Proceedings of the 24th IEEE International
Symposium on Robot and Human Interactive Communication
Kobe, Japan, Aug 31 - Sept 4, 2015

Tap Model that Considers Key Arrangement to Improve Input Accuracy of Touch Panels

Takahisa Tani
The Graduate University for Advanced Studies
2-1-2 Hitotsubashi, Chiyoda, Tokyo
101.8430, Japan
tani@nii.ac.jp

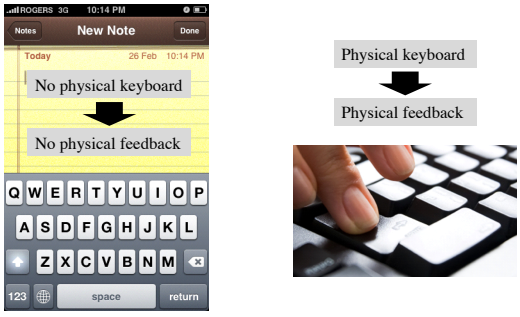
Seiji Yamada
National Institute for Informatics
The Graduate University for Advanced Studies
Tokyo Institute of Technology
2-1-2 Hitotsubashi, Chiyoda, Tokyo
101.8430, Japan
seiji@nii.ac.jp

***Abstract*— The use of mobile devices that utilize touch panels as interfaces, such as smartphones and tablet PCs, has spread in recent years, and these have many advantages. For example, panels can be operated more intuitively than those with conventional physical buttons, and the devices are much more flexible than those that use traditional fixed UIs. However, mistakes frequently occur when inputting with a touch panel because the buttons have no physical boundaries and users cannot get tactile feedback from their fingers. Thus, the input accuracy of touch-panel devices is lower than that of devices with physical buttons. There have been studies on improving input accuracy. Most of them have used language models for typing natural language or probabilistic models to describe the errors made when users tap panels with their fingers. However, these models are not practical because they deal with kinematic errors, not cognitive errors. Thus, we propose a more practical model for improving input accuracy in this paper, in which the tap model includes cognitive errors to avoid tapping neighboring objects to a target object. We consider that our model can describe important properties for designing various UIs depending on practical applications. We also conducted experiments to build our model in a calibrated way and discussed our evaluation of the model and revision of the model.**

I. Introduction

The use of mobile devices that use touch panels as interfaces, such as smartphones and tablet PCs, has spread in recent years, and these have many advantages. The panels can be operated more intuitively than conventional physical buttons, and the devices are much more flexible than those that use traditional fixed UIs, e.g. a forcetap-sensitive approach [6] and touch-based direct manipulation [10].

However, mistakes frequently occur when inputting with touch panels because the buttons have no physical boundaries and users cannot obtain tactile (physical) feedback because the panels never physically change when they are being tapped [9] [Figure 1(a)]. Thus, the input accuracy of touch-panel devices is significantly lower than those using conventional physical buttons [Figure 1(b)]. In addition, users often make unintentional mistakes when using the panels for input. In particular, mobile devices like smartphones usually have a smaller screen and smaller UIs than conventional large UIs on a computer display. Thus, the lack of high input accuracy is serious, and this problem with using small screens for input is called the *fat finger problem* [13] (Figure 2). This is a problem with accuracy in pointing manipulation. Input devices will progress in the future, and the pointing accuracy becomes more important. Thus, it is important to improve this accuracy for various practical applications of touch panels.



(a) Software keyboard. (b) Hardware keyboard.

Fig. 1. Software keyboard on smartphone and hardware keyboard.

There have been many studies on improving the accuracy of software keyboards [Figure 1(a)]. Software keyboards need to have a lot of keys to be placed in a small area to accommodate small screen sizes, especially on mobile devices like smartphones and tablet PCs. Hence, this is a typical example of the *fat finger problem* (Figure 2) because the keys are too small for users to correctly tap them. Most of these studies used two approaches. The first uses language models [4], [1], which have language information such as that from dictionaries. The system can predict the next character by using the pattern of input characters and dictionaries. For example, when the first part of a word is input, the system can predict the next character by matching the input part with words recorded in a dictionary. Although this approach is quite effective for key-typing like that when tapping a software keyboard, it cannot be applied to other input UIs, including tapping on simple buttons that are not

978-1-4673-6703-5/15/$31.00 ©2015 IEEE

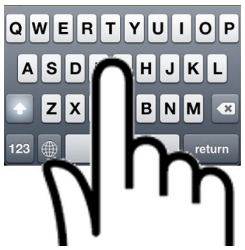


Fig. 2. Fat finger problem.

on a software keyboard. Thus, as the applicable coverage of these models has been significantly restricted to inputting natural language described in dictionary, we need to develop more general models to improve input accuracy for various concrete UIs.

The second approach uses tap models [2], [7] that have information on the difference between the locations of buttons and the points where users tap the keyboard. The system revises the location of points on the screen panel by using the tap models. There have also been studies that have combined both approaches [5], [12], [3].

A more general tap model [14] personalized for individual users was proposed. The model was defined on a continuous x-y coordination of a 2D tap screen and can be personalized by using Gaussian process regression as a quick regression algorithm with a small number of training examples. Also, the learned tap models are utilized for revising the actual tap point to the real target point. Experimental evaluations were done with participants. The main factor of this tap model was considered to be user kinematics because the errors significantly depended on the user's hand used for tapping. Thus, we call such errors *kinematics error* as mentioned later.

The first approach is concerned with reducing a significant gap between a software keyboard and a physical keyboard [9]. Although a physical keyboard key has three states (touched, pressed, and released), a software keyboard usually has only two (touched and released). Due to this difference, the state corresponding to the touched state of a physical key is missing in a touch screen keyboard. TapBoard is a touch screen software keyboard that regards tapping actions as keystrokes and other touches as the touched state. Thus, TapBoard can significantly reduce the gap between pressing keys on a physical keyboard and tapping a software keyboard.

There have been fundamental studies in which limited and concrete applications to practical UIs have not been assumed [8], [14]. Although these studies might provide novel knowledge in terms of general aspects, it is very difficult to use this knowledge to design UIs in practice. Thus, these models are not practical, and the experiments were in impractical environments. This means that the models may have been influenced by more complex factors such as the layouts or colors of the interface.

Thus, we focused on a method in this study that predicts touch points from multiple sensors that was proposed by Weir et al. [14] as mentioned before. Also, it is very important that we propose a more practical model called a *cognitive error model* that includes the influence of neighboring objects, e.g., buttons and links. We think that this model is necessary to actually design a UI on a tap screen because the tapping environment with other tapping targets is quite more practical than that without them.

II. Method of Estimating Tap Points with Multiple Sensors

The following method was proposed in the previous studies [14]. Let $\boldsymbol{s}$ be the input of multiple sensors and (x, y) be the intended location of a user. Here, multiple sensors mean the output of the touch panel, e.g., location, time, size of area, and pressure, and accelerometer. Then, the sensors calculate the function $(x, y) = f(\boldsymbol{s})$ by regression, and the system estimates the intended location from the sensor input by f.

Next, we extend this touch model to one with cognitive errors that is more practical by introducing the relationship between a target object and a neighboring object. Furthermore, we try to introduce incremental learning to improve the model through user execution of the UI.

A. Influence of Interface Layout on Tap Model

A tap location in practical use changes with various factors. It is particularly known that the tap model significantly changes with the differences in how a device is held and how the fingers operate it. We call this difference in tapping points *kinematic error* (e_k). This influence may be solved by estimating these factors with sensors like acceleration sensors.

However, the tap model may change with the interface layout. Let the blue squares be a target in Figure 3. The actual tap location has a distribution like the blue line (a), where a Gaussian distribution is assumed for the touch model. The actual tap distribution slightly shifts to the right because of the device is being grasped with the right hand. If there is another object (the green square) on the right, the distribution will move to the left because the user is aware of the green square and tries to avoid mistouching the green square instead of the blue square. We call this difference in tap points *cognitive error* (e_c).

In addition, the position, size, color, or shape of the object might have an influence.

No previous studies have considered this kind of influence because sensor inputs $\boldsymbol{s}$ do not include this information. Therefore, we propose adding an interface layout $\boldsymbol{l}$ as an input variable of f. We consider that this information will make a touch model much more practical. Since our touch model with an interface layout is basically characterized with (x, y) coordination on a touch panel, it can be applied to various UIs independent of the properties of tappable objects like buttons and icons. Thus, this model has a wide coverage that can be applied the same as conventional touch models. Furthermore, this proposed touch model is very practical

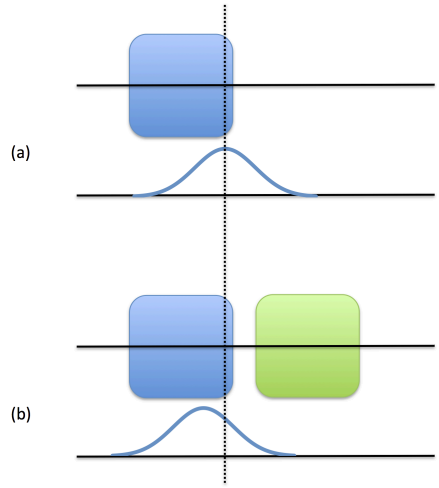


Fig. 3. Tap target and tap location.

and precise because it effectively introduces the influence of neighboring objects in contrast with conventional touch models [8], [14].

Thus, we can summarize the proposed tap model in a simple way like the following equation.

$$(x, y) = f(\boldsymbol{s}, \boldsymbol{l}) = e_k + e_c \tag{1}$$

Since we assume that e_c influences the tap model as well as e_k, we need to confirm the e_c through experiments with participants. In the next section, we conduct an experiment to confirm the e_c and to show that our extended tap model is more practical.

III. Experiment

We conducted an experiment to obtain a large number of training data for confirming e_c and evaluating the accuracy of our touch model.

A. Method

We developed a method of implicitly obtaining training data to evaluate the influence the interface layout had on the tap model. Participants were asked to perform a task in which they tapped a marker on a touch panel. Figure 5 outlines the task windows. The marker disappeared, and others appeared in other positions when the markers were tapped. No marker disappeared until tapped. The target marker that a participant should tap had a white circle in the center, and various neighboring markers appeared around it. We instructed participants to tap a target marker as quickly as possible.

The target marker was 3 $\times$ 3 mm in size. Neighboring markers appeared as shown in Figure 5(b) $\sim$ 5(d). The distance between markers was 1 mm. All parameters and sensed information of taps were recorded and considered. We randomly obtained a sufficient number of markers for our proposed model. The experimental environment with a participant is shown in Figure 4.

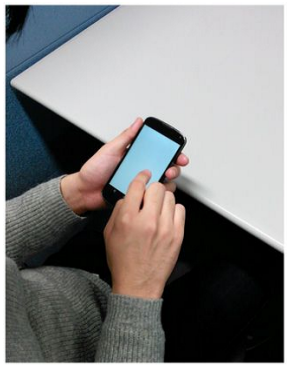


Fig. 4. Experimental environment.

B. Participants

We recruited 10 participants (five males, five females, ages 23 - 54). The group consisted of graduate students and staff members from the computer engineering department of our university and institute. Eight of the participants use smartphones every day, and the remaining two do not.

C. Evaluation

We investigated the two measures to confirm the practicality of e_c and to evaluate the effectiveness of our extended tap model consisting of e_k and e_c. We evaluated the difference in tap locations due to the existence of neighboring markers.

D. Results

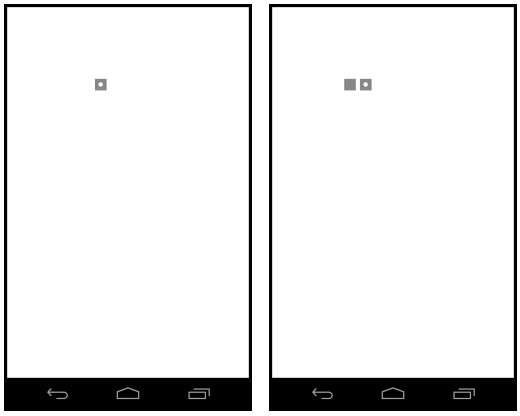
Figure 6 shows the kinematic error (e_k) for the horizontal and vertical axes. The curve was estimated from Gaussian process regression (GPR) [11]. Here, the squares mean the display. For (a) horizontal error, the color shows how much the error shifted to the left side (a unit is a pixel) for each point on the display. The maximum error (white areas) was about 3 mm, and the minimum error (green areas) was about 1 mm. For (b) vertical error, the color shows how much the error shifted to the upper side (a unit is a pixel) for each point on the display. The maximum error (white areas) was about 1 mm, and the minimum error (green areas) was about -1 mm.

Figure 7 and Figure 8 plot the cognitive error (e_c) for both axes and shows patterns having the biggest error. Figure 7 shows average of error for each pattern. Figure 8 shows the model using GPR. Here, we used data that only had one neighboring marker, and the effect of e_k was eliminated.

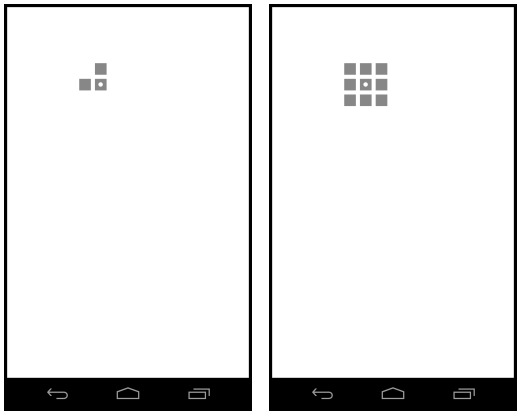
IV. Discussion

A. Tap Model

Figure 6 shows that the tap points shifted to the sides that were closest to the hands and the gaps in the points away

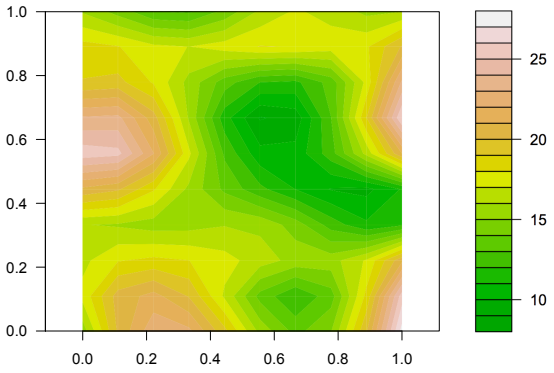


(a) Target marker only. (b) One neighboring marker.

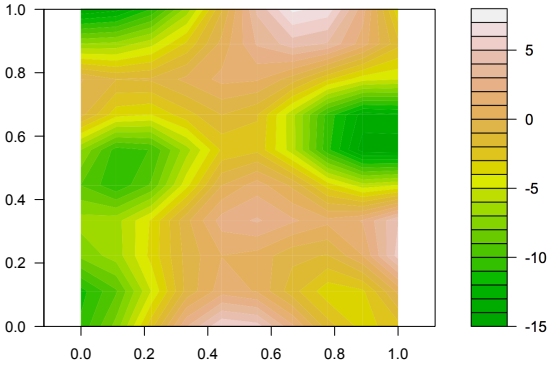


(c) Two neighboring markers. (d) Eight neighboring markers.

Fig. 5. Task windows.



(a) Horizontal.



(b) Vertical.

Fig. 6. Tapping points vs. kinematic errors for both axes.

from these sides were larger than the gaps between nearer points. These features have the same tendencies as those in previous work [8], [14].

Figure **??** shows that the tapping points shifted to the right when neighboring markers were on the left, shifted to the left when neighboring markers were on the right, shifted up when a neighboring marker was below, and shifted below when an additional marker was aboce. These shifts significantly supported the existence of e_c, and by using e_c, our extended model described in eq(1) can predict tapping errors more precisely than the conventional tap model without e_c.

We consider that we can synthesize various patterns with a single neighboring marker to predict errors in all types of cognitive errors. However, developing a concrete procedure for computing this is our future work.

B. Application to Real UIs

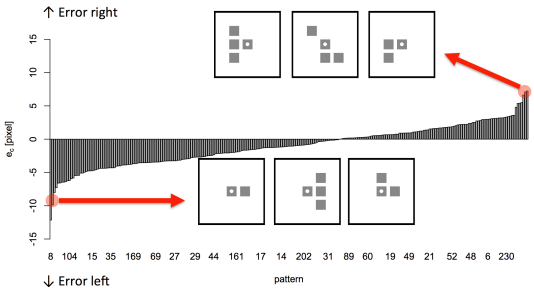
We should consider applying our method to real UIs. The system knows the tap point at execution and the current button layout. Therefore, the system should have data on errors in a button layout relative to a tap point. In other words, the system should estimate the point a user desired to tap from a tap point and the relative button layout. Here, we think that the influence of neighbor markers is very small when neighbor markers are far. Therefore, only the closest marker will be considered for now.

Figure 9 shows examples of real UIs and how to apply our method. Figure 9(a) shows pieces of anchor text on a screen. There are other buttons arranged horizontally above and below. These buttons resemble three markers in our experiment. Figure 9(b) shows buttons in a grid pattern. This pattern resemble eight markers in our experiment.

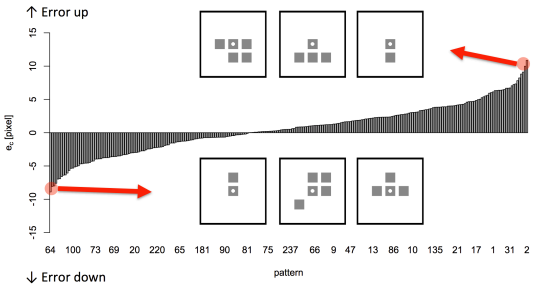
The patterns used in our experiment were very limited. There are many other parameters such as the space between objects, the size of objects, the color of objects, and the shapes of objects. We should consider these.

V. Conclusion

We considered that our tapping model could describe important properties in designing various UIs depending on practical applications in this study. We conducted an experiment to build it in a calibrated way and discussed its evaluation. Although the results suggested that cognitive error existed, we intend to do additional experiments and evaluate our model more thoroughly in future studies.

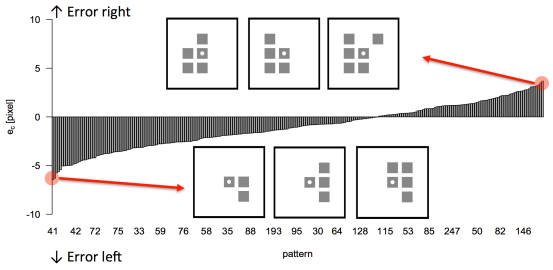


(a) Horizontal.

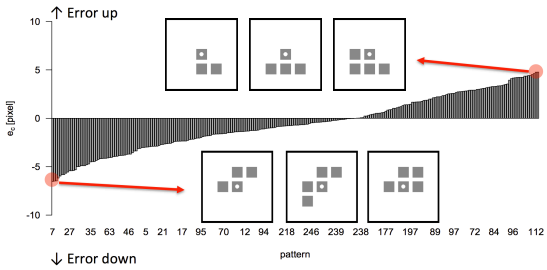


(b) Vertical.

Fig. 7. Averages of cognitive errors for each neighboring marker pattern for both axes.

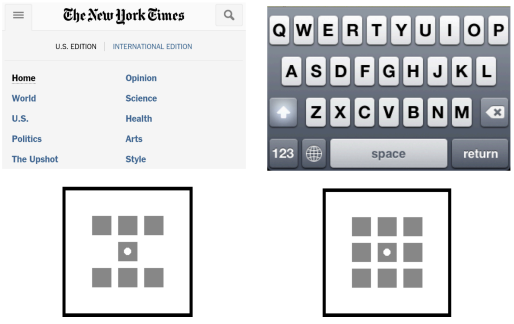


(a) Horizontal.



(b) Vertical.

Fig. 8. Model of cognitive errors for each neighboring marker pattern for both axes using GPR.



(a) Anchor text. (b) Software keyboard.

Fig. 9. Example of applying our method to real UIs.

REFERENCES

[1] S. Heo and G. Lee, "Forcetap: Extending the input vocabulary of mobile touch screens by adding tap gestures," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, ser. MobileHCI'11, 2011, pp. 113–122. [Online]. Available: http://doi.acm.org/10.1145/2037373.2037393

[2] C. Nguyen, Y. Niu, and F. Liu, "Direct manipulation video navigation on touch screens," in *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices and Services*, ser. MobileHCI'14, 2014, pp. 273–282. [Online]. Available: http://doi.acm.org/10.1145/2628363.2628365

[3] S. Kim, J. Son, G. Lee, H. Kim, and W. Lee, "Tapboard: Making a touch screen keyboard more touchable," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI'13, 2013, pp. 553–562. [Online]. Available: http://doi.acm.org/10.1145/2470654.2470733

[4] K. A. Siek, Y. Rogers, and K. H. Connelly, "Fat finger worries: how older and younger users physically interact with pdas," in *Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction*, ser. INTERACT'05, 2005, pp. 267–280. [Online]. Available: http://dx.doi.org/10.1007/11555261_24

[5] J. Goodman, G. Venolia, K. Steury, and C. Parker, "Language modeling for soft keyboards," in *Proceedings of the 7th international conference on Intelligent user interfaces*, ser. IUI'02, 2002, pp. 194–195. [Online]. Available: http://doi.acm.org/10.1145/502716.502753

[6] K. Al Faraj, M. Mojahid, and N. Vigouroux, "Bigkey: A virtual keyboard for mobile devices," in *Proceedings of the 13th International Conference on Human-Computer Interaction. Part III: Ubiquitous and Intelligent Interaction*, 2009, pp. 3–10.

[7] L. Findlater and J. Wobbrock, "Personalized input: improving ten-finger touchscreen typing through automatic adaptation," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI'12, 2012, pp. 815–824.

[8] J. Himberg, J. Häkkilä, P. Kangas, and J. Mäntyjärvi, "On-line personalization of a touch screen based keyboard," in *Proceedings of the 8th international conference on Intelligent user interfaces*, ser. IUI'03, 2003, pp. 77–84.

[9] A. Gunawardana, T. Paek, and C. Meek, "Usability guided key-target resizing for soft keyboards," in *Proceedings of the 15th international conference on Intelligent user interfaces*, ser. IUI'10, 2010, pp. 111–118.

[10] D. Rudchenko, T. Paek, and E. Badger, "Text text revolution: a game that improves text entry on mobile touchscreen keyboards," in *Proceedings of the 9th international conference on Pervasive computing*, ser. Pervasive'11, 2011, pp. 206–213.

[11] M. Goel, A. Jansen, T. Mandel, S. N. Patel, and J. O. Wobbrock, "Contexttype: using hand posture information to improve mobile touch screen text entry," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI'13, 2013, pp. 2795–2798.

[12] D. Weir, S. Rogers, R. Murray-Smith, and M. Löchtefeld, "A user-specific machine learning approach for improving touch accuracy on mobile devices," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ser. UIST'12, 2012, pp. 465–476.

[13] C. Holz and P. Baudisch, "Understanding touch," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI'11, 2011, pp. 2501–2510.

[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.