VIPBoard: Improving Screen-Reader Keyboard for Visually Impaired People with Character-Level Auto Correction

Weinan Shi[12], **Chun Yu**[123†], **Shuyi Fan**[1], **Feng Wang**[13], **Tong Wang**[1], **Xin Yi**[12], **Xiaojun Bi**[4], **Yuanchun Shi**[123]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Key Laboratory of Pervasive Computing, Ministry of Education, China
[3]Global Innovation eXchange Institute, Tsinghua University, Beijing, China
[4]Department of Computer Science, Stony Brook University, Stony Brook, NY, United States
{swn18, fansy15, wangfeng18, wangtong15}@mails.tsinghua.edu.cn, {chunyu, yixin, shiyc}@tsinghua.edu.cn, xjunbi@gmail.com

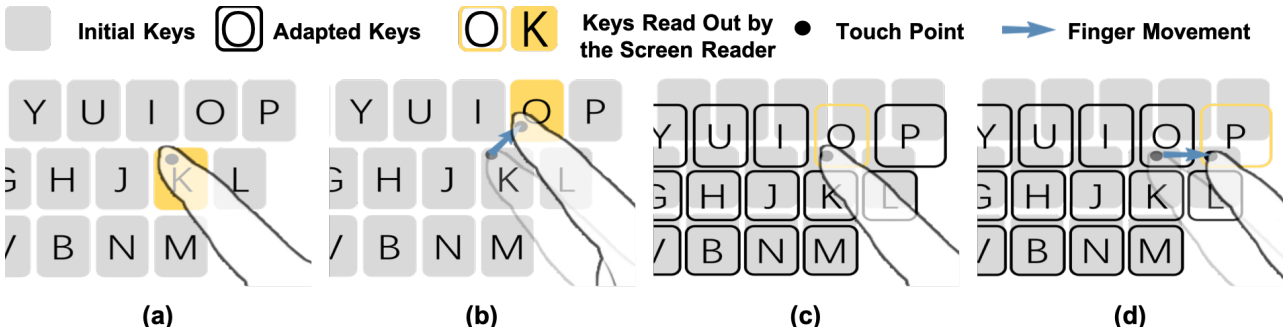


Figure 1: Illustration of inputting "o" after "hell" already entered. (a)(b) On a traditional screen-reader keyboard, the user often touches at a wrong location and performs a calibration to the intended key, which is time-consuming. (c) VIPBoard predicts the intended character "o" and adapts the keyboard layout, which removes the calibration phase. (d) Even if the prediction is wrong, the user can still input other keys (e.g., "P") by moving the finger on the new layout.

ABSTRACT

Modern touchscreen keyboards are all powered by the word-level auto-correction ability to handle input errors. Unfortunately, visually impaired users are deprived of such benefit because a screen-reader keyboard offers only character-level input and provides no correction ability. In this paper, we present VIPBoard, a smart keyboard for visually impaired people, which aims at improving the underlying keyboard algorithm without altering the current input interaction. Upon each tap, VIPBoard predicts the probability of each key considering both touch location and language model, and reads the most likely key, which saves the calibration time when the touchdown point misses the target key. Meanwhile, the keyboard layout automatically scales according to users' touch point location, which enables them to select other keys easily. A user study shows that compared with the current keyboard technique, VIPBoard can reduce touch error rate by 63.0% and increase text entry speed by 12.6%.

CCS CONCEPTS

• **Human-centered computing** → **Text input**; *Accessibility technologies*;

KEYWORDS

Visually Impaired; Text Entry; Smartphone; Auto-correction

ACM Reference Format:
Weinan Shi, Chun Yu, Shuyi Fan, Feng Wang, Tong Wang, Xin Yi, Xiaojun Bi, Yuanchun Shi. 2019. VIPBoard: Improving Screen-Reader Keyboard for Visually Impaired People with Character-Level Auto Correction. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3290605.3300747

† denotes the corresponding author.

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300747

1 INTRODUCTION

In the modern mobile computing era, smartphones are as indispensable to blind or visually impaired (BVI) people as they are to sighted people. However, there exist a number of interaction obstacles preventing BVI users from fully enjoying the benefits of mobile computing. One of them is entering text.

Smartphone text entry is notoriously challenging, even for sighted users: it is difficult to precisely select a small intended key with input finger. Fortunately, almost all the modern touchscreen keyboards for sighted users are so-called *smart* keyboards, which are enhanced with the word-level auto-correction ability. After a user types a word delimiter (e.g., space), the keyboard will correct the input literal string into a word in the dictionary, based on the language context and spatial touch locations.

Unfortunately, BVI users are deprived from such benefits and entering text remains extremely challenging. The main reason is that the existing word-level auto-correction keyboard does not suit the typing behavior of BVI users. Unlike sighted users who can ignore the intermediate letter errors and let the auto-correction correct them, with a screen reader (e.g., TalkBack on Android and VoiceOver on iOS) a BVI user enters a word letter by letter and will not proceed to the next letter until the current one is correctly entered and confirmed. To enter a specific letter, the user first locates the intended key by dragging finger on the keyboard, and then lifts the finger up (or double taps) to confirm the letter once it is reached. They need to assure that each letter is correctly entered because 1) this could avoid the high cost of correcting errors afterwards [9]; 2) audio feedback is more salient and noticeable than visual feedback if the input is wrong, which prevents BVI users from ignoring them. As a result, the screen-reader keyboard is not equipped with word-level auto-correction power and BVI users suffer from low text entry speed (less than 5 WPM [11, 32]).

We present VIPBoard, a *smart* screen-reader keyboard that brings auto-correction ability to BVI users without altering their input behavior. VIPBoard features in two mechanisms. First, it predicts the most probable intended character based on a language model and finger location, and automatically offsets the keyboard to make the predicted letter beneath the finger. Thus, users do not need to move their finger to correct the input. This saves the time and effort. Second, the keyboard layout is re-formed to guarantee that all keys can be accessed by moving the finger in case the predicted character is incorrect and the keyboard is offset inappropriately. This provides a consistent user experience as typing on a traditional non-smart keyboard, which minimizes the learning cost. The benefit of VIPBoard builds on the fact that for most of the times, the prediction is correct.

To evaluate the performance of VIPBoard, we conducted a user study with 14 visually impaired users. We compared VIPBoard with a traditional screen-reader keyboard (e.g., by TalkBack and VoiceOver). Results showed that VIPBoard reduced the touch error rate by 63.0% and increased the text entry speed by 12.6%. Users also showed strong preference of VIPBoard over the traditional keyboard. During the study, we did not inform the participants that different keyboards were used. In such a case, most users (12 out of 14) could not perceive the difference of use between VIPBoard and a traditional screen-reader keyboard. They mistakenly attributed the perceived difference of performance to the varying of their absolute touch ability. This result suggested that VIPBoard was consistent with their familiar typing method and could be mastered with little learning.

In the remainder of this paper, we first review related works. Then we give an introduction to the design of VIPBoard. After deriving a general touch model from a pilot study, we test and analyze the performance of VIPBoard and the traditional screen-reader keyboard through a user study. We conclude the paper by discussing the limitations and future work.

2 RELATED WORK

We will first introduce touch-based input methods used in modern smart keyboards which are mostly designed for sighted users, discuss whether these methods can be used for BVI users, then existing text entry methods designed for BVI users.

Word-level Smart Input Methods

Auto-correction methods have been widely used in modern word-level smart input methods [38, 42] (or sentence-level method [37]). These methods usually decode a user's input after he/she finishes all the touches by a delimiter (e.g., space), and provide a list of candidates with the highest probabilities according to the user's input. Goodman et al. [21] were the first to combine a touch model and a language model to reduce effect of users' input noise. This method has already been extended to numerous contexts, such as different hand postures [19], mobility [18] and screen size [42]. However, to our knowledge, it has not been used in text entry methods for BVI users. Kristenson and Zhai [25] presented an alternative pattern matching method to identify the most likely words corresponding to the user's tap sequences.

Though mostly used by sighted users, some of the works adopting word-level auto-correction methods [26, 35, 44] were also designed or could be used under eyes-free scenario by leveraging muscle memory. However, they were not suitable for BVI users because BVI users will confirm each letter is correctly entered during typing due to the high cost of

correcting errors and the audio feedback. These reasons inspired us to apply a character-level auto-correction method into a soft keyboard for BVI users.

Character-level Smart Input Methods

Character-level smart input methods benefit users by performing prediction each time a character is entered. These techniques usually change the keyboard layouts by each touch according to the predicted probabilities (visibly or invisibly): Fluctuating Optimal Character Layout [10] rearranges the layout so that the most likely next characters are closer to the cursor to minimize KSPC — the average number of keystrokes per character. The Fisheye keyboard [33] and BigKey [5] expand the next keys for a better target acquisition. Key-target resizing [8, 22] algorithms change the underlying target area of each key according to the calculated probabilities.

Explicitly changing keyboard may affect sighted users: they need cost to familiarize a new layout [10], or they may change their behavior according to the layout [5, 33]. Gunawarana et al. [22] emphasized the negative effect of overly aggressive keyboard adaption and proposed an anchored method to solve the problem. However, the problem does not exist for BVI users because they cannot directly "see" the changes. Therefore, layout adaption is possible on soft keyboards for BVI users. Besides, different from those existing prediction methods that benefit next touches after one touch is complete, it is also worthwhile to explore the character-level auto-correction method which corrects input errors within the current touch.

Text Entry for Visually Impaired Users

A traditional way of BVI users using soft keyboards is to use it under screen reader system, which is low efficient (0.66 WPM in [11], 1.32 WPM in [32]). The main challenge for BVI users to enter text using soft keyboard on smartphones is the difficulty of precise targeting [11]. Therefore, researchers have developed different techniques to address this problem.

One direct way of simplifying the targeting problem is to keep the number of targets low, place them at easy-to-reference locations based on the physical device, and keep targets static. Sánchez et al. [34] put 9 numpad-like virtual buttons on the screen. Inputting one character required multiple times of tapping. BrailleType, by Oliveira et al. [31], divided the screen into 6 large buttons and allowed users to enter the text by Braille encoding. These methods could provide more accurate results yet the efficiency might be low because of the low target number.

Gesture-based approaches are alternative solutions as they do not require the user to hit any targets. These methods usually allow users to navigate through all the characters with different layouts. For example, the navigation layout could be 3 different pie-menus [40], alphabet separated by vowels [20] or 8 group of keys in a radial arrangement [11]. Braille encoding texts could also be entered by simple gestures [6, 14, 17, 28, 29]. The main cost of these methods was the cost of learning gestures and the long time used to navigate through the targets.

Besides, hardware-based methods were also developed to assist targeting for BVI users, e.g., by using external devices [15, 24] or haptic feedback [12]. The requirement for special devices made these methods neither pervasive nor convenient for daily use.

To our knowledge, these methods were all aimed at improving the absolute ability to acquire every single character with new, different interfaces. However, optimization based on traditional keyboards were not considered, nor modern smart correction methods. VIPBoard fills the gap by augmenting a screen-reader keyboard with character-level auto-correction methods to reduce the effect of input noise produced by BVI users.

3 DESIGN OF VIPBOARD

In this section, we will first introduce how BVI users use VIPBoard. Then the two main parts of VIPBoard: a Bayesian algorithm to predict the user's intention and a layout adaption strategy. The prediction algorithm predicts a user's intention based on his/her input history and a pre-defined corpus. The adaption strategy will adjust the keyboard layout according to the prediction result to provide consistent interactions for BVI users. We also present some implementation details in the adaption process, which are important for the strategy to provide a better experience for BVI users.

Interaction

The interaction of VIPBoard is based on traditional screen-reader keyboard, which aims at minimizing the learning effort of users. A screen reader provides audio feedback to a BVI user by reading out the UI element's name touched by the user. The user confirms the selection by lifting up his/her finger up[1].

The process of inputting one character on a traditional screen-reader keyboard is consistent with the screen reader, which can be divided into three phases:

(1) **Attempt** (touchdown). The user puts his/her finger on the keyboard, and the screen reader reads out the key name where the touch point locates (Figure 1a).
(2) **Calibration** (touchmove, optional). If the first key read out by the screen reader is the intended one, this step is skipped. Otherwise, the user needs to move his/her finger around to find the target key according

[1]Alternative methods include by performing a double-tap or by tapping with another finger without lifting the finger on the screen.

to the keyboard layout and audio feedback (Figure 1b). The audio feedback is provided each time the touch point enters a new key boundary.

(3) **Confirmation** (touchup). The user lifts his/her finger up from the screen to input the current character.

VIPBoard estimates the probability of characters that might be needed by the user each time after a touchdown event is observed on the screen. The system will then read out the predicted letter with the highest probability. By this way, the user does not need to perform further touch calibration and thus the typing speed is increased (Figure 1c). To achieve consistent interactions with the traditional keyboard, the user should still be able to calibrate in case of a wrong prediction. VIPBoard achieves this by adapting the keyboard layout according to the touch location and predicted result, by which the relative layout of the keyboard still holds (Figure 1d). The confirm operation is the same as the traditional one.

Input Prediction Algorithm

We use a method similar to [21] to predict the most probable key of the input of a user. Let pos denotes the input touch position, pre denotes the history which has been already input by the user. Then for any character c, we can calculate the probability of a user inputting c given pos and pre by:

$$\begin{aligned} P(c \mid pos,\ pre) &= \frac{P(pos,\ c,\ pre)}{P(pos,\ pre)} \\ &= \frac{P(pos \mid c,\ pre)P(c,\ pre)}{P(pos,\ pre)} \\ &\propto P(pos \mid c,\ pre)P(c,\ pre) \end{aligned} \tag{1}$$

In equation 1, $P(c,\ pre)$, which can be referred to as language model, is calculated by a pre-defined corpus, which is similar with [43]:

$$P(c, pre) = \frac{\sum_{W \in S(pre) \wedge W_{n+1}=c} P(W)}{\sum_{W \in S(pre)} P(W)} \tag{2}$$

where n is the length of pre and $S(pre)$ denotes all the words W that $P(pre|W_1W_2 \cdots W_n) > 0$. In other words, $S(pre)$ denotes all the words have a prefix pre. $P(pos|c, pre)$, which can be referred to as touch model, reflects the input noise of the user. Like most previous work [7, 21], we treat each touch independently, so it can be simplified as $P(pos|c)$ and calculated by a bivariate Gaussian distribution. After calculating $P(c|pos, pre)$ for all the characters, we can find the one with the maximal probability as the intention of the user.

Layout Adaption Strategy

The main purpose for layout adaption is to hold a consistent interaction and layout with the traditional screen-reader keyboard. A particular advantage for this step is that users can still perform calibration in case of a wrong prediction, thus can input any character on the keyboard. The layout adaption strategy was adopted after touchdown event triggers and a most intended key was calculated by the previous algorithm. The layout restores to the initial one after a character has been confirmed (i.e. a touchup event triggers). This design is to avoid the overly aggressive adaption of the layout [22], which may result in some ultra small keys that are hard to be input by the user. The main steps can be described as follows:

Step 1: Calculate an offset vector between touch position and the most intended key in the initial layout. The offset vector is the minimal path from the key boundary to the touch position.

Step 2: Translate the most intended key to a new position according to the offset vector, after which the touch position is contained by the new key boundary.

Step 3: Translate and scale other keys relatively to the most intended key and make sure all the keys are in the keyboard boundary.

Step 4: If there exist keys which are too small [2], retain the initial layout and read out the key corresponding to the touch position. Otherwise, adopt the new layout and read out the most intended key.

Figure 2a shows an example of an adapted new layout as well as the initial layout.

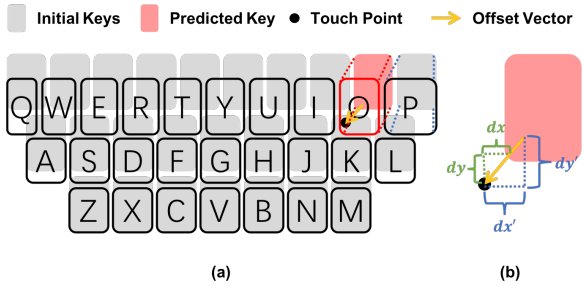


Figure 2: An example of the adapted keyboard layout. (a) The offset of the predicted most intended key (O) and the scale of other keys (e.g., P). (b) Unoptimized offset vector (dx, dy) and optimized offset vector (dx', dy').

Further Optimization Details

While implementing the adaption strategy, we found some important details which could improve the experience of the user during pilot study:

(1) If a second touch lies in the nearby area of the previous input key in the previous layout, then the same layout should

[2] According to our pilot study, if a key is smaller than half of the original key size, either in width or height, it would be easily skipped during finger movement.

be used. This design is based on the observation that most users locate a nearby key according to the position of the last touch. So it may raise confusion to the users if the same location returns different key names. In our prototype, we empirically set this "nearby area" to the same key area of the last touch according to our pilot study.

(2) To minimize the adaptation of the keyboard, the optimal choice of the offset vector is between the boundary of the original key and the touch point. However, this would put the touch point on the boundary between keys after the adaption, which would cause the hit keys (and the corresponding audio feedback) to switch frequently due to finger jitter. To avoid this problem, we deliberately move the touch point more into the key region by about 20% key size to generate an optimized offset vector (Figure 2b).

(3) When scaling keys in the keyboard, the widths and the heights of the keys will change. If the change of a particular key is small, it is more unlikely for the users to feel the difference and a more consistent experience can be provided when exploring over this key. We assume that the intended key is not too much far from the predicted one, so we applied a gradient scale strategy: the keys near the predicted key scale less than those far away from the key.

4 STUDY 1: DERIVING TOUCH MODEL

We conducted a pilot study to collect touch data of BVI users. These data can not only be used to fit a touch model but also help us understand the touch behavior of BVI users. We will use the fitted model to implement our prototype.

Participants and Apparatus

We recruited 8 BVI users (4 males, 4 females) to input words using an interface the same as the traditional keyboard. The average age of the participants was 23.8 (SD=1.4). 4 of the participants were totally blind, while the other 4 were with very low vision. All the participants used screen-reader keyboards in their daily lives and knew the approximate position of each key on the QWERTY layout. Each participant was compensated $8 for the study.

A Huawei P20 phone [4] with a 5.8 inch touchscreen running Android 7.0 was used in this study. The keyboard was implemented in an Android app for full control. The keyboard layout was the same as Gboard[3]. We used Google TTS engine to convert texts to speech output. The experiment setup and the interface are shown in Figure 3.

Design and Procedure

After the introduction of the system, the participant was asked to familiarize with the size and layout of the keyboard as well as the TTS engine output for about 3 minutes. Then each participant was asked to input 100 words with the keyboard. For each word, the system first read out the whole

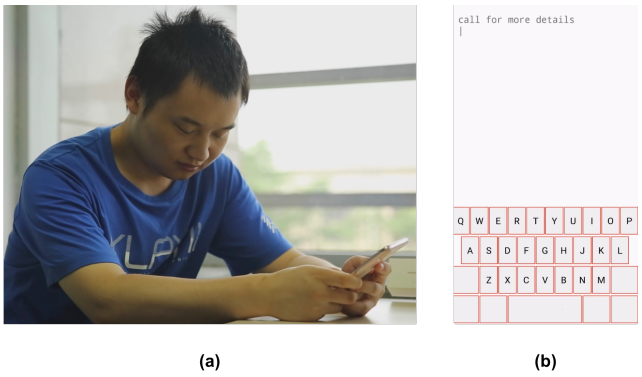


Figure 3: (a) Experiment setup. (b) The interface of the study.

word, then the spelling of the word. The participants were asked to enter the word at the location "as accurately as the perceived locations of the target keys". If the touch location was beyond the border of the keyboard interface, the system would read out "out of range" and the data would not be recorded in this case. Otherwise, the system would give a correct audio feedback no matter where the touch location was. The target characters and the corresponding touch locations were recorded. The participants could restart the task word if they made a mistake in the spelling. We would drop the data containing misspelling to ensure the validity of labeling.

The 100 words used in the study were selected from high-frequency words in ANC corpus [2]. They were selected to contain sufficient number of all the 26 characters.

Results

We removed the outliers which were 3 keys away from the target key center. In total, we get 5383 touch points from 26 different characters. The average input speed was 20.83 (SD=3.29) WPM, which was much higher than the normal input speed of BVI users [11, 32]. The difference was mainly due to the correct audio feedback which removed the calibration time while typing. The average error rate, which was calculated by the boundary of each key was 62.8% (SD=13.2%). We also found that 90% of users' input errors occured within a range of 2.5 key widths. These results are all consistent with the results in [30]. The extremely high error rates indicate an auto-correction method is necessary for BVI users to improve their input performance.

We also fitted the data to 26 bivariate Gaussian distributions, which are shown in Figure 4. While the key width is 6.39 mm and key height is 10.07 mm, the average offset was -0.90 mm for X axis and 3.37 mm for Y axis[3]. This result indicates that users tend to touch shift towards the bottom

[3]Positive directions are right for X axis and bottom for Y axis.

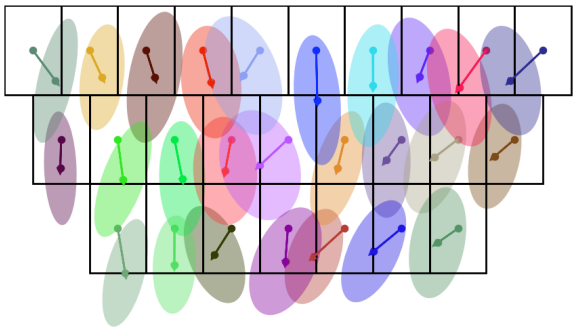


Figure 4: The fitted general touch model. The ellipses show 1 standard deviation of the distribution. The arrows show the offset from each key center to the corresponding distribution center.

of the keyboard. The average standard deviation was 2.92 mm for X axis and 6.47 mm for Y axis, which means more input noise on Y direction than X direction.

5 STUDY 2: PERFORMANCE EVALUATION

We conducted a user study to evaluate the text entry performance of VIPBoard. In the evaluation, we first implemented a prototype of VIPBoard, which supports both English and Chinese input. Then we conducted a within-subjects study to compare the performance between VIPBoard and the traditional keyboard.

Participants

We recruited 14 visually impaired students (7 males, 7 females) from a special education college, with an average age of 23.7 (SD = 1.1). 8 of them were totally blind, while the others were with very low vision. None of them attended Study 1. They were reported to have used touchscreen smartphones for an average of 4.9 years (SD=1.6). All the participants were familiar with QWERTY layout and knew the approximate position of each key. They were native Chinese and knew the spell of English words. Each participant was compensated $30.

Apparatus

The device used in the evaluation was the same as in Study 1. The experiment interface was similar as in Figure 3b. We implemented both English and Chinese input using the same interface. Both keyboards use Google TTS engine to convert text to voice output. For English input, we use the 50,000 words with the highest frequency in ANC corpus [2] as our corpus for VIPBoard. For Chinese input, we use *Pinyin*[4], a phonetic spelling system in Roman characters to input Chinese characters. The corpus of Chinese words and the Pinyin decoding system is transferred from Google Pinyin IME [1]. We used the general touch model collected in Study 1.

After interviewing a group of BVI users, we designed and implemented a set of gesture interactions which were acceptable for them. Users could perform a gesture anywhere on the screen when using either keyboard:

- Swipe left: backspace
- Swipe right: space/confirm
- Swipe up/down: navigate among candidate lists (for Chinese input only, as there exist many homophones with the same Pinyin string)
- Swipe left with two fingers: delete all input contents
- Swipe up with two fingers: read out all input contents

Experiment Design

We used a within-subjects, two-factors design (session and technique) to test the performance of VIPBoard. Participants were asked to enter for 4 sessions with 5 phrases in each session, with both traditional keyboard and VIPBoard, in both English and Chinese. The phrase set for English input was T-40 set from [41], phrases for Chinese input were those with suitable length parsed from [13]. In total, 14 participants $\times$ 20 phrases/condition $\times$ 2 keyboards $\times$ 2 languages = 1120 phrases were entered.

Sample English phrase:	call for more details
Sample Chinese phrase:	希望 你 可以 做 一个 好梦
Corresponding pinyin string:	xiwang ni keyi zuo yige haomeng

Figure 5: Examples of English, Chinese test phrase and the corresponding pinyin string of the Chinese phrase.

Procedure

After the introduction of the basic use and interaction of the keyboard, participants were first asked to entering 2-4 sentences to familiarize with the interaction, keyboard size and the TTS engine. We then informed them that they would use two keyboards which shared the same interaction design without telling them the order they would use. Then each participant was asked to enter phrases under different conditions. The order of using different keyboards was counterbalanced among all the participants. Participants were asked to rest for at least 2 minutes during different sessions. After finishing all the phrases, each participant was asked to give subjective feedback for both two keyboards through a questionnaire and an interview. After they had finished the study and had given comments on the differences between

[4]https://en.wikipedia.org/wiki/Pinyin

two keyboards, we would tell them the order of keyboards they used.

Participants were asked to enter the required text "as quickly and accurately as possible" with their most comfortable postures. Users were allowed to correct errors during typing or leave them uncorrected if they wanted. Each phrase was entered word by word. When entering each phrase, the system will first read out the whole sentence for the user, and then split the current task into words. After the user confirmed the word (which was done by a right swipe), the system will switch to and read out the next word. The participant could swipe down with two fingers to re-listen to the current task.

6 RESULTS

Measurements

Except for traditional text entry performance measurements (e.g., speed, error rate and learning effect), we are also interested in the following measurements:

- **Miss rate (MR).** Miss rate is defined as the ratio of touches which miss the target at the attempt phase. According to the interaction analysis in Section 3, lower MR indicates that fewer touches need the calibration phase, which results in less input time and higher input speed. Lower MR can also provide more confidence and a more smooth input experience for the users. This metric is exactly where the optimization of VIPBoard aims at.
- **Time distribution.** Time distribution can reflect the time cost in different parts of typing and give us a better understanding of the main difference of typing speed between VIPBoard and the traditional technique.

In the following analysis, we will consider English and Chinese input separately with two factors: session and technique. After checking the pre-conditions, we will conduct all the significance testing except for the subjective ratings by repeated measures ANOVA (RM-ANOVA).

Overall Speed

We measured text entry speed in words per minute (WPM), which was calculated as follows [27]:

$$WPM = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5} \tag{3}$$

where $|T|$ is the length of the final input string and S is the elapsed time in seconds from the first to the last touch in a trial. For Chinese input, we use the pinyin string as the final input string (See Figure 5 as an example). The text entry speed was calculated word by word without considering the time of selecting candidates.

Table 1: Average text entry speed (WPM) of two keyboards

	VIPBoard	Traditional keyboard
English input	8.14 (SD=1.62)	7.23 (SD=1.67)
Chinese input	8.61 (SD=2.04)	7.57 (SD=1.95)

RM-ANOVA shows that text entry speed on VIPBoard is significantly higher than on the traditional keyboard under both languages ($F_{1,13} = 23.67, p < .01$ for English input; $F_{1,13} = 9.51, p < .01$ for Chinese input). As shown in Table 1, the average text entry speed by using VIPBoard was 12.6% (13.7%) higher than by using the traditional keyboard for English (Chinese) input. On VIPBoard, users typed around 5 more characters in a minute on average.

Overall Error Rate

We measured character-level error rate using corrected error rate (CER) and uncorrected error rate (UER) [39]. Users tended to fix most, if not all, of their errors during typing, leaving few in the final transcribed string, as shown in Table 2. No significant difference was found by RM-ANOVA bewteen CER ($p = .99$ for English input; $p = .95$ for Chinese input) and UER ($p = .11$ for English input; $p = .37$ for Chinese input) of the two keyboards. The results showed that the error processing on the two keyboards was not different for BVI users.

Table 2: Average corrected (CER) and uncorrected (UER) error rates of two keyboards

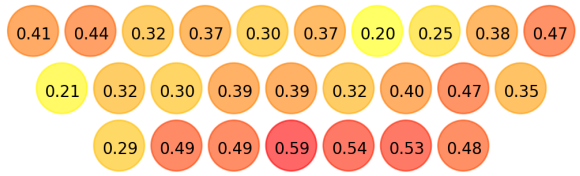
		VIPBoard	Traditional keyboard
CER	English input	3.51% (SD=1.92%)	3.54% (SD=2.25%)
	Chinese input	4.46% (SD=2.30%)	4.48% (SD=2.60%)
UER	English input	1.30% (SD=2.36%)	1.94% (SD=2.28%)
	Chinese input	1.28% (SD=1.88%)	1.03% (SD=1.52%)

Miss Rate

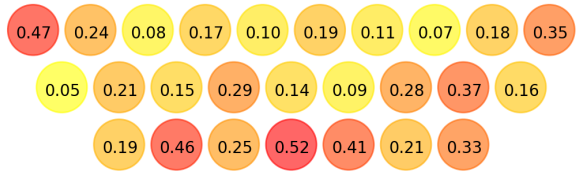
As shown in Table 3, VIPBoard could reduce 63.0% (60.0%) hit errors compared with the traditional keyboard for English (Chinese) input. RM-ANOVA shows significant improvement under both languages ($F_{1,13} = 152.73, p < .01$ for English; $F_{1,13} = 67.34, p < .01$ for Chinese). We also calculated the miss rate of each key for both techniques, as shown in Figure 6. From the figures we can conclude that miss rates of VIPBoard are all lower than those on the traditional keyboard except for the "q" key, which may due to the small sample size (less than 100) in the study.

Table 3: Average miss rate (MR) of two keyboards

	VIPBoard	Traditional keyboard
English input	14.2% (SD=8.10%)	32.8% (SD=12.9%)
Chinese input	14.8% (SD=6.30%)	34.6% (SD=14.1%)



(a) Traditional keyboard.



(b) VIPBoard.

Figure 6: Miss rates of each key.

Time Distribution

We divided the time of inputting one whole word into four parts:

- **Entering time**. The time of entering each character, which starts with a touchdown event and ends with a touchup event.
- **Interval time**. The elapsed time between two adjacent touches.
- **Confirm time**. The time of swiping right (English input) or the time from navigating in the candidate list to swiping right (Chinese input).
- **Other time**. Time of deletion and other interactions (e.g., swiping up with two fingers or other undefined gestures).

We calculated the average time of each part when entering one character, which were shown in Figure 7. RM-ANOVA shows that entering time of VIPBoard was significantly shorter than traditional keyboard ($F_{1,13} = 21.0, p < .01$ for English input; $F_{1,13} = 7.26, p < .05$ for Chinese input), while no significant difference was found among other three parts. This result shows that VIPBoard could effectively reduce the average entering time, and not influence other input parts at the same time. The results also indicate that there is possible space for more efficient accessible soft keyboard design in the future, such as reducing interval time.

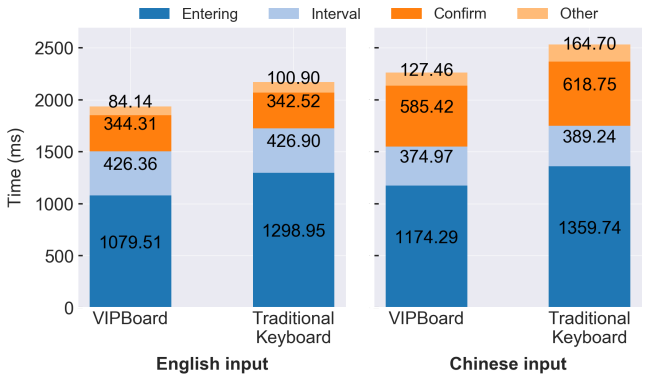


Figure 7: Time distribution of two keyboards.

Learning Effect

Speed vs. Session. We calculated average text entry speeds in each session, which were shown in Figure 8. RM-ANOVA shows a significant learning effect of session ($F_{3,39} = 4.92, p < .01$ for English input; $F_{3,39} = 12.56, p < .001$ for Chinese input). Post-hoc pairwise comparison with Sidak adjustment showed that the text entry speeds in session 3 and 4 were significantly higher than session 1, which indicates that users could achieve a relatively high speed after practicing for only 5-10 sentences.

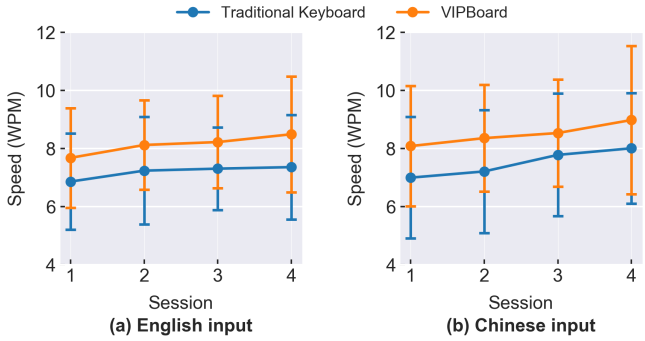


Figure 8: Average text entry speed in each session. The error bars show the standard deviations.

Miss Rate vs. Session. We calculated average MRs in each session, which were shown in Figure 9. RM-ANOVA found no significant learning effects of MR ($p = .09$ for English, $p = .13$ for Chinese), which indicates users could achieve a relatively low MR quickly after they started to use VIPBoard.

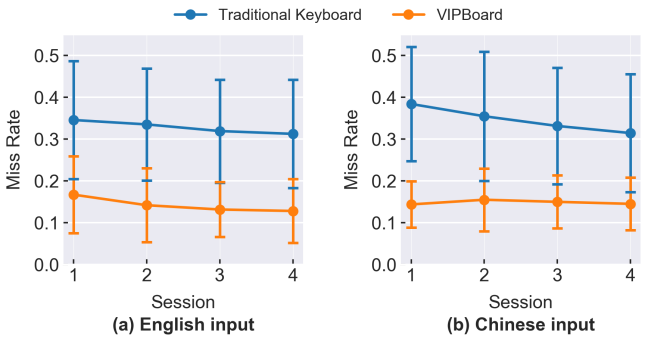


Figure 9: Average miss rate in each session. The error bars show the standard deviations.

Subjective Feedback

Subjective Ratings. After finishing the text entry part, each participant was asked to evaluate two keyboards with a 7-point Likert scale for mental demand, physical demand, temporal demand, effort, fatigue and overall preference. The average scores and the standard deviations were shown in Table 4. Wilcoxon tests show that VIPBoard outperforms the traditional keyboard significantly among all the aspects ($p < .01$).

Table 4: Average subjective ratings of two keyboards. The higher the score is, the better the keyboard is.

	VIPBoard	Traditional keyboard
Mental demand	5.79 (SD=1.05)	3.79 (SD=1.37)
Physical demand	5.36 (SD=1.01)	4.00 (SD=1.04)
Temporal demand	5.79 (SD=0.89)	3.79 (SD=1.81)
Effort	5.64 (SD=0.84)	3.71 (SD=1.86)
Fatigue	5.57 (SD=0.65)	3.36 (SD=1.34)
Overall preference	6.14 (SD=0.77)	3.36 (SD=1.28)

Perception of Layout Adaption. It is worth noticing that layout adaption in VIPBoard is "transparent" to BVI users because they only rely on the audio feedback to complete the input. Current adaption strategy is designed for consistency of use and the visualization is for demonstration only. Almost all of the participants (12/14) could feel the reduction of miss rate, but they could not recognize the role of our algorithm. In their cognition, they felt they "touched more accurately" on one keyboard with the same interactions.

However, there were few very skilled participants (1/14) who could perceive the changing boundary of keys among different touches. The change may cause confusion of the key location for these users, which may slightly influence the confidence and then the typing performance of them. However, according to the interview, he could adapt to the interface and overcome the discomfort very quickly, so we still claim that VIPBoard is a good solution for BVI users to enter text on a soft keyboard. A further long-term study may be needed to study the change of the user's mental model due to layout adaption, which we leave as future work.

Qualitative Feedback. During interview, participants showed highly preference for the consistent interaction design.

"The keyboard is very easy to learn and use, I nearly cannot feel the difference between the two keyboards except for the high touch accuracy when using VIPBoard." (P3, P8)

The gesture interactions have also been proved as a suitable design for BVI users.

"I love the gestures designed for the keyboards. They are convenient and can avoid misoperations." (P7, P9)

Besides, participants showed high expectation and strong willingness to use VIPBoard in their everyday lives.

"I hope VIPBoard can be integrated into commercial input methods with more encoding supported, like Shuangpin[5]*. I can not wait for using it on my own phone."* (P1, P3, P5, P10, P12)

7 PERFORMANCE ANALYSIS

In this section, we will analyze the factors which can affect the final performance of our algorithm.

Effect of Language Model

We found our algorithm highly relies on the language model. We calculated the number of available choices for the next character after a part of the word has already been input according to the corpus we used, which was shown in Figure 10a. We can find a sharp drop of the number of choices along with the input length: after inputting 2 characters, an average of 17.5 characters could be the next input one; after inputting 4 characters, only 3.6 characters could be the next one. With longer input, the number decreased to under 2.

We further analyze the result by considering the distribution of the available characters on the keyboard. Recall that we found that most users' input errors were within 2.5 key widths in Study 1. Thus we calculated the average number of available characters within 2.5 key widths of the target key center in Figure 10b. The results here are much smaller than that in Figure 10a: after inputting 4 characters, there are only 0.8 keys available around the target key within 2.5 key widths, where about 85% of the words were still not finished at this time. This result indicates that our algorithm could get a relatively accurate prediction result without regard to the accuracy of the touch model. We also collect each participant's personalized touch model according to their input

[5]Another phonetic-based Chinese input method, see https://en.wikipedia.org/wiki/Chinese_input_methods_for_computers#Shuangpin.

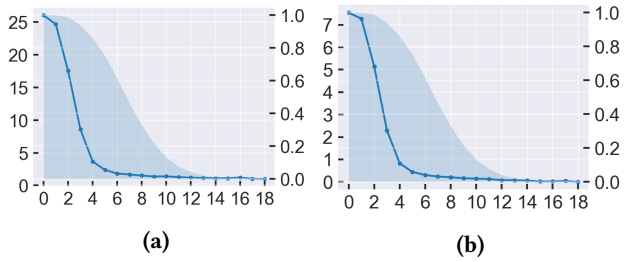


Figure 10: (a) Number of available choices for next input vs. input length. (b) Number of available keys within 2.5 key widths of the target key vs. input length. The shadow areas show the percentage of words longer than the input length on the right y-axis.

data and ran a simulation using the personalized touch model to check whether the miss rate would reduce compared with the general model results. The result, which showed nearly no improvement, also confirmed this analysis.

Effect of Touch Accuracy

We are also interested in how a user's touch accuracy affect the performance after using VIPBoard. We thus use the miss rate on the traditional keyboard as the measurement of touch accuracy. Higher miss rate indicates that the user's touches are with more input noise. We calculated the miss rate difference between the traditional keyboard and VIPBoard to measure the gain by using VIPBoard. A higher difference indicates that the user can benefit more from VIPBoard. We calculated these two values for all the 14 participants under 2 languages and plotted them in Figure 11.

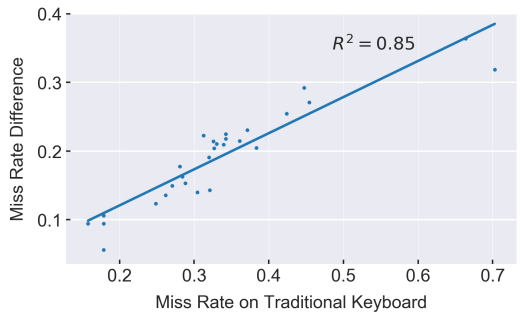


Figure 11: Miss rate difference vs. touch accuracy.

From Figure 11, we found a relatively strong linear positive correlation between the performance gain and touch accuracy. We conducted a linear regression and plot the fitted line in the figure ($R^2 = 0.85$). Thus we can conclude that users with lower touch accuracy can benefit more from using VIPBoard.

8 LIMITATIONS AND FUTURE WORK

In this section, we will analyze some limitations in the work, which may also help us improve VIPBoard in the future.

In study 2, we recruited only 14 participants to evaluate text entry performance. The small amount is due to the high requirements of BVI participants: they were required to be familiar with screen readers, QWERTY layout, both English and Chines pinyin input. However, a wide range of different touch accuracy (miss rate from 17.4% to 67.6%, as shown in Figure 11) was covered by these participants, and the results were consistent among all the participants. This indicates that the results in this paper are convincing and universal.

Currently, we applied a unigram language model and a general touch model in VIPBoard. However, it has been proved that a more detailed language model (e.g., n-gram model [23], personalized language model [16, 36]) could provide better performance for text entry tasks [23, 42]. Also, auto-completion may further improve the performance. We will consider these optimizations in the future.

Non-alphabetic characters (e.g., numbers, punctuations) are also important for text input and can be easily added in VIPBoard. We can add a symbol keyboard to enter special characters and use another gesture (e.g., swipe down with two fingers) to switch between different keyboards.

9 CONCLUSION

In this paper, we propose VIPBoard, a smart keyboard for BVI users which applies a character-level auto-correction algorithm to augment screen-reader keyboard. VIPBoard predicts the most probable character of users' intention, which aims at reducing the calibration time of BVI users. Then we designed a layout adaptation strategy, which provides consistent interactions and experience with traditional screen-reader keyboards. In an evaluation study of 14 BVI users, we found that VIPBoard could improve text entry speed by 12.6%, and reduce the miss rate of each touch by 63%. Users could master VIPBoard after a small amount of practice and they provided positive feedback compared with the traditional keyboard. VIPBoard indicates that adding auto correction to a screen-reader keyboard, though only at the character level, improved the performance of text entry and was widely welcomed by BVI users. We hope VIPBoard can bring smartness to screen-reader soft keyboards and benefit BVI users.

ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Plan under Grant No. 2016YFB1001200, the Natural Science Foundation of China under Grant No. 61672314 and No. 61572276, Tsinghua University Research Funding No. 20151080408, and also by Beijing Key Lab of Networked Multimedia.

REFERENCES

[1] 2009. PinyinIME. Retrieved August 23, 2018 from https://android.googlesource.com/platform/packages/inputmethods/PinyinIME/

[2] 2011. The Open American National Corpus(OANC). Retrieved August 23, 2018 from http://www.anc.org

[3] 2018. Gboard - the Google Keyboard. Retrieved January 4, 2019 from https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin&hl=en_US

[4] 2018. Huawei P20 Smartphone. Retrieved January 4, 2019 from https://consumer.huawei.com/en/phones/p20/

[5] Khaldoun Al Faraj, Mustapha Mojahid, and Nadine Vigouroux. 2009. BigKey: A Virtual Keyboard for Mobile Devices. In *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*, Julie A. Jacko (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 3–10.

[6] Shiri Azenkot, Jacob O. Wobbrock, Sanjana Prasain, and Richard E. Ladner. 2012. Input Finger Detection for Nonvisual Touch Screen Text Entry in Perkinput. In *Proceedings of Graphics Interface 2012 (GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 121–129. http://dl.acm.org/citation.cfm?id=2305276.2305297

[7] Shiri Azenkot and Shumin Zhai. 2012. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 251–260. https://doi.org/10.1145/2371574.2371612

[8] Tyler Baldwin and Joyce Chai. 2012. Towards Online Adaptation and Personalization of Key-target Resizing for Mobile Devices. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces (IUI '12)*. ACM, New York, NY, USA, 11–20. https://doi.org/10.1145/2166966.2166969

[9] Nikola Banovic, Tovi Grossman, and George Fitzmaurice. 2013. The Effect of Time-based Cost of Error in Target-directed Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1373–1382. https://doi.org/10.1145/2470654.2466181

[10] Tom Bellman and Scott I. Mackenzie. 1998. A Probabilistic Character Layout Strategy for Mobile Text Entry. In *Graphics Interface*. 168–176. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.898

[11] Matthew N. Bonner, Jeremy T. Brudvik, Gregory D. Abowd, and W. Keith Edwards. 2010. No-Look Notes: Accessible Eyes-Free Multi-touch Text Entry. In *Pervasive Computing*, Patrik Floréen, Antonio Krüger, and Mirjana Spasojevic (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 409–426.

[12] Stephen Brewster, Stephen Brewster, Faraz Chohan, and Lorna Brown. 2007. Tactile Feedback for Mobile Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 159–162. https://doi.org/10.1145/1240624.1240649

[13] Tao Chen and Min-Yen Kan. 2013. Creating a live, public short message service corpus: the NUS SMS corpus. *Language Resources and Evaluation* 47, 2 (01 Jun 2013), 299–335. https://doi.org/10.1007/s10579-012-9197-9

[14] G. Chomchalerm, J. Rattanakajornsak, U. Samsrisook, D. Wongsawang, and W. Kusakunniran. 2014. Braille dict: Dictionary application for the blind on android smartphone. In *2014 Third ICT International Student Project Conference (ICT-ISPC)*. 143–146. https://doi.org/10.1109/ICT-ISPC.2014.6923237

[15] Cecil D'silva, Vickram Parthasarathy, and Sethuraman N. Rao. 2016. Wireless Smartphone Keyboard for Visually Challenged Users. In *Proceedings of the 2016 Workshop on Wearable Systems and Applications (WearSys '16)*. ACM, New York, NY, USA, 13–17. https://doi.org/10.1145/2935643.2935648

[16] Andrew Fowler, Kurt Partridge, Ciprian Chelba, Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2015. Effects of Language Modeling and Its Personalization on Touchscreen Typing Performance. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 649–658. https://doi.org/10.1145/2702123.2702503

[17] Brian Frey, Caleb Southern, and Mario Romero. 2011. BrailleTouch: Mobile Texting for the Visually Impaired. In *Universal Access in Human-Computer Interaction. Context Diversity*, Constantine Stephanidis (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 19–25.

[18] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. WalkType: Using Accelerometer Data to Accomodate Situational Impairments in Mobile Touch Screen Text Entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2687–2696. https://doi.org/10.1145/2207676.2208662

[19] Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: Using Hand Posture Information to Improve Mobile Touch Screen Text Entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2795–2798. https://doi.org/10.1145/2470654.2481386

[20] D. Gonçalves, J. A. Jorge, H. Nicolau, T. Guerreiro, and P. Lagoá. 2008. From Tapping to Touching: Making Touch Screens Accessible to Blind Users. *IEEE MultiMedia* 15 (12 2008), 48–50. https://doi.org/10.1109/MMUL.2008.88

[21] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 194–195. https://doi.org/10.1145/502716.502753

[22] Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability Guided Key-target Resizing for Soft Keyboards. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. ACM, New York, NY, USA, 111–118. https://doi.org/10.1145/1719970.1719986

[23] Nils Klarlund and Michael Riley. 2003. Word N-grams for Cluster Keyboards. In *Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods (TextEntry '03)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 51–58. http://dl.acm.org/citation.cfm?id=1628195.1628202

[24] D. Kocieliński and J. Brzostek-Pawłowska. 2013. Improving the accessibility of touchscreen-based mobile devices: Integrating Android-based devices and Braille notetakers. In *2013 Federated Conference on Computer Science and Information Systems*. 655–658.

[25] Per-Ola Kristensson and Shumin Zhai. 2005. Relaxing Stylus Typing Precision by Geometric Pattern Matching. In *Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI '05)*. ACM, New York, NY, USA, 151–158. https://doi.org/10.1145/1040830.1040867

[26] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. 2017. BlindType: Eyes-Free Text Entry on Handheld Touchpad by Leveraging Thumb's Muscle Memory. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 18 (June 2017), 24 pages. https://doi.org/10.1145/3090083

[27] I Scott Mackenzie. 2002. A note on calculating text entry speed. *Unpublished work. Available online at http://www. yorku. ca/mack/RN-TextEntrySpeed. html* (2002).

[28] Sergio Mascetti, Cristian Bernareggi, and Matteo Belotti. 2012. TypeInBraille: Quick Eyes-Free Typing on Smartphones. In *Computers Helping People with Special Needs*, Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 615–622.

[29] Hugo Nicolau, Kyle Montague, Tiago Guerreiro, João Guerreiro, and Vicki L. Hanson. 2014. B#: Chord-based Correction for Multitouch Braille Input. In *Proceedings of the 32Nd Annual ACM Conference on*

Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 1705–1708. https://doi.org/10.1145/2556288.2557269

[30] Hugo Nicolau, Kyle Montague, Tiago Guerreiro, André Rodrigues, and Vicki L. Hanson. 2015. Typing Performance of Blind Users: An Analysis of Touch Behaviors, Learning Effect, and In-Situ Usage. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '15)*. ACM, New York, NY, USA, 273–280. https://doi.org/10.1145/2700648.2809861

[31] João Oliveira, Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2011. BrailleType: Unleashing Braille over Touch Screen Mobile Phones. In *Human-Computer Interaction – INTERACT 2011*, Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 100–107.

[32] João Oliveira, Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. 2011. Blind People and Mobile Touch-based Text-entry: Acknowledging the Need for Different Flavors. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '11)*. ACM, New York, NY, USA, 179–186. https://doi.org/10.1145/2049536.2049569

[33] Mathieu Raynal and Philippe Truillet. 2007. Fisheye Keyboard: Whole Keyboard Displayed on PDA. In *Human-Computer Interaction. Interaction Platforms and Techniques*, Julie A. Jacko (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 452–459.

[34] Jaime Sánchez and Fernando Aguayo. 2007. Mobile Messenger for the Blind. In *Universal Access in Ambient Intelligence Environments*, Constantine Stephanidis and Michael Pieper (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 369–385.

[35] Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. 2018. TOAST: Ten-Finger Eyes-Free Typing on Touchable Surfaces. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 33 (March 2018), 23 pages. https://doi.org/10.1145/3191765

[36] KUMIKO TANAKA-ISHII. 2007. Word-based predictive text entry using adaptive language models. *Natural Language Engineering* 13, 1 (2007), 51–74. https://doi.org/10.1017/S1351324905004080

[37] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 659–668. https://doi.org/10.1145/2702123.2702135

[38] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain Text Entry on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2307–2316. https://doi.org/10.1145/2556288.2557412

[39] Jacob O. Wobbrock and Brad A. Myers. 2006. Analyzing the Input Stream for Character- Level Errors in Unconstrained Text Entry Evaluations. *ACM Trans. Comput.-Hum. Interact.* 13, 4 (Dec. 2006), 458–489. https://doi.org/10.1145/1188816.1188819

[40] Georgios Yfantidis and Grigori Evreinov. 2006. Adaptive blind interaction technique for touchscreens. *Universal Access in the Information Society* 4, 4 (01 May 2006), 328–337. https://doi.org/10.1007/s10209-004-0109-7

[41] Xin Yi, Chun Yu, Weinan Shi, Xiaojun Bi, and Yuanchun Shi. 2017. Word Clarity As a Metric in Sampling Keyboard Test Sets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4216–4228. https://doi.org/10.1145/3025453.3025701

[42] Xin Yi, Chun Yu, Weinan Shi, and Yuanchun Shi. 2017. Is it too small?: Investigating the performances and preferences of users when typing on tiny QWERTY keyboards. *International Journal of Human-Computer Studies* 106 (2017), 44 – 62. https://doi.org/10.1016/j.ijhcs.2017.05.001

[43] Xin Yi, Chun Yu, Weijie Xu, Xiaojun Bi, and Yuanchun Shi. 2017. COMPASS: Rotational Keyboard on Non-Touch Smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 705–715. https://doi.org/10.1145/3025453.3025454

[44] Suwen Zhu, Tianyao Luo, Xiaojun Bi, and Shumin Zhai. 2018. Typing on an Invisible Keyboard. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 439, 13 pages. https://doi.org/10.1145/3173574.3174013