TapBoard: Making a Touch Screen Keyboard More Touchable

Sunjun Kim, Jeongmin Son, and Geehyuk Lee
Human-Computer Interaction Lab, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon, South Korea
{kuaa.net, jmin.sohn, geehyuk}@gmail.com

Hwan Kim, and Woohun Lee
Department of Industrial Design, KAIST
291 Daehak-ro, Yuseong-gu, Daejeon, South Korea
{hwan.kim, woohun.lee}@kaist.ac.kr

ABSTRACT
A physical keyboard key has three states, whereas a touch screen usually has only two. Due to this difference, the state corresponding to the touched state of a physical key is missing in a touch screen keyboard. This touched state is an important factor in the usability of a keyboard. In order to recover the role of a touched state in a touch screen, we propose the TapBoard, a touch screen software keyboard that regards tapping actions as keystrokes and other touches as the touched state. In a series of user studies, we validate the effectiveness of the TapBoard concept. First, we show that tapping to type is in fact compatible with the existing typing skill of most touch screen keyboard users. Second, users quickly adapt to the TapBoard and learn to rest their fingers in the touched state. Finally, we confirm by a controlled experiment that there is no difference in text-entry performance between the TapBoard and a traditional touch screen software keyboard. In addition to these experimental results, we demonstrate a few new interaction techniques that will be made possible by the TapBoard.

Author Keywords
TapBoard; text-entry method; touch screen keyboard.

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: User Interfaces - Input devices and strategies.

General Terms
Design; Human Factors.

INTRODUCTION
Tablet computers with multi-touch screens are increasingly popular. As more applications are being designed for tablets, the role of the touch screen software keyboard is becoming more important. Tablet computers with large touch screens, such as the Apple iPad and the Samsung Galaxy Tab, have software keyboards with a full-size, complete QWERTY layout, similar to that of a laptop. Because of the similarity between the software keyboard and the laptop keyboard in terms of both form and size, users tend to type on the touch screen software keyboard with all ten fingers and attempt to touch-type, as they do on a physical keyboard [2].

An intrinsic problem in the design of touch screen software keyboards is that touch screens usually have only two states (released and touched), whereas real physical keys have three states (released, touched, and pressed). Due to this mismatch in the number of states, touch screen keyboard designs have had to disregard the touched state of physical keys, simulating only the released and pressed states, as illustrated in Figure 1. As the result, touch screen software keyboards do not currently have a state that corresponds to the touched state of physical keys.

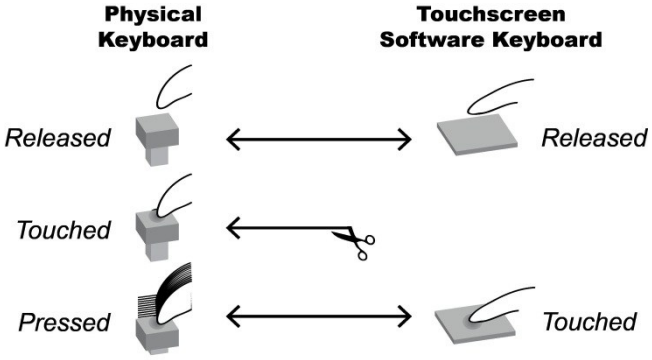


Figure 1. Mapping between the states of physical keys and the states of touch screens. Due to the mismatch in the number of states, the touched state of physical keys is not mapped to any state in touch screens.

The touched state actually plays an important role in the usability of physical keyboards [6, 10, 18, 23]. For instance, a keyboard user can rest their fingers in the touched state during typing. Users can also feel the texture of the keys, allowing the chance to align their fingers. An ultimate solution to restore the role of the touched state in a touch screen keyboard seemed to be the use of a three-state touch screen [4, 11], but the relevant technology is not yet ready for the market. As a currently feasible solution, we consider the alternative mapping illustrated in Figure 2, where the released and touched states of physical keys are mapped to the released and touched states of touch screens, respectively. In contrast to the case in Figure 1, the pressed state of physical keys is disregarded instead of the touched state. In fact, the role of the pressed state of physical keys is not as important as that of the touched state. The pressed

CHI 2013, April 27–May 2, 2013, Paris, France.

state is usually a transient state that is visited only briefly during a keystroke event. A keystroke event, which is not a state, is more important than the pressed state. Therefore, we decided to disregard the pressed state, and simulate a keystroke event with a tapping event. A consequence of this mapping is that there will be some restriction in utilizing the touched state, as a brief touch will be interpreted as a tapping action. If one wishes to rest their fingers on the touch screen keyboard, the duration should be longer than a certain threshold. Despite this small limitation, the touched state of a touch screen has now become available for other purposes. The result is the TapBoard, a touch screen software keyboard that can be operated by a tapping action. A tapping event, as will be defined precisely later, is defined not only by its short duration, but also by a small movement on the screen.

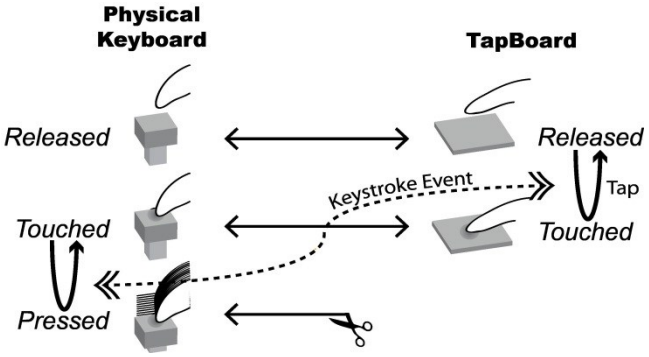


Figure 2. Mapping between the states of a physical key and the states of a touch screen in TapBoard. The pressed state of a physical key is disregarded, and the keystroke event of a physical key is now mapped to the tapping event of a touch screen.

We made a few assumptions when we first conceived the TapBoard. First, we could observe that people do use tapping when they type on a touch screen keyboard, and therefore we expected that TapBoard would be compatible with users' existing typing skills. Second, we expected that users will quickly be able to adapt to TapBoard and utilize the touched state, e.g., for resting their fingers on the screen. Third, we expected that TapBoard will be as efficient as an ordinary touch screen keyboard, because it is compatible with the existing typing skill of most users. All these assumptions require experimental support; hence, we conducted three user studies to collect experimental data. In the first, we verify the first assumption that underlies the TapBoard design, and observe user behavior on both physical and touch screen software keyboards. In the second study, we elicit several behavioral observations to support the second assumption and fine-tune the parameters of TapBoard. In the third study, we compare the text entry performance of a traditional touch screen software keyboard and TapBoard, and obtain experimental support for the third assumption.

In the following, we first summarize related works on touch screen text-entry methods and user typing behavior. We then present the detailed design of TapBoard, and describe three empirical studies. After the experimental results, we introduce some new possibilities enabled by the recovered touched state of TapBoard. Finally, we conclude the paper by summarizing the contributions of the current research.

RELATED WORK

Because typing skills transfer from the physical to software keyboards [20], we introduce a number of relevant studies on the characteristics of text entry on a physical keyboard and a touch screen software keyboard. In the following subsections, we discuss the importance of the touched state in physical keyboards and why it should be offered in large-size touch screen devices.

Text Entry Using Physical Keyboards

Expert touch-typists can type very quickly on physical keyboards without any visual attention. Several studies targeting such typists have pointed out that the removal of kinesthetic and tactile feedback significantly reduced the typing performance [2, 6]. In particular, Crump and Logan claimed that simply removing the keycaps, which takes away only tactual information and not kinesthetic movement, also had a negative influence on performance [6]. Other studies confirm that tactile information from the keys is the main factor in finger motion planning [6, 10, 18]. It seems clear that touch-typing and fast typing exploit the tactile cues provided by the touched state of physical keyboards.

Text Entry using Touch Screen Software Keyboards

As physical keyboards come in different sizes, touch screen software keyboards are implemented in various forms, from 3-inch devices to full-sized tabletops. Compared to physical keyboards, touch screen software keyboards are known to suffer from poor performance [2, 5, 9, 26, 32]. Interestingly, however, well-designed touch screen software keyboards offer comparable performance to that of physical keypads in small-size devices such as smartphones [13, 15]. Because both keyboards rely heavily on the user's intensive visual attention, the typing skill required for them is similar. In this case, the experience of physical keyboards can be easily transferred to software keyboards.

In contrast, typing performance on touch screen devices larger than a tablet is considerably worse than that on physical keyboards [5, 9, 24]. A number of studies have attempted to overcome the performance limitation using layout adaptation [8, 25], different layouts and sizes [17, 19, 26], and tactile feedback [22]. However, even in the best possible condition, where participants could ignore typing errors, results with touch screen keyboards were still 31% slower than with physical keyboards [9].

Possible Feedbacks for Touch Screen

As discussed in the previous subsection, the touched state plays a critical role in the use of physical keyboards. Therefore, we expect solutions providing the touched state in software keyboards [29, 30, 31] to be a strong candidate for improving their performance. TouchFire [30] and SLAP Widget [31] offer a tangible keyboard object on the touch screen. Only keystrokes are transferred to the touch screen by a mechanical structure, and other touches are blocked. Tactus technology [29] developed a deformable touch screen surface, which generates physical bumps dynamically. TeslaTouch [3] LATPaD [16], and STIMTAC [1] exhibit a dynamically changeable surface texture. The above techniques enable users to get information from the surface of the screen while sweeping their fingers across it.

TAPBOARD

TapBoard should accept only brief touches within the size of a key as valid keystrokes and ignore other touches. In order to do so, TapBoard runs a state machine, as shown in Figure 3, for each touch point. The state machine, which is initially in the [Idle] state, makes a transition to the [Touched] state on a [Touch(i)] event, where i is the index of the current touch point. On this transition, it resets a timer t and displacement d, creates a key-input object k, and outputs a [Preview(k)] event. The state machine returns to the [Idle] state when one of the following three events occur. (1) The timer t exceeds a timeout threshold τ. (2) The displacement d exceeds a displacement threshold δ. (3) A [Release(i)] event occurs. In the first two cases, a [Cancel(k)] event is output for the key-input object k. In the third case, a [Complete(k)] event is output for the key-input object k. Only a [Complete(k)] event is considered as a keystroke.

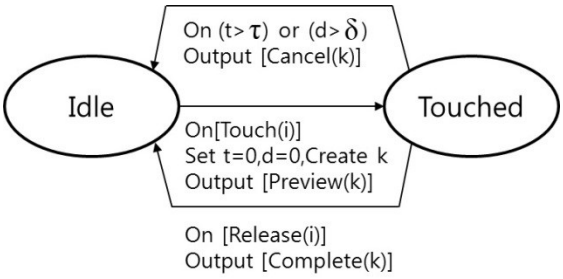


Figure 3. State machine of each touch point in TapBoard.

A TapBoard prototype was implemented in C# so that it can run on both the Samsung Slate 7 and Samsung SUR40. The prototype does not support all the keys of a standard full keyboard, offering only the alphabet keys, shift keys, a space bar, enter key, and backspace key, as shown in Figure 4. The TapBoard algorithm applies to all keys except the backspace and shift keys. These are excluded based on the results of experiment 1, which is described later.

As shown in Figure 4, the TapBoard layout replicates the dimensions of a physical keyboard. The timeout parameter τ was initially set to 300 ms, but was changed to 450 ms based on the result of experiment 2, which is described later. The distance threshold parameter δ was initially set to the width of an alphabet key, which is 80 pixels (about 15 mm). In addition to implementing the basic logic shown in Figure 3, we had to find some workarounds to handle the non-ideal behavior of the touch screen device, e.g., occasional spurious touch-and-release events, when five or more fingers are touching the screen. We handled at most five touch events simultaneously to prevent such erroneous behavior.

Figure 4. Screen snapshot showing the design of the prototype touch screen keyboard.

EXPERIMENTS

In a series of experiments, we validated the effectiveness of the TapBoard concept. First, we collected typing logs from physical keyboards and touch screen software keyboards to set the initial parameters for TapBoard. Second, we investigated whether users can adapt to the TapBoard concept. Finally, we studied whether there is a performance difference between using TapBoard and a conventional touch screen keyboard.

Experiment 1: Inspecting Typing Behavior

We conducted a within-subjects experiment with three different keyboard conditions: an instrumented physical keyboard with touch sensors (*physical*), a software keyboard typical of tablet PCs (*tablet*), and a software keyboard typical of tabletop PCs (*tabletop*). The experiment was carried out to measure the keystroke durations, i.e., key press durations on *physical* and touch durations on *tablet* and *tabletop*. We also observed the variation in typing behavior with conditions. For this experiment, 12 university students (six males and six females, mean age 22.7 years) were recruited. All of them were touch typists with traditional physical keyboards, but did not have much typing experience with software keyboards on tablet or tabletop computers.

Apparatus

We used a Samsung Slate 7 as the *tablet* and a Samsung SUR40 as the *tabletop*. The Samsung SUR40 uses an optical touch screen that is sensitive to hovering objects, and therefore, in the case of the Samsung SUR40,

participants had to wear black-painted cotton gloves that left only their fingers uncovered. These gloves prevented erroneous touch inputs from the palm of the hand.

In the *physical* keyboard, we implemented a conductive touch sensor using similar techniques to [12]. Participants wore a conductive pad connected to a function generator producing a 1 MHz sine wave. The touch sensor scanned the transferred signal through the human body. We multiplexed the touch sensor input with six 74HC4051 8-channel analog multiplexers, so the sensor unit was capable of sensing 48 touches individually. We attached copper tapes to the alphabet keys, shift keys, backspace key, space bar, and enter key, and wired them to the touch sensor unit to capture touch events (Figure 5).

Figure 5. Implementation of the *physical* condition, which is an instrumented physical keyboard with touch sensors.

In all conditions, we developed a transcription test program (Figure 4) in which participants must transcribe test phrases shown at the top of the screen. For the *tablet* and *tabletop* conditions, the software keyboard was shown below the textbox. Each key on the software keyboard changes color when touched. Software keyboards are programmed to record touch and release events with timestamps and corresponding characters. For the *physical* condition, only the given text and a transcription textbox were shown on the screen, and participants were asked to transcribe the words using the instrumented keyboard. Touch and release events from the touch sensor, as well as key press and release events from the keyboard itself, were recorded with timestamps and corresponding characters. The keyboard dimensions, such as key size and distance between keys, were controlled to be identical in all three conditions.

Procedure

Each participant was asked to transcribe 50 sentences that were selected randomly from the MacKenzie and Soukoreff phrase sets [21] for each condition. They were required to type as fast and as accurately as possible. Participants typed 10 sentences in a training session, and 50 sentences in a test session. The order of devices was fully counterbalanced across participants to avoid carryover effects. Each participant took 40 min to finish the entire typing session.

Result

We recorded all touches from *tablet* and *tabletop*, and all touches (*touch)* and key presses (*press*) from *physical*. For touch data in the *physical* condition, we excluded touch durations of less than 10 ms, which were considered to be due to device noise. A total of 24058 touches on *tablet*, 24262 touches on *tabletop*, and 21796 key presses and 39366 touches on *physical* were collected.

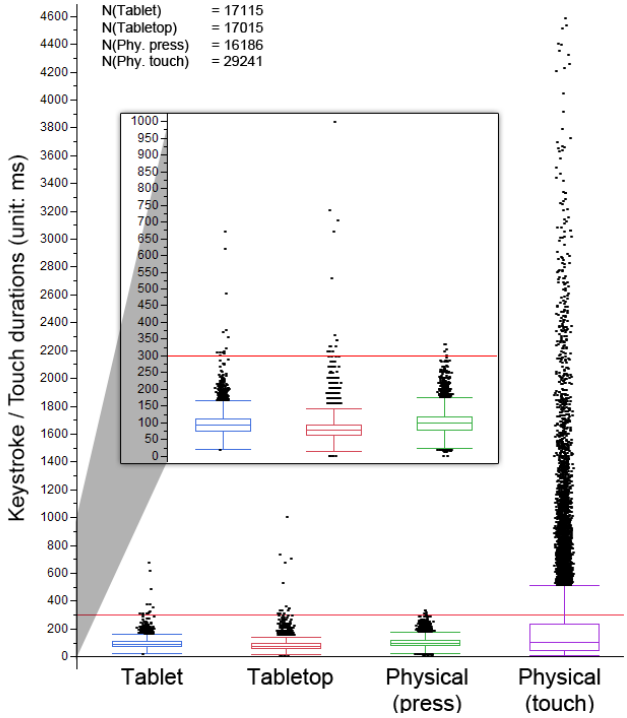


Figure 6. Box plots of touch duration and keystroke duration for alphabet keys. Touch durations of the touch screen devices and keystroke durations of the physical keyboard have similar distributions under 300 ms, while touch durations of the physical keyboard exhibit a significant tail over 300 ms.

We first analyze alphabet keystrokes. In total, 17162 touches on *tablet*, 17015 touches on *tabletop*, and 16186 key presses and 29241 touches on *physical* are analyzed. Figure 6 illustrates the distribution of keystrokes from each device. The 99.9 percentile values for *tablet*, *tabletop*, and *physical press* were 296 ms, 267 ms, and 285 ms, respectively. Most keystrokes took less than 300 ms. However, *physical touch* exhibits a significantly different distribution. More touches were counted than *physical presses* because fingers tend to touch surrounding keys during typing. Only 81% (23675 touches) of them fall within 300 ms. Short touches (<300 ms) are mainly induced by keystrokes. Long touches (>300 ms), which are of most interest, are mainly induced by resting or home-row searching.

Next, we analyze the other keys. The enter, space, and backspace keys exhibit similar distributions to the alphabet keys. For the shift keys, we observed a significant increase in touch duration and keystroke duration (Figure 7)

compare to those for the alphabet keys. Additionally, users activated the shift key longer on *tablet* and *tabletop* than on *physical*. Therefore, we cannot apply the TapBoard concept to the shift key. In addition, although it is not observed in this experiment, the backspace key often requires auto-repeat functionality. Thus, TapBoard is not suitable for the backspace and shift keys.

Finally, we inspected logs from the touch sensor and found some interesting patterns. (1) The number of simultaneously touched keys varied by participant. Figure 8 plots two extreme participants. P1 tends to touch and press exactly one intended key. In contrast, P12 often rested all fingers (nine touch counts). (2) Participants often realigned their fingers. Figure 9 shows a representative aligning pattern. (3) Resting touches were concentrated on home-row keys (ASDFGHJKL) and the space bar (70.8%), followed by upper-row keys (QWERTYUIOP, 21.8%). Fingers rarely rested on lower-row keys (ZXCVBNM, 1.4%). Figure 10 plots the distribution of resting touches.

In summary, most of the keystrokes are short (<300 ms), which supports our assumption. We also found that the touched state of physical keyboards exhibits totally different characteristics. Participants mainly utilize the touched state for aligning their fingers on the home row.

Experiment 2: Typing behavior with TapBoard

We conducted an observational study with two different touch screen devices: a tablet PC (*tablet*) and a tabletop PC (*tabletop*). We designed an experiment in which participants typed and waited during conversations. We expected TapBoard to lead users to rest their fingers on the touch screen keyboards and find the home row while waiting. For this experiment, five participants were recruited. All of them were university students (two males and three females, mean age 23.0 years). All of them were touch typists with traditional physical keyboards

Apparatus

This experiment again used the Samsung Slate 7 and Samsung SUR40. Subjects were provided with a simple chat program in both conditions. The program layout was identical to the transcription program of experiment 1, except that the software keyboard adopted the TapBoard algorithm with timeout threshold τ = 300 ms and distance threshold δ = 80 px. For a more natural typing experience, we allowed the participants to type in their mother tongue (Korean). The program recorded touch and release events with timestamps and corresponding characters. In addition, the software keyboard gave additional visual feedback when subjects put at least four fingers on the home-row.

Procedure

The participants were asked to chat with a moderator. The moderator and each participant talked about their daily life and interests. e.g., "what is your hobby?" or "please describe Mr./Miss Right." In addition, we gave the following instruction.

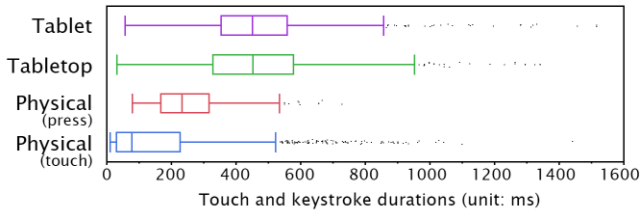


Figure 7. Box plots of touch durations and keystroke durations for the shift key.

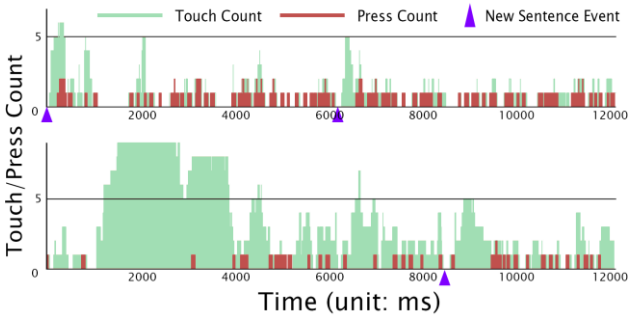


Figure 8. Typing behavior plot. Touch count represents number of touched keys, and press count represents number of pressed keys at a certain time. New sentence events take place when participants are given a new sentence to be transcribed.

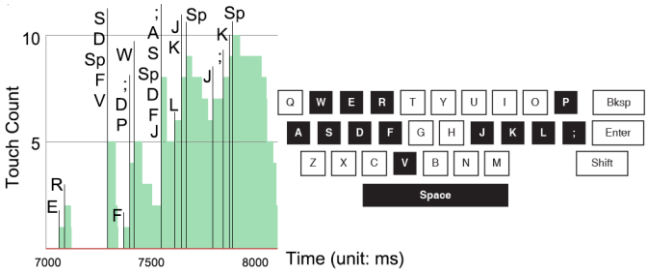


Figure 9. An aligning behavior pattern (P4).

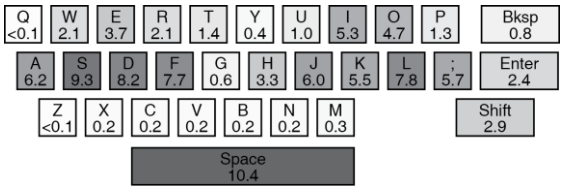


Figure 10. Distribution of resting touches. (unit: %)

"This chat session will be conducted through a special keyboard called TapBoard. Its appearance is not very different from that of a general keyboard, but it will cancel your input if you touch the surface for longer than 300 ms."

We conveyed this instruction just once before the start of each session in order that the participants understood the key feature of TapBoard naturally without any enforcement. After receiving this instruction, they tried cancelling their touches with one finger, two fingers, five fingers, and then eight fingers. Then, they conducted a 10 min chat session. Each participant had a session with both the *tablet* and *tabletop* conditions. Three participants started with *tablet*, and the other two started with *tabletop* in their first session.

Result

We observed resting behavior from four out of five participants. Figure 11 illustrates the representative resting behavior from one participant. Participants tended to rest their fingers (represented by *Canceled touch count*) while the moderator was typing questions (represented by *Moderator*). Similar behavior was observed for four participants across both devices. The exception was P2, who did not rest his fingers at all. P2 leant against the chair backrest during the whole of the chat sessions, and crossed his arms after he finished his replies. The debriefing with P2 revealed that he did not realize that TapBoard allowed him to rest his hands on the device.

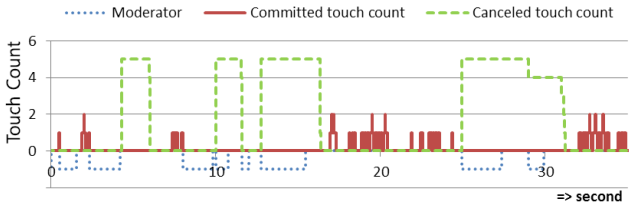


Figure 11. Resting behavior observed in participant P1.

Accumulated resting time percentages (canceled touches) are shown in Figure 12. With the exception of P2, participants actively rested for up to 29% of the total experiment time. We observed an interesting result related to the order of the conditions. P1 and P3 show a lower resting rate with the *tablet*, which was their starting condition. However, P4 and P5 show a higher resting rate with *tabletop*, which was their starting condition. We cautiously claim that *tabletop* induces more resting behavior due to its affordance.

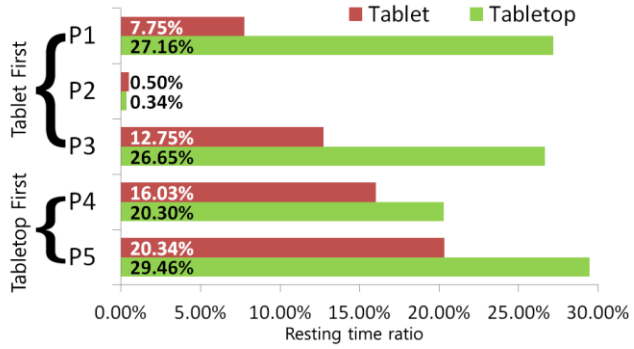


Figure 12. Resting time ratio within total experiment time for each participant and condition.

The timeout threshold τ is the dividing line between typing and resting, and there is a tradeoff. As we increase the threshold value, typing becomes easier, and as we decrease it, resting becomes easier. We collected canceled touches during typing and resting, and calculated the expected error rates along with different timeout thresholds from 300 ms to 1000 ms. We added two error rates and found the minimum point: 470 ms for *tablet* and 440 ms for *tabletop* (Figure 13). As a result, we conclude that 450 ms would be the balanced timeout threshold value, i.e., the tradeoff between typing and resting.

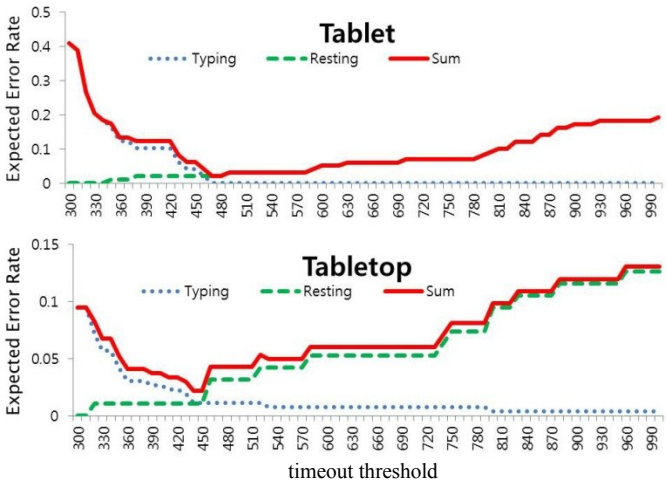


Figure 13. Expected error rates for different values of the timeout threshold parameter.

Experiment 3: Text Entry Performance

We conducted a comparative study with TapBoard keyboards (*TapBoard*) and traditional touch screen software keyboards (*Normal*) to measure text entry performance. The goal of this experiment was to show that TapBoard does not have an adverse effect on text entry performance. We expected participants to find the two conditions indistinguishable during the consecutive typing sessions. For this experiment, 10 university students were recruited (four males and six females, mean age 22.4 years). All of them were touch typists with traditional physical keyboards, but did not have a lot of experience with software keyboards on tablet computers.

Apparatus

In this experiment, transcription programs on the Samsung Slate 7 were given to subjects for both conditions. For the *TapBoard* condition, the timeout threshold τ was set to 450 ms based on the results of experiment 2.

Procedure

The participants were asked to perform transcription tasks with both *TapBoard* and *Normal*. They transcribed sentences randomly picked from the MacKenzie and Soukoreff phrase sets [21]. Each session consisted of continuous transcription for 20 min. Typing sessions alternated between *TapBoard* and *Normal*. Participants performed five sessions for each condition, thus giving a total of ten sessions per participant. The order in which the devices were used was counterbalanced. The ten sessions were spread over three consecutive days to prevent fatigue.

Result

We measured the typing speed and error rates according to metrics proposed by Soukoreff and MacKenzie [27]. Figure 14 shows the results for each condition. We then analyzed the results using two-way repeated measure ANOVA. *Session* and *Condition* are within-subject factors.

Figure 14. Performance measurements for *Normal* and *TapBoard* keyboards. Error bars represent the 95% confidence interval.

Typing Speed is measured in Words Per Minute (WPM), which is defined by Characters Per Second (CPS) * 60 / 5. The *Corrected Error Rate* (CER) is the frequency of errors committed but corrected during typing. *Not Corrected Error Rate* (NCER) is the frequency of errors left in the transcribed text. Finally, *Total Error Rate* (TER) is the unified error metric that combines these two error rates. Table 1 presents statistical test results for each performance metric.

	Condition ($F_{1,9}$ / p-value)	*Session* ($F_{4,36}$ / p-value)	Interaction ($F_{4,36}$ / p-value)
WPM	2.76 / .13	**10.50 / <.0001**	1.30 / .29
CER	1.52 / .25	**3.52 / <.05**	1.61 / .19
NCER	0.0003 / .998	**2.84 / <.05**	1.74 / .16
TER	1.36 / .27	**3.08 / <.05**	2.04 / .11

Table 1. Two-way repeated measure ANOVA results. *Condition* has two levels (*TapBoard* and *Normal),* and *Sesssion* has five levels (five sessions for each condition). Bold text represents significant main effect.

In summary, *Session* has a significant main effect, and *Condition* and *Condition*Session* interactions do not exhibit significant effects for all performance metrics. We conclude that *TapBoard* and *Normal* are not statistically different in their performance. To examine the equivalence of the performance of the two methods more rigorously, we picked two consecutive sessions and examined their equivalence using Two One-Sided t-Tests (TOST). For example, we compared session one and session two, session three and session four, and so on. Because we alternated the conditions, any two consecutive sessions consist of one *TapBoard* session and one *Normal* session. At the 95% confidence level, the test result indeed exhibited statistical equivalence for all session pairs.

Additional session

In the experiments thus far, the distance threshold δ was 80 px. A smaller δ value will enable better utilization of the touched state, but may increase the chance of canceling a legitimate keystroke. The value of 80 px worked well, but we hope to minimize δ for better utilization of the touched state, e.g., for better gesture operation. This, however, should be possible without degrading typing performance.

In order to estimate the touch displacement while typing on a touch screen, we conducted a pilot study with four graduate students (all male, ages from 23 to 28 years). They typed for 5 min in the *Normal* condition and we collected 5297 touch data. A maximum 50 px displacement between a touch and a release was observed (Max = 49.04 px, Mean = 7.94 px, SD = 6.07). Based on this result, we set δ = 50 px and conducted an experiment to see whether this reduction affected the typing performance. The experiment was a continuation of experiment 3. We conducted one additional session with *TapBoard* and *Normal* with the same participants (except P8, whom we could not contact). Then, we compared the new results with those of the last sessions of experiment 3. We analyzed the results with two-way repeated measure ANOVA. Within-subject factors are the two-level *Threshold* (δ = 80 px and δ = 50 px) and two-level *Condition* (*TapBoard* and *Normal)*. Table 2 presents the results.

	Condition ($F_{1,8}$ / p-value)	*Threshold* ($F_{1,8}$ / p-value)	Interaction ($F_{1,8}$ / p-value)
WPM	.59 / .49	.21, .66	.52 / .49
CER	.33, .58	.12 / .73	.58 / .47
NCER	.04 / .85	1.89 / .21	2.21 / .18
TER	.32 / .59	.21 / .66	1.09 / .33

Table 2. Two-way repeated measure ANOVA results for the additional session.

None of the factors exhibit significant effects for any performance metric. Thus, we conclude that TapBoard with δ = 50 px would be as effective as with δ = 80 px.

UTILIZATION OF TAPBOARD TOUCH STATE

As we have shown that TapBoard does not adversely affect typing performance on a touch screen, the next step is to show how the touched state of TapBoard may be utilized to enhance the touch screen keyboard experience.

Resting While Typing

As in the case of a physical keyboard, TapBoard users will be able to rest their fingers between typing operations. This possibility was in fact verified in the second experiment. Most of the participants in the experiment would rest their fingers on the touch screen while they waited for their turn to type in a conversation session. We expect that this feature of the TapBoard will be better appreciated when people are involved in a careful writing task, as they have to pause frequently between typing operations in order to find the best words or expressions for their work. We also expect that this feature will be more useful for a tabletop computer with a large touch surface, because a large and stable surface has the affordance to invite resting behavior [9]. This was the reason why we chose a tabletop computer

as well as a tablet computer in experiments 1 and 2. The results of experiment 2 actually support this expectation, i.e., participants showed a stronger tendency to rest on the keyboard in the case of a tabletop computer.

A closely related possibility is "anchored typing." By anchored typing, we mean a typing operation with one finger while the other fingers rest on the touch screen, as shown in Figure 15. This behavior is commonly observed when a user is repeatedly using special keys, such as a shortcut key or an arrow key. For instance, in order to read a long web page, one would use a page-down key repeatedly. Anchored typing is often a comfortable and stable typing technique for this kind of task. This scenario may sound somewhat outdated, as page turning is now usually done with a finger gesture on a touch screen. However, applications for which text-entry is a primary operation will need a software keyboard as a major input tool, and then the use of special keys will continue to be a viable interactive option.

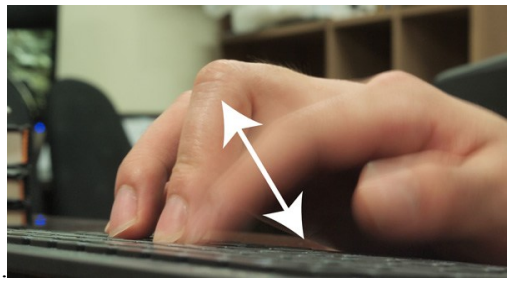


Figure 15. Anchored typing. One finger is repeating keystrokes while the other fingers are resting on the keyboard.

Tactile Feedback

The importance of tactile feedback in the use of a physical keyboard cannot be overemphasized. Users can feel the texture of the keys, and have a chance to align their fingers before starting to type. The touched state of TapBoard will enable users to feel the texture of the keyboard on a touch screen. This may sound meaningless, as there is no texture on a touch screen. This is true at present, but we anticipate that touch screens will have texture in the near future. There are already some early studies toward this goal, such as TeslaTouch [3], LATPaD [16] and STIMTAC [1]. One company is also presenting an early prototype of a programmatically deformable touch screen [29]. All of these tactile feedback technologies may only be meaningful when a user can touch and feel the surface.

Even before such an advanced tactile feedback technology becomes available, some researchers have begun studying the benefit of fine textures on a touch screen for typing performance. Kim and Lee [14] studied the effect of a thin, tactile overlay on a screen keyboard on typing performance. With a combination of a tactile overlay and clicking sound feedback, text entry performance and user preference were significantly improved. There is also a commercial product with a similar goal, known as TABLSKIN [28]. In order to study the benefit of being able to feel a touch screen keyboard, we constructed a tactile overlay using urethane film stickers (0.2 mm thick) on an ordinary OHP film, as shown in Figure 16a. In a pilot study, we could observe that participants were able to "blind-type" on the touch screen after approximately 10 min training. Another interesting possibility is typing with only a transparent template, i.e., without a graphical representation of a keyboard on the screen, as shown in Figure 16a. A clear advantage of this "texture-only" typing is that a software keyboard is not occluding an application window. In Figure 16b, for example, a user is typing on a web page using only the tactile overlay while reading the web page using the whole screen area.

Figure 16. (a) Transparent template, (b) template on contents.

Adaptive Keyboard

As TapBoard allows users to feel a touch screen more, it will also allow a touch screen to feel the users' touch more. A screen keyboard will be able to track the positions of hands and fingers when users rest their hands on the screen keyboard. A software keyboard may be instantiated under the hands when users rest their hands on a touch surface such as on a tabletop computer. The keys of a screen keyboard may adjust their positions to conform to the finger positions of individual users better. This concept, an adaptive screen keyboard, was in fact shown in [25]. In their design, the touch screen instantiates a left or right half of the keyboard when it detects four touch points of a hand. However, they confess to a usability problem due to the difficulty of distinguishing an intentional typing touch and an unintentional touch on the home row by a returning finger. This problem does not exist in the case of TapBoard, because only tapping is regarded as an intentional typing touch. The TapBoard will be a more effective basis for a robust realization of an adaptive screen keyboard. A basic implementation of an adaptive touch screen keyboard based on the TapBoard is shown in Figure 17. In the figure, a software keyboard is following the hands as a user aligns their fingers on the touch screen.

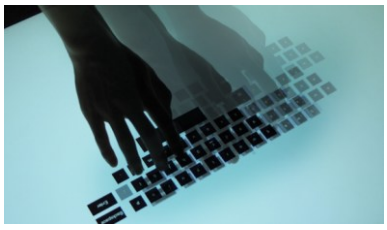


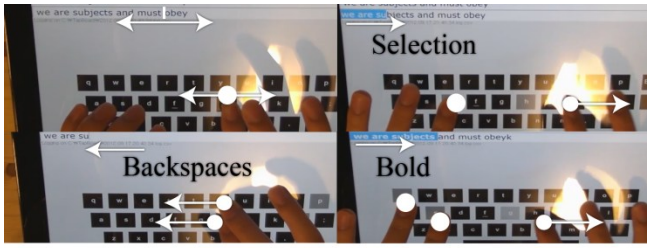
Figure 17. Adaptive touch screen keyboard implementation.

TapBoard Gestures

The most useful possibility enabled by TapBoard is that of using typing operations and gesture operations seamlessly. This is possible because only tapping is regarded as a typing operation, and all other movements on the screen can be utilized as gestures. For example, it will be possible to move the text cursor by a dragging action between typing operations without leaving the keyboard. In the following, we summarize some representative examples.

Text cursor control: It is often necessary to move a text cursor while typing in order to insert or delete a word. Instead of reaching for arrow keys or pointing with a thick finger in a textbox, a dragging gesture on the TapBoard may be used to move the text cursor (Figure 18a). As one drags further, the cursor will move further. Similarly, a dragging gesture with two fingers may be used for backspace operations (Figure 18b).

Text selection and formatting: Text selection may be done by combining the same cursor control gestures and a modifier gesture by the non-dominant hand. For instance, one may be able to select a portion of text by moving the text cursor with one of the fingers of the non-dominant hand anywhere on the keyboard (Figure 18c). This may sound like selecting words using arrow keys with a shift-key down, which is true, but the static gestures of the non-dominant hand can be more diverse. For instance, with two fingers, it may be possible to emphasize formatting (changing the font to italic or bold style) instead of text selection when the text cursor moves (Figure 18d).



(a) Cursor movement gesture (c) Selection gesture
(b) Backspace gesture (d) Emphasizing gesture

Figure 18. Text cursor control, selection, and formatting implementation.

GUI widget control: It is often necessary to mix typing and GUI widget control. In a form-filling task, e.g., on a web page, it is necessary to jump between different text fields. The gesture for text cursor control may be extended to deal with this problem. In fact, the design of TapBoard gestures to cover all of the primary operations will be a non-trivial design problem, and is not pursued in this paper. Instead, we implemented an example where one can mix typing operations and simple gestures to selectively accept the suggestion of a text box, such as that of Windows Explorer. A screen shot of an example implementation is shown in Figure 19. As users type an initial keyword, the text box shows a list of suggestions. They may select one of them using a vertical gesture and change the last word in the suggestion using a backspace gesture and typing. At this point, the text box will show an updated suggestion list. This cycle of interaction between users and the search box continues until they are satisfied with the current input. This is just an example of a new interaction style that may become possible when it is possible to conveniently switch between typing and GUI control.

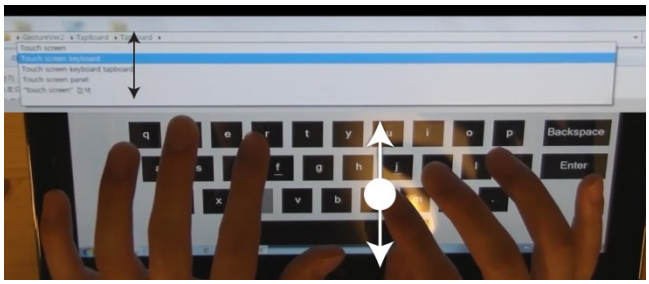


Figure 19. GUI widget control implementation.

Writing while typing: A keyboard may not provide all of the symbols that a user wants to input. In this case, writing (drawing) a symbol may be a more effective alternative. A writing operation may be performed with the dominant hand while all of the non-dominant hand fingers are down (a modifier gesture). Writing input may be translated into a symbol by a gesture recognizer, as in [7], or may be used as it is, i.e., as a handwritten symbol. We did not implement an example for this scenario, as a similar scenario has already been extensively discussed by Findlater et al. [7].

CONCLUSION

We proposed the concept of TapBoard, and verified its feasibility in a series of experiments. First, we showed that TapBoard is compatible with the existing typing skill of users. Second, we showed that users can adapt to TapBoard easily and utilize the touched state, e.g., for resting their fingers. Third, we showed that TapBoard is as efficient as an ordinary touch screen keyboard. After these experimental verifications, we demonstrated new interaction techniques that will be made possible by TapBoard. We expect TapBoard to enhance the touch screen interaction experience significantly, especially by enabling seamless integration of typing operations and GUI operations. An immediate next step is to extend the concept of TapBoard beyond a keyboard and make the whole touch screen more "touchable."

ACKNOWLEDGEMENT

This work was supported by the IT R&D program of MKE/KEIT. [KI10041244, SmartTV 2.0 Software Platform]

REFERENCES

1. Amberg, M., Giraud, F., Semail, B., Olivo, P., Casiez, G., and Roussel, N. Stimtac: a tactile input device with programmable friction. In *Proc. UIST '11 Adjunct*, ACM (2011), 7–8.
2. Barrett, J., and Krueger, H. Performance effects of reduced proprioceptive feedback on touch typists and casual users in a typing task. *Behaviour & Information Technology 13*, 6 (1994), 373–381.
3. Bau, O., Poupyrev, I., Israr, A., and Harrison, C. Teslatouch: electrovibration for touch surfaces. In *Proc. UIST '10*, ACM (2010), 283–292.
4. Buxton, W. A three-state model of graphical input. In *INTERACT*, vol. 90, Citeseer (1990), 449–456.
5. Chaparro, B., Nguyen, B., Phan, M., Smith, A., and Teves, J. Keyboard performance: ipad versus netbook. *Usability News 12*, 2 (2010).
6. Crump, M., and Logan, G. Warning: This keyboard will deconstruct - the role of the keyboard in skilled typewriting. *Psychonomic Bulletin & Review 17* (2010), 394–399.
7. Findlater, L., Lee, B., and Wobbrock, J. Beyond qwerty: augmenting touch screen keyboards with multi-touch gestures for non-alphanumeric input. In *Proc. CHI '12*, ACM (2012), 2679–2682.
8. Findlater, L., and Wobbrock, J. Personalized input: Improving ten-finger touchscreen typing through automatic adaptation. In *Proc. CHI '12*, ACM (2012), 2453–2462.
9. Findlater, L., Wobbrock, J. O., and Wigdor, D. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. In *Proc. CHI '11*, ACM (2011), 2453–2462.
10. Gordon, A., and Soechting, J. Use of tactile afferent information in sequential finger movements. *Experimental brain research 107*, 2 (1995), 281–292.
11. Heo, S., and Lee, G. Forcetap: extending the input vocabulary of mobile touch screens by adding tap gestures. In *Proc. MobileHCI '11*, ACM (2011), 113–122.
12. Hinckley, K., and Sinclair, M. Touch-sensing input devices. In *Proc. CHI '99*, ACM (1999), 223–230.
13. Hoggan, E., Brewster, S., and Johnston, J. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proc. CHI '08*, ACM (2008), 1573–1582.
14. Kim, S., and Lee, G. Typing on a touch surface: Effect of feedback with horizontal touch keyboard and vertical display setup. In *Proc. APCHI '12*, ACM (2012), 525–530.
15. Lee, S., and Zhai, S. The performance of touch screen soft buttons. In *Proc. CHI '09*, ACM (2009), 309–318.
16. Levesque, V., Oram, L., MacLean, K., Cockburn, A., Marchuk, N. D., Johnson, D., Colgate, J. E., and Peshkin, M. A. Enhancing physicality in touch interaction with programmable friction. In *Proc. CHI '11*, ACM (2011), 2481–2490.
17. Li, F., Guy, R., Yatani, K., and Truong, K. The 1line keyboard: a qwerty layout in a single line. In *Proc. UIST '11*, ACM (2011), 461–470.
18. Logan, G., and Crump, M. Hierarchical control of cognitive processes: The case for skilled typewriting. *Psychology of Learning and Motivation-Advances in Research and Theory 54* (2011).
19. MacKenzie, I., and Zhang, S. The design and evaluation of a high-performance soft keyboard. In *Proc. CHI '99*, ACM (1999), 25–31.
20. MacKenzie, I., Zhang, S., and Soukoreff, R. Text entry using soft keyboards. *Behaviour & information technology 18*, 4 (1999), 235–244.
21. MacKenzie, I. S., and Soukoreff, R. W. Phrase sets for evaluating text entry techniques. In *Ext. Abstracts CHI '03*, ACM (2003), 754–755.
22. McAdam, C., and Brewster, S. Distal tactile feedback for text entry on tabletop computers. In *Proc. BCS HCI '09*, British Computer Society (2009), 504–511.
23. Rekimoto, J., Ishizawa, T., Schwesig, C., and Oba, H. Presense: interaction techniques for finger sensing input devices. In *Proc. UIST '03*, ACM (2003), 203–212.
24. Ryall, K., Morris, M., Everitt, K., Forlines, C., and Shen, C. Experiences with and observations of direct-touch tabletops. In *Proc. Tabletop '06*, IEEE (2006).
25. Sax, C., Lau, H., and Lawrence, E. Liquidkeyboard: An ergonomic, adaptive qwerty keyboard for touchscreens and surfaces. In *Proc. ICDS '11* (2011), 117–122.
26. Sears, A. Improving touchscreen keyboards: design issues and a comparison with other devices. *Interacting with Computers 3*, 3 (1991), 253 – 269.
27. Soukoreff, R. W., and MacKenzie, I. S. Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *Proc. CHI '03*, ACM (2003), 113–120.
28. TABLSKIN. http://tablskin.com/.
29. Technology, T. http://www.tactustechnology.com/.
30. TouchFire. The screen-top keyboard for ipad. http://www.kickstarter.com/projects/740785012/touchfire-the-screen-top-keyboard-for-ipad.
31. Weiss, M., Jennings, R., Wagner, J., Khoshabeh, R., Hollan, J., and Borchers, J. Slap: Silicone illuminated active peripherals. *Ext. Abstracts of Tabletop 8* (2008).
32. Wigdor, D., Perm, G., Ryall, K., Esenther, A., and Shen, C. Living with a tabletop: Analysis and observations of long term office use of a multi-touch table. In *Proc. Tabletop '07*, IEEE (2007), 60–67.