Journal of Information Processing Vol.22 No.2 410–416 (Apr. 2014)

[DOI: 10.2197/ipsjjip.22.410]

Regular Paper

HMM-Based Probabilistic Flick Keyboard Adaptable to Individual User

Toshiyuki Hagiya[1,a)] Tsuneo Kato[1,b)]

Received: April 1, 2013, Accepted: December 4, 2013

Abstract: To provide an accurate and user-adaptable software keyboard for touchscreens, we propose a probabilistic flick keyboard based on hidden Markov models (HMMs). Touch and flick operations for each character are modeled by HMMs. This keyboard reduces input errors by taking the trajectory of the actual touch position into consideration and by user adaptation. We evaluated the performance of an HMM-based flick keyboard and maximum-likelihood linear regression (MLLR) adaptation. Experimental results showed that a user-dependent model reduced the error rate by 28.3%. In a practical setting, the MLLR adaptation to a specific user with only 10 words reduced the error rate by 16.6% and increased the typing speed by 11.9%.

Keywords: PDA interfaces, usability, input technologies, personalization

1. Introduction

With the recent popularity of smartphones and tablets, keyboard input on a touch screen display has become essential. A touch screen display provides the benefits of a large display area and a flexible software keyboard that can change its orientation, switch languages, and modify the key layout. Despite these benefits, a number of users still prefer physical keyboards because they make more errors with a software keyboard [1], [2]. Major reasons for the errors are the lack of tactility and the small keys [3]. A software keyboard is sometimes too small for fingers to hit the correct keys [4].

Various keyboards have been proposed to improve the usability. The flick keyboard, which determines a character through a combination of the touched area and the following flick direction, is a popular method, especially for Japanese. Formerly, the standard input method for Japanese was the toggle input which determines a character by pressing one of the ten keys several times to switch the characters assigned to the key. The flick keyboard has become popular because it is possible to switch characters with a single flick operation (**Fig. 1**) instead of pressing a key multiple times. However, the detection areas and the flick directions are fixed on a conventional flick keyboard, although the distribution of the touch positions and trajectories change depending on various factors such as the size of the user's hand, and the user's hand posture. Moreover, the actual trajectory is not a straight line but an arc, which causes errors in interpreting the flick directions. Users have different preferences in hand posture and often change the hand posture depending on their situation [5]. For these reasons, it is important to adapt the keyboard to the user's individual characteristics.

[1] KDDI R&D Laboratories, Inc., Fujimino, Saitama 356–8502, Japan
[a)] to-hagiya@kddilabs.jp
[b)] tkato@kddilabs.jp

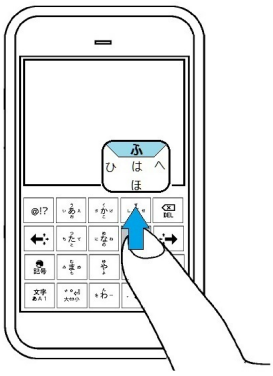


Fig. 1 Input screen with a flick keyboard.

We propose a hidden Markov model (HMM) based probabilistic flick keyboard that takes into consideration the time series of the touch positions while the user is touching the screen. HMMs, which are known for their use in temporal pattern recognition such as speech recognition, are effective in pattern recognition of signal sequences that extend in time and are continuous [6]. Evaluation of the time series by a probabilistic model is expected to improve the typing accuracy because the features of each user's touch and flick operations are taken into account. In addition, to adapt the touch distributions modeled by HMMs to an individual user and a specific hand posture, we adapted the HMMs by using a maximum-likelihood linear regression (MLLR) with a small amount of data.

This paper is organized as follows. In Section 2, we describe the related research. In Section 3, we describe the proposed technique in detail. Next, we evaluate the basic performance of an HMM-based flick keyboard off-line in Section 4 and evaluate its performance on an actual mobile device in Section 5. Then, we discuss the performance in comparison with other studies in Section 6. Finally, we present our conclusions in Section 7.

2. Related Research

Previous studies on reducing typing errors fall into two categories: language-model-based and touch-model-based. In the language-model-based methods, the next characters or words are predicted and displayed [7], [8], and the next keys with high probabilities are displayed with bigger key sizes [9]. However, the performance of the language-model-based method is heavily dependent on a dictionary. Unregistered words, such as new words, abbreviations, or colloquial expressions, are ignored.

In contrast, in a touch-model-based method examined in one study, a wealth of touch data that were collected through a smartphone game were analyzed, and patterns were identified in users' touch offsets [10]. Another study found that users touched the surface with a consistent offset from targets. Subsequent studies showed that a user's perceived contact point was approximated by the center of the fingernail, which was above the actual contact point along the finger's axis [11]. Other studies proposed transforming the keyboard layout according to actual touch distributions [12], [13]. In these studies, the center of each key moved to follow the centroid of actual touch distributions, and the boundary was set in the middle of the keys. However, the studies reported that some users were confused by the dynamic changes in key layout.

Furthermore, some studies proposed methods of transforming the detection areas using both the language-model-based and touch-model-based approaches. The probability that a particular character would be next was assumed to be the product of two probabilities, one from the language model and one from the touch model [14], [15]. Reference [14] used character n-grams for the language model and the Gaussian mixture model (GMM) of stylus input positions in the touch model. Reference [15] proposed methods of transforming the detection areas based on the typing history in the language model, although the central region of the key remained fixed to suppress excessive transformation.

Moreover, a study showed that users often change their hand postures [5]. Hence some studies analyzed the touch distributions for different hand postures, and proposed a keyboard that would adapt to users and hand postures [16], [17]. Reference [16] proposed to switch to various keyboard models by detecting the hand posture based on tap sizes and time elapsed between taps. Reference [17] defined a hierarchical submodel for each user and hand posture, in order to adapt each model according to the amount of available training data. Finally, some studies analyzed the difference in touch distributions exhibited while sitting and walking, and proposed adaptation techniques for these contexts [18], [19].

All of these studies involved keyboards where one touch corresponded to one character. Some keyboards were proposed that used input trajectories, such as "VOKB [20]", "Swype [21]" and various flick keyboards. However, these keyboards were not studied for their adaptability to methods that use touch distributions, although improving them by changing the dictionary or the layout of the keys was investigated.

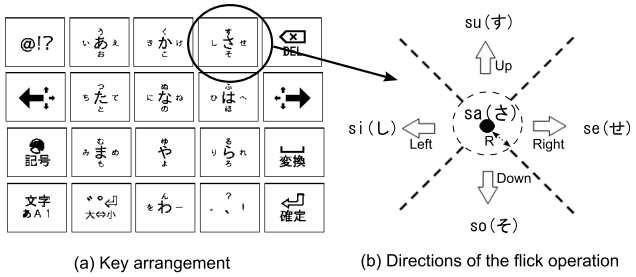
3. Probabilistic Flick Keyboard based on HMMs

The flick keyboard, which determines a character through a combination of the touched area and the following flick direction, is a popular method, especially for Japanese. Japanese characters can be input by a vowel and a consonant preceding the vowel. They can be organized in a matrix with ten consonants of the rows and five vowels of the columns in **Table 1**. The flick keyboard assigns the consonants to the keys, and assigns the vowels to the directions. The characters which have vowel "A" correspond to staying at the center, and the others which have vowels "I", "U", "E", "O" correspond to the four flick directions as shown in **Fig. 2** in a conventional Japanese flick keyboard. However, the detection areas and the flick directions are fixed on a conventional flick keyboard although the distribution of the touch positions and trajectories change depending on various factors.

Hence, we propose an HMM-based probabilistic flick keyboard that uses the time series of the touch positions and adapts to the user. HMMs are effective for pattern recognition in signal sequences that can extend in time and be continuous, such as speech and gestures [6]. It is considered that HMM is useful for a flick keyboard because touch trajectories in flick operations also have these characteristics. The HMM-based flick keyboard determines a character by evaluating the time series of the touch positions while a user is touching the screen, whereas the conventional flick keyboard senses only the positions of touch and release events. The key areas and flick directions are defined as shown in Fig. 2. Using a time series and a probabilistic model to determine a character is expected to improve the typing accuracy because the features of the individual user's touch and flick operations can be taken into consideration. In this method, the flick operations of a character are modeled by a left-to-right HMM as shown in **Fig. 3**. The flick operations of each character are modeled by a HMM. The probability of each character is calculated by searching the HMM network with the Viterbi algorithm, as described later. The search determines which character has the highest probability at the terminal state of the last time frame. Each state has mixtures

Table 1 Japanese characters represented by Roman alphabet letters.

	A	I	U	E	O
	a (あ)	i (い)	u (う)	e (え)	o (お)
K	ka (か)	ki (き)	ku (く)	ke (け)	ko (こ)
S	sa (さ)	si (し)	su (す)	se (せ)	so (そ)
⋮			⋮		
W	wa (わ)	-	-	-	wo (を)



(a) Key arrangement (b) Directions of the flick operation

Fig. 2 Key arrangement and directions of the flick operation with a conventional Japanese flick keyboard.

of normal distributions for four-dimensional feature parameters. The four parameters are two-dimensional time series of the touch coordinates (x_t, y_t) and the displacement $(\delta x_t, \delta y_t)$ between the current position and the initial position of the touch.

Next, we explain the Viterbi algorithm used in this method. The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence corresponding to a particular state. Suppose we are given a HMM with N states and input a character with a flick operation from time 0 to T. The most likely sequence of the state corresponding to the observations is given by the following relations;

$$\delta_{t,j} = \begin{cases} \pi_i & (t = 0) \\ \max_i[\delta_{t-1,i} a_{ij}] \cdot b_i(o_t) & (1 \le t \le T) \\ & (0 \le i \le j \le N) \end{cases} \quad (1)$$

For the last time frame, the probability of a character on the most likely path (P) is obtained from the following equation.

$$P = \max_i[\delta_{T,i}] \quad (2)$$

Here $\delta_{t,j}$ is the probability of the most likely state that accounts for the first t observations and ends in state j. π_i is the initial probability of being in state i. o_t is the feature parameters for time t. a_{ij} is the probability of transitioning from state i to state j, and $b_i(o_t)$ is the observation probability for a symbol in state i. $b_i(o_t)$ is a mixture of the normal distribution with a diagonal covariance matrix, the same as the GMM shown by the equation:

$$b_i(o_t) = \sum_{k=1}^{K} \omega_k N(o_t|\mu_k, \Sigma_k) \quad (3)$$

where K is the number of mixtures of normal distributions. $N(o_t|\mu_k, \Sigma_k)$ denotes a normal distribution with a mean vector μ_k and a covariance matrix Σ_k. The connection between the models is ergodic. No language model is used.

In addition, we evaluate the performance of MLLR with various number of words used in the adaptation to individual users and typing postures. MLLR is a model adaptation technique that estimates a set of linear transformations for the mean and covariance parameters with a small amount of data [22]. In MLLR, both the mean and covariance parameters are transformed and re-estimated as

$$\mu'_k = \mathbf{A}\mu_k + b \quad (4)$$

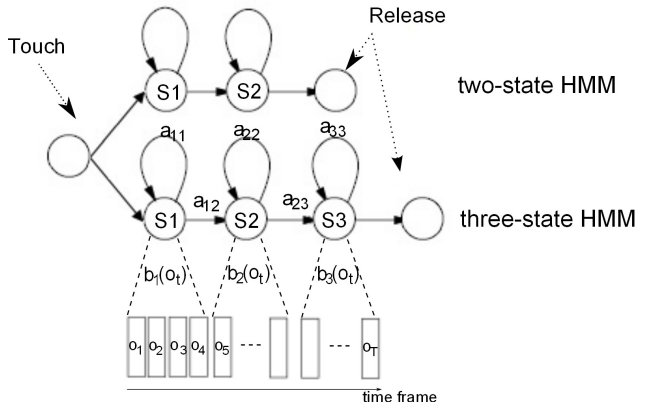


Fig. 3 Configuration of the HMM network and assignment of the time series observations.

$$\Sigma'^{-1}_k = \mathbf{H}^T \Sigma_k^{-1} \mathbf{H} \quad (5)$$

where μ_k is the mean vector, Σ_k is the covariance matrix of Eq. (3), and μ'_k and Σ'_k are the adapted mean vector and covariance matrix, respectively. (**A**, b, **H**) are estimated in a maximum-likelihood (ML) manner.

4. Evaluation of Flick Keyboard based on HMMs

In this section, we first analyze the touch distributions for each flick operation with each hand posture. Second, we evaluate the performance of the HMM-based flick keyboard and the MLLR adaptation off-line by comparing their input accuracy with that of the conventional flick keyboard.

4.1 Data Collection

Participants input words in whatever hand posture they normally use of those as shown in **Fig. 4** (a)–(e), while sitting a chair. Hand postures (a) and (b) are operations with one thumb, i.e. holding and typing with one hand. Hand postures (c) and (d) are operations with one index finger, or holding with one hand and typing with the other. Hand posture (e) uses both thumbs for typing while holding with both hands. Hand postures (a) and (c) are typing with the right hand. The device obtains the touch coordinates (x_t, y_t) by using a standard Android API, i.e., an application program receives the finger motion as down/move/up events. Then the device records the data sampled at 50 Hz because the raw data have randomness in timing, and the time information cannot be correctly taken into consideration for HMM. We considered that 50 Hz was high enough to detect operations because 97.7% of the operations took more than 0.10 sec (average 0.32 sec). We employed a conventional flick keyboard, in which the key areas were defined in a rectangular region as shown in Fig. 2 (a). The flick motions were in the standard four directions, plus staying at the center when the movement was less than 25 pixels, as shown in Fig. 2 (b). The participants typed 100 words per set, and typed six sets with a break between each set. Six hundred words with a length of 3–6 (average 4.25) characters

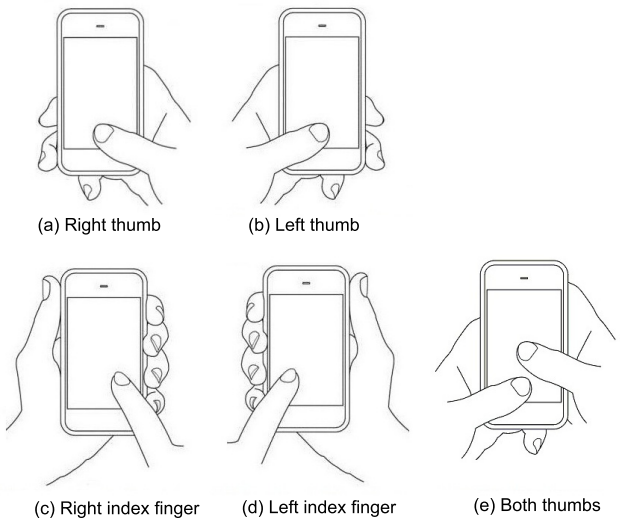


Fig. 4 Hand postures.

Table 2 Participants using each hand posture.

Participants' hand posture	male	female	Total
(a) Right thumb	8	2	10
(b) Left thumb	3	0	3
(c) Right index finger	3	2	5
(d) Left index finger	0	3	3
(e) Both thumbs	0	0	0

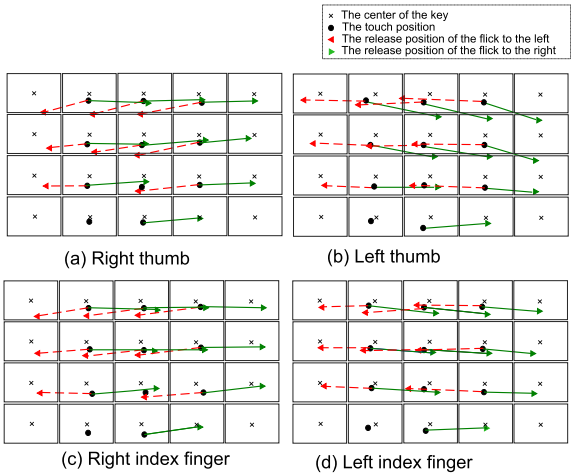


Fig. 5 Touch distribution of the horizontal flick operation with each hand posture.

were randomly chosen from JUMAN dictionary [23] for each participant. When participants typed incorrect characters, they were permitted to correct by using the delete key or the back key and retyping. We chose twenty-one participants (fourteen male, seven female) between the ages of 23 and 60 (average 31.8), who had experience with a flick keyboard (but, were not researchers in this field). Fifteen participants (eleven male, four female) were right-handed and the rest were left-handed. **Table 2** shows the breakdown of participants using each hand posture. None used (e). The device was a Nexus S running Android 4.0 with a 4.0-inch screen whose resolution was 800×480 pixels (223 ppi). The size of a key was 84×96 pixels.

4.2 Distribution of Users' Touch Positions

In this section, participants' touch distributions are shown for each hand posture because there is a large difference in distributions for each hand posture. **Figures 5** and **6** show the averages of all participants' touch and release positions for each flick direction in each hand posture. With respect to the touch position, similar tendencies were observed in the same typing hand. Horizontally, the center of the touch positions leaned in the direction of the typing hand in all hand postures. The biases were bigger for one thumb than one index finger. The mean horizontal differences were 5.9 pixels (SD = 13.0) with one index finger, and 10.0 pixels (SD = 12.6) with one thumb. On the other hand, the vertical center of the touch positions generally leaned to the bottom. The mean vertical differences were 19.8 pixels (SD = 16.4) with one index finger, and 16.4 pixels (SD = 17.3) with one thumb. The reason of these differences was that the participants set their typing hand in a home position that was different for each hand posture, and then moved their hand as little as possible.

Next, **Table 3** shows the movement between the touch and the

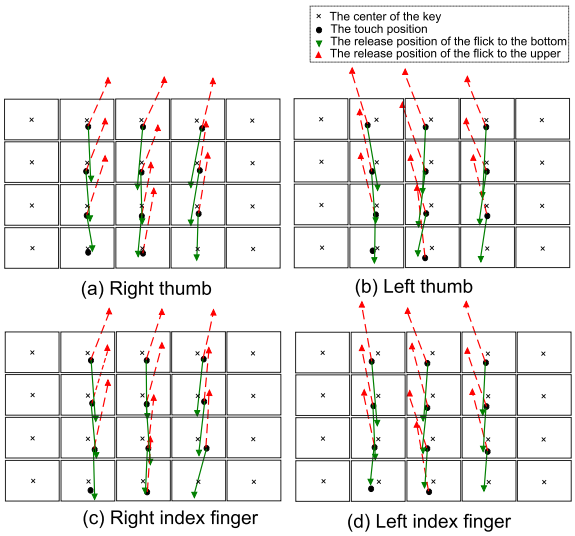


Fig. 6 Touch distribution of the vertical flick operation with each hand posture.

Table 3 Movement between the touch and the release positions of the flick directions with each hand posture.

Hand posture	Movement between the touch and the release position (δx, δy)			
	upper	right	bottom	left
(a)	(20.0, 105.5)	(101.1, 5.2)	(−7.7, −98.0)	(−94.3, −18.7)
(b)	(−25.4, 112.5)	(103.1, −20.9)	(5.7, −103.1)	(−105.4, 5.8)
(c)	(18.0, 110.0)	(104.7, 5.2)	(−2.4, −100.4)	(−93.5, −9.6)
(d)	(−25.0, 110.5)	(102.1, −10.8)	(2.0, −101.1)	(−103.4, 5.0)

release position for each direction with each hand posture. The flick directions were leaned to the upper right side for operations towards the upper and the right directions, and to the bottom left side for operations towards the left in hand postures (a) and (c) that used right-hand typing. A tendency to reverse left and right was seen in hand postures (b) and (d) that used left-hand typing. The displacements were larger with the one-thumb posture than the one-finger posture, especially with operations towards the outside or upper side. The reason was that the movement of the hand was limited in the one-thumb posture, so that the input trajectory drew not a straight line but an arc.

4.3 Training and Evaluation Method

We trained the HMM models with the collected data sampled at 50 Hz, and compared three kinds of user models: user-dependent (UD), user-independent (UI), and user-adaptation (UA). The UD model was trained with 500 words of input by the user him/herself using the Baum-Welch algorithm. The UI model was trained with 12,000 words of input by twenty other participants. The UA model was obtained by MLLR adaptation based on the UI model with a limited number of words of input by the user him/herself. The number of states of the HMMs was set to two or three because the minimum number of touch events acquirable in a series of operations by the Android API is two by a touch operation, and three by a flick operation. Two-state HMMs were assigned to the characters which have vowel "A", corresponding to staying at the center, and three-state HMMs were assigned to the characters which have vowels "I", "U", "E", "O", corresponding to the four flick directions. The number of mixtures of the normal distribution varied in the range of 1-6. The number of words for the

Table 4 Average ER and ERR for all participants with different training models.

HMM model	recognition error rate (ER) [%]	error reduction rate (ERR) [%]
A	3.20	6.71
B	2.47	28.3
E_0	3.43	-

MLLR adaptation was varied in the range of 5-100. The number of classes for the regression tree was 8. The cross validation (CV) for evaluation was conducted by leave-one-out, that is 6-fold for each user for the UD model and 21-fold for the UI and UA models. We evaluated their performances by comparing their error rate (ER) [24] with the input error rate of the conventional flick keyboard (E_0).

4.4 Experimental Results of the Off-line Evaluation

4.4.1 Baseline Evaluation for Model Selection

To investigate the validity by using time series data, we compared the method that was implemented by the two-state HMM models that did not have self-transition and were just assigning the touch and release position to a particular state (Model A), with the proposed method with the two-or-three state HMM models, by using the time series of touch positions (Model B). We used the UD model for evaluation and set the number of mixtures at 4.

As shown in **Table 4**, the average E_0 of all participants was 3.43%, and ER was 3.20% with Model A and 2.47% with Model B. In other words, the UD model reduced the ER by 6.71% and 28.3%. The performance of Model B was better than that of Model A. A Welch's T-test ($p < 0.05$) showed that there were significant differences in ER for each model. Therefore, we use Model B hereafter.

4.4.2 Evaluation of the HMM-based Flick Keyboard Off-line

First, we focus on the input accuracy of the UD and the UI models by changing the number of mixtures of the normal distribution. **Figure 7** shows the relationship between the number of mixtures and the average ER of the UD and the UI models. The number of mixtures that best reduced the ER was 2 for the UD model, and 4 for the UI model. No great improvement was produced by larger numbers. The average E_0 of all participants was 3.43%, and the ER was 2.47% for the UD model and 3.04% for the UI model. In other words, the UD and the UI models reduced the ER by 28.3% and 11.4%, respectively, compared with E_0. A Welch's T-test ($p < 0.05$) showed that there were significant differences in ER for each keyboard. The reason why the ER of the UD model was lower with a small number of mixtures was that the touch positions were stable for each individual participant because the same hand posture was used. The ER of the UI model was higher than E_0 with a small number of mixtures because the HMM model was trained by data from various participants who used various hand postures. The ER of the UI model became lower than E_0 with more than two mixtures. We consider that the UI model was able to reduce the ER because the touch distributions were more or less similar within the same hand posture.

Next, we describe the influence of the number of words on the UA model. **Figure 8** shows the relationship between the number

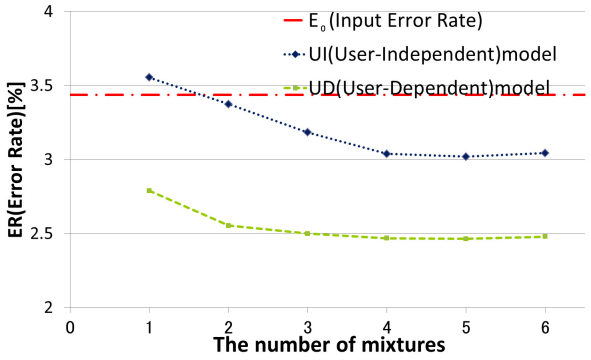


Fig. 7 Average ER for all participants versus the number of mixtures of normal distributions.

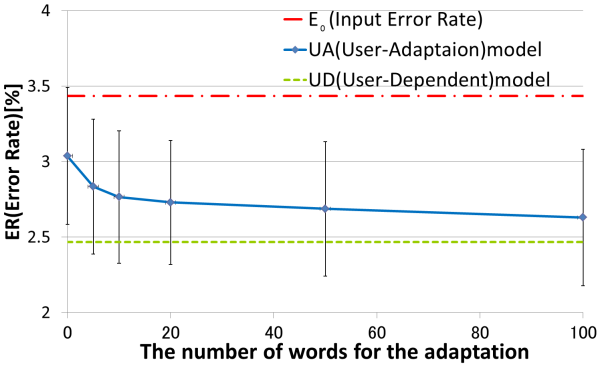


Fig. 8 Average ER for all participants versus the number of words used for adaptation; vertical bars show standard deviation.

of words used for adaptation and the average ER for all participants. The ER was greatly reduced by adaptation with 5–20 words. The ER as more words were used approached that of the UD model asymptotically. The results show that the adaptation by the MLLR is effective even in early stages of its use, because the MLLR is able to improve the performance with a smaller amount of training data than the UD model requires.

5. Evaluation on the actual mobile device

In this section, we evaluate the total performance of the HMM-based flick keyboard on a mobile device, assuming usage in the real environment.

5.1 Training and Evaluation Method

Participants typed six sets of 60 words each using the conventional flick keyboard and the HMM-based keyboard. The words, with lengths of 3–6 (average 4.25) characters, were randomly chosen from the JUMAN dictionary [23] for each participant. The presentation order of the keyboards and the set of words were determined at random. We evaluated their performance by comparing each ER and words per minute (WPM) using the following equation [24], [25].

$$WPM = \frac{L-1}{s} \times 60 \times \frac{1}{\bar{l}} \tag{6}$$

Here, s is the time in seconds measured from the first key press to the last in a set, including deletes and other edit and modifier keys. L is the length of the final text in the set entered by the participant in a set. Although $\bar{l}$ means the average length of

Table 5 Breakdown of participants by hand posture.

Participants' hand posture	male	female	Total
(a) Right thumb	4	0	4
(b) Left thumb	2	0	2
(c) Right index finger	1	2	3
(d) Left index finger	0	1	1
(e) Both thumbs	0	0	0

Table 6 Average ER, ERR and WPM for all participants for every keyboard (standard deviation in parentheses).

Keyboard	recognition error rate (*ER*) [%]	error reduction rate (*ERR*) [%]	WPM
Normal	3.80 (0.98)	-	17.0 (4.90)
HMM	3.17 (0.90)	16.6 (6.02)	19.3 (5.04)

a word, the constant 5 is conventionally used in this field [24]. The constant 60 is the number of seconds per minute. The HMM model was the UA model that was adapted on the device based on the UI model of the previous section. Data for adaptation were the correct input data of the first 10 words of each set. Correct data means that a character identical to the displayed character was input and not deleted. We evaluated 50 words, skipping the first 10 words of each set. We chose ten participants (seven male, three female) between the ages of 24 and 57 (average 33.0) who had not been selected in the previous section. Seven participants (five male, two female) were right-handed and the rest were left-handed. **Table 5** shows the breakdown of participants who used each hand posture.

5.2 Experimental Results Obtained on Actual Mobile Device

Table 6 shows the average ER and WPM for every keyboard. The ER and WPM were respectively 3.80% and 17.0 with the conventional flick keyboard, and 3.17% and 19.3 with the HMM-based flick keyboard. In other words, the HMM-based keyboard reduced the ER by 16.6% and increased WPM by 11.9% compared to the conventional flick keyboard. The improvement in WPM was due to the fact that the time needed to correct input errors was reduced. A Welch's T-test ($p < 0.05$) showed that there were significant differences in ER and WPM for each keyboard. These results show that the proposed method is also effective on an actual mobile device.

6. Discussion

The HMM-based flick keyboard decreased the ER and increased the WPM significantly. Moreover the error reduction rate in our study was around 20% for the flick input, which is similar to that in some of the latest studies [16], [17] on touch input, such as qwerty. Therefore, we consider that our approach is effective for the flick keyboard. Additionally, even more performance improvement can be expected from using an n-gram language model in addition to the results of these studies.

On the other hand, mobile devices are used with various different hand postures in the real environment, whereas the participants in our experiment did not switch hand postures. References [16] and [17] propose to detect hand posture by touch distributions and to change models based on the detected hand posture. The rate of detection of hand posture was less than 90% in these studies, and it was stated this error rate might not degrade the performance. If the rate is high enough as argued in Ref. [16], the model should be adapted periodically with more training data in our methods. In addition, more improvement is expected by the adaptation based on the model which is defined as each hand posture instead of UI model because touch distributions of flick operations are different for hand posture shown in Section 4.2. In contrast, if the rate of detection is not enough, the model should continue being adapted with the latest input data in a short interval.

Besides, users typed in various situations, such as while standing and walking in the real environment, although we focused on typing only while sitting. Reference [18] showed that on the whole, the average touch positions while walking were similar to those of while sitting, although the variance was greater. Meanwhile, Ref. [19] analyzed the differences in detail and showed that touch distributions of the touch operation differed depending on which side of the foot struck the ground. Considering these differences, it seems that the adaptation is difficult without the use of any external sensors. the ways that sensor information might be used in the adaptation to various contexts.

7. Conclusion

In this paper, we investigated an HMM-based flick keyboard that uses the time series of the touch coordinates to select the most probable intended character. The results showed that the proposed method with a user-dependent model reduced the error rate by 28.3% compared to the conventional flick keyboard offline. Moreover, the MLLR adaptation with 10 words reduced the error rate by 16.6% and increased the typing speed by 11.9% on an actual mobile device. As a future area of study, we will investigate the application of these findings in the real environment and in various contexts.

References

[1] Lee, S. and Zhai, S.: The performance of touch screen soft buttons, *CHI'09 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.309–318 (2009).

[2] Sears, A., Revis, D., Swatski, J., et al.: Investigating touchscreen typing: the effect of keyboard size on typing speed, *Behaviour &Information Technology*, Vol.12, pp.17–22 (1993).

[3] Hoggan, E., Brewster, S. and Johnston, J.: Investigating the effectiveness of tactile feedback for mobile touchscreens, *CHI'08 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.1573–1582 (2008).

[4] Sunghyuk, K., Donghum, L. and Min, K.C.: Effect of key size and activation area on the performance of a regional error correction method in a touch-screen QWERTY keyboard, *Proc. International Journal of Industrial Ergonomics*, Vol.39, No.5, pp.888–893 (2009).

[5] Shiri, A. and Shumin, Z.: Touch Behavior with Different Posture on Soft Smartphone Keyboard, *MobileHCI'12 Proc. 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, pp.251–260 (2012).

[6] Rabiner, L.R. and Juang, B.H.: An introduction to hidden markov models, *IEEE Magazine on Accoustics, Speech and Signal Processing*, Vol.3, No.1, pp.4–16 (1986).

[7] Nils, K. and Michael, R.: Word n-grams for cluster keyboards, *TextEntry'03 Proc. 2003 EACL Workshop on Language Modeling for Text Entry Methods*, pp.51–58 (2003).

[8] Ahmet Cunyed Tantug: A Probabilistic Mobile Text Entry System for Agglutinative Languages, *IEEE Trans. Consumer Electronics*, Vol.56, No.2 (2010).

[9] Khaldoun, A.F., Mustapha, M. and Nadine, V.: BigKey: A Virtual

Keyboard for Mobile Devices, *Proc. 13th International Conference, HCI International 2009*, San Diego, CA, USA, Part III (2009).
[10] Henze, N., Rukzio, E. and Boll, S.: 100,000,000 taps: Analysis and improvement of touch performance in the large, *MobileHCI'11 Proc. 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, pp.133–142 (2011).
[11] Holz, C. and Baudisch, P.: The Generalized Perceived Input Point Model and How to Double Touch Accuracy by Extracting Fingerprints, *CHI'10 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.581–590 (2010).
[12] Himberg, J., Hakkila, J., Kangas, P., et al.: On-line Personalization of a Touch Screen Based Keyboard, *IUI'03 Proc. 8th International Conference on Intelligent User Interfaces*, pp.77–84 (2003).
[13] Leah, F. and Jacob, W.: Personalized input: Improving ten-finger touchscreen typing through automatic adaptation, *CHI'12 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.815–824 (2012).
[14] Goodman, J., Venolia, G., Steury, K., et al.: Language Modeling for Soft Keyboards, *IUI'02 Proc. 7th International Conference on Intelligent User Interfaces*, pp.194–195 (2002).
[15] Asela, G., Tim, P. and Christopher, M.: Usability Guided Key-Target Resizing for Soft Keyboards, *IUI'10 Proc. 15th International Conference on Intelligent user Interfaces*, pp.111–118 (2010).
[16] Goel, M., Jansen, A., Mandel, T., et al.: ContextType: Using hand posture information to improve mobile touch screen text entry, *CHI'13 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.2795–2798 (2013).
[17] Yin, Y., Ouyang, T.Y., Partridge, K., et al.: Making Touchscreen Keyboards Adaptive to Keys, Hand Postures, and Individuals - A Hierarchical Spatial Backoff Model Approach, *CHI'13 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.2775–2784 (2013).
[18] Hagiya, T. and Kato, T.: Probabilistic Keyboard Adaptable to User and Operating Style Based on Syllable HMMs, *21st International Conference on Pattern Recognition*, Tsukuba, Japan, pp.65–68 (2012).
[19] Goel, M., Findlater, L., Wobbrock, J.O., et al.: WalkType: Using accelerometer data to accomodate situational impairments in mobile touch screen text entry, *CHI'12 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.2687–2696 (2012).
[20] Hashimoto, M. and Togasi, T.: A virtual oval keyboard and a vector input method for pen-based character input, *CHI'95 Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp.254–255 (1995).
[21] Swype - Nuance, available from ⟨http://www.nuance.com/products/swype/index.htm⟩.
[22] Leggetter, C.J. and Woodland, P.C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models, *Computer Speech and Language*, Vol.9, pp.171–185 (1995).
[23] Kurohashi, S. and Kawahara, D.: Japanese Morphological Analysis System JUMAN 7.0 Users Manual. available from DOI: http://nlp.ist.i.kyoto-u.ac.jp (accessed 2013-02-08).
[24] Arif, A.S. and Stuerzlinger, W.: Analysis of Text Entry Performance Metrics, *Proc. Science and Technology for Humanity* (*TIC-STH 2009*), pp.100–105, IEEE (2009).
[25] Yamada, H.: A historical study of typewriters and typing methods: From the position of planning Japanese parallels, *The Journal of Information Processing*, Vol.2, pp.175–202 (1980).

Toshiyuki Hagiya received his B.E. and M.E. degrees from Kyoto University in 2008 and 2010, respectively. He is currently with KDDI R&D Laboratories, Inc. He has been engaged in research and development of spoken dialogue systems and interactive user interface. He is a member of IPSJ.

Tsuneo Kato received his B.E., M.E. degrees and Ph.D. from The University of Tokyo in 1994, 1996, and 2011, respectively. He joined Kokusai Denshin Denwa Co. Ltd. in 1996. He is currently with KDDI R&D Laboratories, Inc. He has been engaged in research and development of automatic speech recognition, spoken dialogue systems, and interactive user interface. He received IPSJ Kiyasu Special Industrial Achievement Award in 2011. He is a member of ASJ, IPSJ, IEEE, and IEICE.