced. The most common method is "multi-tap" used on a numerical keypad. Using multi-tap method, a key is pressed multiple times to access the letters assigned to that key. For instance, pressing the "2" key once displays an "a", twice displays a "b", and three times displays a "c". To enter two successive letters that are assigned to the same key, the user must either pause or hit a next button. There are also predictive input methods for numerical keypad such as T9. The user presses

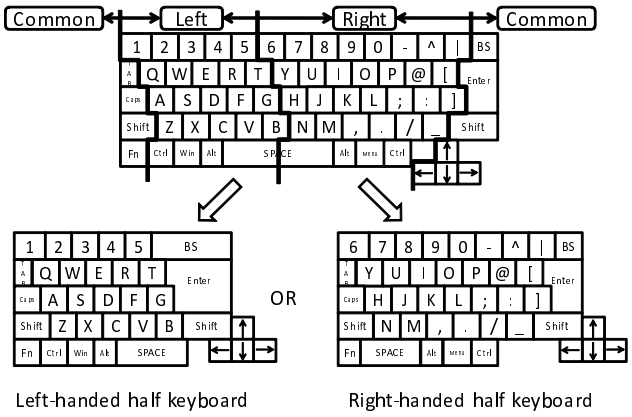

Figure 3: The division of keyboard

each key only once. Once the space key is pressed, the system tries to match all possible interpretations of the entered key sequence to the words that are contained in the dictionary. For instance, "4663" matches "home", "good", and so on. The Matias Wearable Half Keyboard[5] and Twiddler[6] by Handykey are devices that are on shelves for the wearable computing. The key arrangement of Wearable Half Keyboard is left-half qwerty as shown in Figure 2. A key input with holding space key is converted to the corresponding key input in right-half keyboard. The key arrangement in holding space key is the mirror-reversed arrangement of right-half keyboard[7] as shown in Figure 2. Twiddler is a hand-worn device, and it has a joystick and four modifier keys on its front surface and twelve keys on its back surface. Instead of pressing keys in sequence to produce a letter, multiple keys are pressed simultaneously to generate a chord. All of them are the one-handed methods, and therefore the key arrangements are different from that of keyboard. As above, the particular actions are required to make do with low keys. Moreover, the input speed of these method is much lower than that of keyboard because a keyboard is input with both hands.

In contrast, the proposed method exploits the existing ability of inputting a keyboard because it employs the traditional key arrangement and the accustomed input action as much as possible. The propose method uses the keying interval to estimate an input word. Keying interval has much information, for example, the method of Joyce and Gupta[8] uses the keying interval for authentication.

3. PROPOSED SYSTEM

3.1 Design

The proposed method uses one half of the keyboard divided into right and left halves as shown in Figure 3. The keys that are frequently used, such as Space key and Enter key, are arranged on both halves. The user hits keys with one hand and hits outside of keyboard with the other hand. Hereinafter, a hand hitting keys are called "keying hand", and the other hand hitting outside of the keyboard are called "non-keying hand", respectively. The input action of proposed method is similar to that of traditional keyboard. The user inputs as if he/she types with both

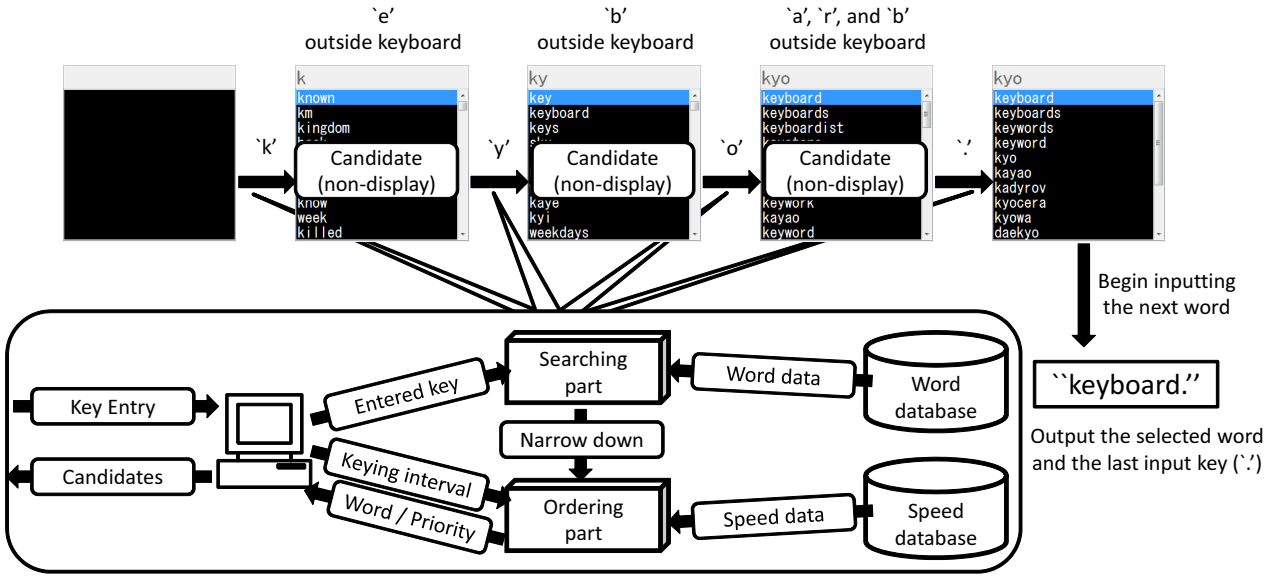

Figure 4: The system flowchart

hands, even though his/her non-keying hand hits outside of keyboard. The simulant hitting action makes the keying interval, which also arises when the user uses a full keyboard, between the keying with keying hand. Using the keying interval, the proposed system estimates the number of keyings that were supposed to be input with non-keying hand. In this way the proposed system makes it possible to reduce the size of keyboard to half, remaining the input action.

Figure 4 shows the system flowchart and the screenshots when the user inputs "keyboard" with one's right hand as the keying hand. The proposed system has two databases; the word database and the speed database. At each alphabetic keying, the candidates for the input word are narrowed down in the searching part using the word database. After that, the proposed system displays the candidates in order of priority calculated in the ordering part using the speed database. The proposed system calculates the priority using the estimated number of keyings that had been supposed to be input with non-keying hand. When a non-alphabetic key is entered, the proposed system shifts into the word-selection phase, and the user selects the input word. When the user enters the first key of the next word, the proposed system outputs the selected word and the non-alphabetic key which have been input at last. If the target word is displayed at the top and selected by default, the user can begin inputting the next word immediately without extra action in word-selection phase. Therefore, if the proposed system can estimate input word accurately, the input action of proposed method is almost like that of full keyboard, and the system does not impose extra operations on the user.

As described above, the proposed system displays the candidates for input word through the selection part and the ordering part. We explain each part in the following.

3.2 Searching Part

The proposed system looks up the input word in the word database prepared in advance. Normal dictionaries contain only basic forms of each word though the user uses the inflected forms such as plural, past, and progressive. Therefore, The proposed system crawls web pages on Wikipedia and constructs the word database. This crawling also accepts neologisms and proper names.

The proposed system employs a trie tree structure for searching a word from the word database. A trie tree is an ordered tree data structure that is used to store an associative array. No node in the tree stores a key associated with that node, but its position in the tree defines a key. All the descendants of a node have a common prefix of the string associated with that node, and the root node is associated with the empty strings. Figure 5 shows the trie tree structure that is employed when one's right hand is defined as the keying hand. The root node has eleven child nodes associated with a letter and one leaf node associated with empty string. The eleven letters are 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'u', and 'y', which are supposed to be input with the keying hand (right hand in this case) with a full keyboard. The eleven child nodes also have eleven child nodes associated with a letter and one leaf node. The leaf nodes store words that consist of all letters of the key associated with the position of that node and zero or more letters that are supposed to be input with the non-keying hand (left hand in this case). For example, a leaf node whose superiors are associated with 'i' and 'y' stores the words that match the regular expression "[a-gq-tv-xz]*i[a-gq-tv-xz]*y[a-gq-tv-xz]*" such as "variety" and "sideways". The leaf node that is directly connected to the root node stores the words that is composed of only input with the non-keying hand.

The proposed system initiates a search at the root node. When the user hits a alphabetic key, the searched node switches to the child node associated with the corresponding letter. The candidates at this time are the words that are stored in all leaf nodes of sub-tree whose root node is the searched node. When a non-alphabetic key (e.g., space key and period key) is entered, the candidates are switched to the words that are stored in the leaf node connected with the searched node directly. So when the user inputs keys

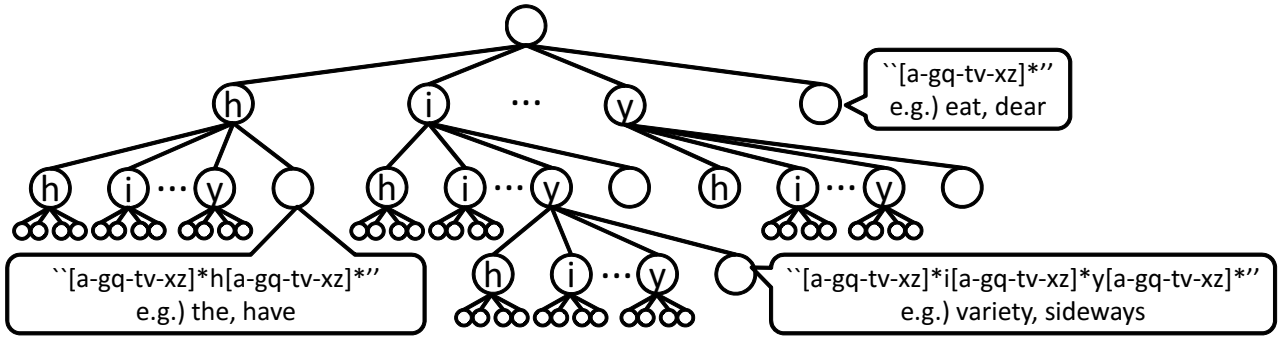

Figure 5: The trie tree

'k_0' and 'k_1' with the right hand, the candidates are the words that match the regular expression "[a-gq-tv-xz]*k_0[a-gq-tv-xz]*k_1[a-z]*". And then, when the user inputs a non-alphabetic key, the candidates are the words that match the regular expression "[a-gq-tv-xz]*k_0[a-gq-tv-xz]*k_1[a-gq-tv-xz]*".

However, there is a problem that the word database cannot encompass all words completely. There may be some neologisms and proper nouns that are not contained in the word database. Therefore, the proposed system has the inverting function that employs a input method like Wearable Half Keyboard described in Section 2. Specifically, a key input are converted to the input of correspond key shown in Figure 2 while the invert key is pressed. For example 'j' key input is converted to 'f' key input, and 'n' is converted to 'b'. This inverting input is also beneficial for search refinement. The proposed system sometimes displays a huge number of candidates when only few key inputs with the keying hand is obtained. In particular, when there is no input with the keying hand, the candidates are all words that consist of only the letters that are supposed to be input with the non-keying hand with a full keyboard. A letter that is entered with the inverting function excludes the words that do not contain that letter from the candidates. For example, if the user inputs 'k', 'y', and 'o', and then inputs 's' with the inverting function, "keyboard" is excluded from the candidates though "keyboards" is displayed as a candidate.

3.3 Ordering Part

For ordering the candidates, the proposed system calculates the priority using the estimated number of keyings that is supposed to be entered with the non-keying hand. The keying interval between key entries with the keying hand, whose length is expected to be same as if the user used a full keyboard, is used for the calculation of priority.

The system collects data of typing speed of the user during use of the proposed system. Those data are stored in the speed database. One data contains a combination of interval time between the key entry with the keying hand and number of keyings with the non-keying hand. The interval when there is n keyings with the non-keying hand is denoted by t_n. For example, if the user input "keyboard " with one's right hand as the keying hand, the keying intervals between the 'k' and the 'y' and between the 'y' and the 'o' are stored as t_1, and the keying interval between the 'o' and the last Space key is stored as t_3. We set the maximum of n to 4 because there is rarely a word that has more than five successive input with one hand. Hereinafter, the data whose n is more than 5 is treated as t_4. The proposed system estimates the number of keyings with the non-keying hand using probability density function of Gaussian distribution that is calculated from the average t_{μ_n} and the standard deviation t_{σ_n} of each n. When the keying interval x is given, the likelihood $f_n(x)$ of inputting n key with the non-keying hand is calculated according to the following equation.

$$f_n(x) = \frac{1}{\sqrt{2\pi}t_{\sigma_n}} \exp\left(-\frac{(x-t_{\mu_n})^2}{2t_{\sigma_n}^2}\right)$$

After the calculation of $f_n(x)$ for all n (0 to 4), the proposed system calculates the weight $p_n(x)$, that is the probability that x contains n keyings with the non-keying hand.

$$p_n(x) = \frac{f_n(x)}{\sum_{i=0}^{4} f_i(x)} + \alpha$$

The α constricts the influence when the typing is a little slower than the average. Moreover, outliers of interval, which occur with a much slower failed typing, are excluded from the calculation. We set the threshold of outlier to $t_{\mu_4} + 2 \times t_{\sigma_4}$. The all p_ns ($n = 0, 1, 2, 3, 4$) that have outliers are determined to 1. Even if there is one outlier, the other intervals can rectify the estimated priority because it is usual that there are some input with the keying hand in a word.

At last, the priorities of each word are calculated. We set default priorities of each word W_{base} to the number of appearance while crawling for construction of word database. When the user inputs "$k_1 k_2 \cdots k_n k_{n+1}$" with the keying hand, $W(k'_0, k'_1, \cdots, k'_n)$, which is the priority of the word "[k'_0 keying with the non-keying hand]k_1[k'_1 keying with the non-keying hand]$k_2 \cdots k_{n-1}$[k'_{n-1} keying with the non-keying hand] k_n[k'_n keying with the non-keying hand]k_{n+1}" (k_{n+1} is a non-alphabetic key), is calculated according to the following equation, given the time interval between k_i and k_{i+1} x_i.

$$W(k'_0, k'_1, \cdots, k'_n) = W_{base} * \prod_{i=1}^{n} p_{k'_i}(t_i)$$

The proposed system does not consider x_0, the time interval before k_1, because x_0 includes the selecting time of the previous word. The word "$k_0 k_1 \cdots k_{n-1} k_n$" is added to candidate with the lowest priority for accepting the word that

Figure 6: The prototype

Figure 7: An usage example

does not registered to the word database. Using the inverting function described above, the user can input any words.

The user selects the input word by using up and down arrow keys defined in advance. We also implement the sort function to facilitate the selection when it is hard to find the input word in candidates. That function has two way for sorting candidates; alphabetical order and word-length order. The input word is determined when the user starts inputting the next word. Therefore, the additional action is not required if the input word is displayed at the top of the candidates. After the selecting, the speed database is updated. A set of the interval x and the number of keying with non-keying hand n' is registered. The x that is larger than $t_{\mu_{n'}}+2\times t_{\sigma_{n'}}$ is considered to be a outlier and excluded. The speed database has latest one hundred data of each n'.

3.4 Prototype

We implemented a prototype of the proposed method. Figure 6 shows the prototype of left half keyboard. We assigned '.', ',', [Backspace], and [Enter] key to 'h', 'y', '6', and [caps lock] key, respectively. Moreover, the down arrow key, the up arrow key, the sort key, and the invert key are assigned to [alt, opt], [win, cmd], [ctrl], and [shift] key, respectively. We assume that the prototype is attached to one's thigh as shown in Figure 7. The prototype can be of course used on a desk.

4. EVALUATION

For the validation of proposed method, we investigated input speed and proficiency. We collected the input data from 5 participants. All participants are intermediate or advanced users of keyboard, and all of them have the ability to touch-type. The experimental period was 5 days, and the participants had input a mail text that has about 200 words at each day with 5 input methods; full keyboard method, multi-tap method, Wearable Half Keyboard method, left-half proposed method, and right-half proposed method. When using the proposed method, the participants did not narrow the candidates using the invert key. The word database that we used in the experiment stores 140317 words. The results are described below.

4.1 Input Speed

Table 1 lists the input speed when using each method. "WPM" (Word Per Minute) means how many words (a word is standardized to five keystrokes) are entered per a minute. All participants input the most quickly when using full keyboard method, and the second most quickly when using left-half proposed method. Their input speed of all method was improved. That tendency was confirmed also when using full keyboard method, though they were supposed to be accustomed to that method. It might be because the use of proposed method made the participants more conscious of the key arrangement.

Almost all participants input faster with left-half proposed method than with right-half proposed method regardless of their dominant hand. The left half of the keyboard has more alphabetic keys and more frequently-used keys than the right half. There are non-alphabetic key such as ',' and '.' on the right half of the keyboard, and 15 out of 26 alphabetic keys are allocated to the left half. Additionally, the characters that are three most frequently used are 'e', 't', and 'a', and their appearance ratio is 12.7%, 9.1%, and 8.2%, respectively. The sum of appearance ratio of characters whose alphabetic keys are allocated to the left half of the keyboard is 58.9%. The more keyings are obtained, the more the proposed system can narrow down the candidates for the input word. The candidates narrowed down shortens the time for selecting the input word, and the input time as a result.

4.2 Breakdown of input time

The input time when using the proposed method can be divided into two phases; keying and selecting. Keying time is from the first keying with keying hand to the keying of non-alphabetic key, that is before the candidates are displayed. Selecting time is from the keying of non-alphabetic key to the first keying of the next word, that is before the input word is output. We investigated the breakdown of input time per a word. The results are listed as Table 2. We confirmed that the participants had been able to master the simulant keying with non-keying hand early, because the keying time had remained virtually unchanged. Meanwhile, the selecting time tended to be decreasing because the estimation accuracy of the input word had increased, or because the participants had been accustomed to the selecting operation.

However, the selecting time occupied the large part of the input time. The selecting time when using the left-half proposed method is shorter than when using the right-half. The percentage of selecting time when using the left-half proposed method was 80.2% at the first day, and 76.0% at the last day. When using the right-half, it was 84.1% at the first day, and 83.2% at the last day. From above, we confirmed that it is essential to reduce the selecting time for the achievement of faster input.

4.3 The presented position

To reveal the factor that decreased the selecting time, we investigated the calcuated priority of input word. Table 3 lists the rate of input words that are displayed at the top of the candidates and at the initial screen containing 10 words. The rate of input words that are displayed at the initial screen tended to be increasing, however the rate of input words that are displayed at the top was not converged. For being displayed at the top, the input word has to have the highest priority in the words that are stored in the same node of the trie tree. That means that the rate of the input words that are displayed at the top depends on the number of appearance while crawling for construction of word database. We found that increasing of the priority

Table 1: The input speed of each method

Participant	Input method	Input speed [WPM]				
		Day 1	Day 2	Day 3	Day 4	Day 5
A	Full keyboard	44.3	52.7	53.4	59.4	53.7
	Multi-tap	8.7	9.2	11.9	11.4	10.2
	Wearable Half Keyboard	12.2	14.5	18.1	14.8	17.0
	Left-half proposed	19.5	17.8	19.9	22.4	22.7
	Right-half proposed	10.4	12.1	11.4	14.3	17.2
B	Full keyboard	24.1	22.0	25.6	27.5	28.3
	Multi-tap	7.9	7.1	9.3	10.7	9.7
	Wearable Half Keyboard	9.0	9.3	10.3	12.0	12.7
	Left-half proposed	13.3	10.4	10.2	14.3	15.4
	Right-half proposed	8.5	7.9	10.0	8.5	11.1
C	Full keyboard	21.0	20.7	22.9	23.1	22.2
	Multi-tap	7.5	8.5	8.9	9.4	8.9
	Wearable Half Keyboard	9.1	9.3	9.3	10.7	12.4
	Left-half proposed	9.6	10.2	9.8	10.7	10.7
	Right-half proposed	7.2	9.7	7.9	8.1	8.7
D	Full keyboard	23.9	20.2	23.8	26.2	26.8
	Multi-tap	6.3	5.9	7.4	7.4	7.3
	Wearable Half Keyboard	7.5	9.3	10.0	10.0	10.5
	Left-half proposed	10.8	9.7	13.5	14.9	15.3
	Right-half proposed	9.1	8.1	10.8	11.1	12.3
E	Full keyboard	18.9	25.0	22.4	24.9	27.6
	Multi-tap	8.7	10.0	10.6	10.3	12.1
	Wearable Half Keyboard	6.8	9.6	9.2	11.2	13.7
	Left-half proposed	6.5	8.0	10.7	13.4	15.1
	Right-half proposed	7.0	8.6	10.4	10.7	10.1
Ave.	Full keyboard	26.4	28.1	29.6	32.2	31.7
	Multi-tap	7.8	8.1	9.6	9.8	9.7
	Wearable Half Keyboard	8.9	10.4	11.4	11.7	13.2
	Left-half proposed	11.9	11.2	12.8	15.1	15.8
	Right-half proposed	8.4	9.3	10.1	10.5	11.9

Table 2: The detailed input time

Method	Detail	Spent time [ms]				
		Day 1	Day 2	Day 3	Day 4	Day 5
Left	Keying	929	963	814	799	796
	Selecting	3758	3690	3245	2550	2577
Right	Keying	884	892	894	771	758
	Selecting	4673	4613	3747	3859	3749

Table 3: The presented position

Method	Position	Rate [%]				
		Day 1	Day 2	Day 3	Day 4	Day 5
Left	Top 1	53.6	46.0	44.6	50.4	55.5
	Top 10	87.9	88.8	90.1	93.1	94.9
Right	Top 1	46.2	46.6	45.7	58.6	37.9
	Top 10	84.9	81.6	86.0	85.2	85.0

Table 4: The estimation accuracy of keying number

Method	Accuracy [%]				
	1	2	3	4	5
Left	74.0	70.1	72.5	73.3	74.1
Right	72.4	73.5	73.6	79.4	75.2

of the input word shorten the selecting time, because there is a rough correlation between the rate of the input words that are displayed at the initial screen listed in Table 3 and the selecting time listed in Table 2. The average selecting time when the input word is displayed at the top was 1.4 seconds, and the average when the input word is displayed at the initial screen was 1.8 seconds. However, the average selecting time when the input word is not displayed at the initial screen was considerably long, 9.2 seconds. The expected input speed, which is calculated using only input time when the input words are displayed at the top, is 23.9 WPM when using the left-half proposed method and 21.9 WPM when using the right-half.

4.4 Estimation accuracy

Table 4 lists the estimation accuracy of the number of keyings with the non-keying hand. The accuracy means the rate that keying intervals containing i hitting with the non-keying hand have the largest p_i in p_0, p_1, p_2, p_3, and p_4. We had expected that the accuracy became higher as day went by because the more keying data was obtained. However, Table 4 indicates that the estimation accuracy had remained virtually unchanged. Then we investigated the calculation results on the last day, and the results are listed in Table 5. Table 5 indicates that the accuracy decreased as the input number of keying increased, and that the accuracy when the input number had been more than 2 was significantly low. Most of the misestimation is calculated lower than the actual input number of keyings. It might be caused because the slightly longer intervals that could not be excluded as outliers raised the average interval. Because using the raised average, the number of keyings is misestimated to be less than the actual number when the participants input smoothly.

5. DISCUSSION

We indicate the possibility of improving. As noted in above section, shortening the selecting time is necessary for reducing the input time.

5.1 Accuracy Improvement

It is important to estimate the number of keyings with non-keying hand with high accuracy. The estimation with high accuracy raises the displayed position of the input word, and shortens the selecting time. For the improvement of estimation accuracy, there are two approaches; exclusion of longer intervals, and employment of the other distribution for calculation. The exclusion of longer intervals can be accomplished by shortening the threshold of outlier, $t_{\mu_i} + 2 \times t_{\sigma_i}$. There is a possibility to improve the estimation accuracy by employing not Gaussian distribution but Poisson distribution, because there is a lower limit of the keying interval.

If the left-half proposed method could estimate the number of keying with the non-keying hand with 100 % accuracy, the 67.0 % of all input words were displayed at the top, and the 97.2 % were displayed at the initial screen. If the right-half, the 62.7 % were displayed at the top, and 94.8 % were displayed at the initial screen. Moreover, if the left-half proposed method could somehow estimate the number of keying before the first keying with the keying hand with 100 % accuracy, the 87.2 % were displayed at the top and the 99.4 % were displayed at the initial screen. If the right-half, 78.6 % were displayed at the top and 98.5 % were displayed at the initial screen.

5.2 Predictive Algorithm

The proposed method considers only two factors in calculating the priority; the keying interval and the appearance number while crawling. The input word, however, is influenced by the previous word. Moreover, when the user is responding to a mail, it is influenced by the word contained in the received mail. Many predictive algorithms have been proposed such as [10], which is popular as a input method implemented in mobile phones. The accuracy of priority calculation might be improved by introducing such algorithms. N-gram word prediction[11] are one of the word prediction methods used widely. In n-gram word prediction method, the previous $n-1$ words are used to predict the current (n^{th}) word. We investigated the presented position of input word by the calculation using n-gram method; unigram ($n = 1$) and bigram ($n = 2$) whose text corpora are all article on Wikipedia. The unigram method is equal to W_{base}, which is default priority of the proposed method before being calculated. The results are listed in Table 6. A comparison between Table 3 and Table 6 indicates that the proposed method could displayed at the higher position than using only default priority. Though the results of the bigram method were a little better than that of the proposed method, the estimation accuracy of the proposed method would be able to be improved as described above. Therefore, we think that the proposed method, the estimation of the keying number using the keying interval, is an efficient method. Moreover, combining n-gram with the proposed method would increase the accuracy of word estimation.

5.3 Suited Dictionary

We used the only word database that is constructed by collecting the sentences in Wikipedia. However, the input speed of the proposed method might be improved by using the other different database based on usage scenes. In the experiment, we let the participants input the sentences that are likely appeared in a mailer application. Therefore, first-person and second-person words such as "I", "my", "you", and "your", which rarely appear on Wikipedia, often appeared. These words have few letters that is hit by left hand, then the presented position is much influenced by the W_{base} (the appearance number while crawling) because the candidates are not narrowed down sufficiently by the keying interval. The user should be able to compose a mail by using W_{base} considering the appearance frequency on mails. The word database that considers one's taste or usage history also can help to order.

6. CONCLUSION

In this paper, we propose a method to miniaturize a keyboard by excluding the half. The proposed method leverages

Table 5: The results of estimated keying (5th day)

Input	Output	0	1	2	3	4
Left	0	85.6%	11.8%	2.0%	0.6%	0.0%
	1	17.3%	64.5%	14.7%	3.0%	0.5%
	2	4.1%	59.2%	32.7%	4.1%	0.0%
	3	0.0%	25.0%	50.0%	20.0%	5.0%
	4	0.0%	0.0%	50.0%	50.0%	0.0%
Right	0	90.3%	9.7%	0.0%	0.0%	0.0%
	1	22.2%	66.7%	9.8%	0.7%	0.7%
	2	3.1%	44.6%	41.5%	7.7%	3.1%
	3	0.0%	22.9%	54.3%	14.3%	8.6%
	4	0.0%	0.0%	0.0%	28.6%	71.4%

Table 6: The presented position when using N-gram

Method	N-gram	Rate [%]	
		Top 1	Top 10
Left	Unigram	49.5	83.5
	Bigram	57.8	92.4
Right	Unigram	35.2	82.9
	Bigram	46.5	91.7

the ability of keyboard input, which is mastered previously. When using the proposed method, one hand hits keys on the device as if the user uses a full keyboard, and the other hand hits somewhere outside of the device. Though there are only half keys of a full keyboard, the system estimates the input word using the keying interval. The results of evaluation using a prototype indicated the effectivity of proposed method, and we found the possibility to improve. Our future work is to improve the input speed and to adjust for another languages.

7. REFERENCES

[1] I. Scott MacKenzie and R. William Soukoreff: Text Entry for Mobile Computing: Models and Methods, Theory and Practice, Human-Computer Interaction, Vol. 17, No. 2–3, pp. 147–198 (2002).

[2] B. Thomas, S. Tyerman, and K. Grimmer: Evaluation of Text Input Mechanisms for Wearable Computers, it Virtual Reality, Vol. 3, No. 3, pp. 189–199 (1998).

[3] Y. Liu, X. Liu, and Y. Jia: Hand-Gesture Based Text Input for Wearable Computers, Proceedings of the ICVS 2006 IEEE International Conference on Computer Vision Systems, p. 8 (2006).

[4] R. Rosenberg and M. Slater: The Chording Glove: A Glove-Based Text Input Device, IEEE Transactions on Systems, Man, and Cybermetics, Part C: Applications and Reviews, Vol. 29, No. 2, pp. 186–191 (1999).

[5] Matias Wearable Half Keyboard: http://www.halfkeyboard.com/wearable/index.html.

[6] K. Lyons, T. Starner, D. Plaisted, J. Fusia, A. Lyons, A. Drew, E. W. Looney: Twiddler Typing: One-Handed Chording Text Entry for Mobile Phones, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 671-678 (2004).

[7] E. Matias, I. S. MacKenzie, and W. Buxton: Half-qwerty: Typing with One Hand using Your Two-Handed Skills, Companion of the CHI 1994 Conference on Human Factors in Computing Systems, pp. 51–52 (1994).

[8] R. Joyce and G. Gupta: Identity authentication based on keystroke latencies, Communications of the ACM, Vol. 33, No. 2, pp. 168–176 (1990).

[9] R. E. Lewand: Cryptological Mathematics, Mathematical Society of America.

[10] T. Masui: POBox: An Efficient Text Input Method for Handheld and Ubiquitous Computers, In Proceedings of the HUC 99 International Symposium on Handheld and Ubiquitous Computing, pp. 289–300 (1999).

[11] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai: Class-Based N-Gram Models of Natural Language, Computational Linguistics, Vol. 18, No. 4, pp. 467–479 (1992).