BlindType: Eyes-Free Text Entry on Handheld Touchpad by Leveraging Thumb's Muscle Memory

YIQIN LU, CHUN YU, XIN YI, and YUANCHUN SHI, Department of Computer Science and Technology, Tsinghua University, Tsinghua National Laboratory for Information Science and Technology, and Beijing Key Lab of Networked Multimedia
SHENGDONG ZHAO, National University of Singapore

Eyes-free input is desirable for ubiquitous computing, since interacting with mobile and wearable devices often competes for visual attention with other devices and tasks. In this paper, we explore eyes-free typing on a touchpad using one thumb, wherein a user taps on an imaginary QWERTY keyboard while receiving text feedback on a separate screen. Our hypothesis is that users can transfer their typing ability obtained from visible keyboards to eyes-free use. We propose two statistical decoding algorithms to infer users' eyes-free input: the absolute algorithm and the relative algorithm. The absolute algorithm infers user input based on the absolute position of touch endpoints, while the relative algorithm infers based on the vectors between successive touch endpoints. Evaluation results showed users could achieve satisfying performance with both algorithms. Text entry rate was 17-23 WPM (words per minute) depending on the algorithm used. In comparison, a baseline cursor-based text entry method yielded only 7.66 WPM. In conclusion, our research demonstrates for the first time the feasibility of thumb-based eyes-free typing, which provides a new possibility for text entry on ubiquitous computing platforms such as smart TVs and HMDs.

CCS Concepts: • **Human-centered computing → Text input**; **Ubiquitous computing**; **Mobile devices**;

Additional Key Words and Phrases: Eyes-free input, text entry, smart TV interaction, HMDs

ACM Reference format:
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

1 INTRODUCTION

In the post-PC era, human-computer interaction increasingly occurs on mobile and wearable computing devices. Interacting with these computing devices often competes for visual attention with other tasks (e.g., mobility tasks) [21]. It is thus desirable if interaction with computing devices can be performed in an eyes-free manner [31].

This work is supported by the National Key Research and Development Plan under Grant No. 2016YFB1001200, the Natural Science Foundation of China under grant 61672314, grant 61572276, and grant 61521002, and Tsinghua-Samsung Beijing R&D Center Joint Lab.
Author's addresses: Y. Lu, C. Yu, X. Yi, and Y. Shi, Department of Computer Science and Technology, Tsinghua University; S. Zhao, National University of Singapore.

2474-9567/2017/6-ART18 $15.00
https://doi.org/10.1145/3090083

While there are a few well-designed mobile eyes-free menu selection techniques (e.g., earPod [33], blindsight [16], etc.), entering text eyes-freely in a mobile scenario remains a challenging problem. Although eyes-free text input can be easily achieved using a physical keyboard, in a mobile scenario, users today often need to interact with a virtual keyboard on a touch surface and maneuver using a single-handed approach. The combination of these constraints makes accurate and quick eyes-free text input much more difficult to achieve [4, 6, 26]. Voice input can be a viable alternative. However, speaking in public can be disruptive and raise privacy/security concerns. Thus, it is desirable to investigate an effective touch-based eyes-free text input method for mobile scenario.

We envisioned a future scenario in which an eyes-free touch-based text input method would be particularly useful: a user handholds a touchpad as input and interacts with a second output screen such as HMD, large display, etc. Note that in this scenario, the input surface and the output surface are decoupled (see Fig. 1 and Fig. 14). Users touch type on an "imaginary keyboard" on the touchpad eyes-freely while receiving feedback from the HMD or large display. Compared to the current approach where users hold a mobile phone and type the text on the screen, this approach does not require a user to switch his/her attention between the input surface and the output surface. Users can focus on the output screen to receive information about the text input as well as the environment. This also allows for a more natural posture while performing text input: instead of looking down and bending the arm to hold the mobile phone while typing, one can stand up, and look straight, while typing with one hand with the arm remaining mostly straight.

In this paper, we explore the possibility of eyes-free text entry using one thumb: a user does not look at the touchpad, but relies on his/her muscle memory to move the thumb and tap. Meanwhile, the output screen only displays candidate words for the user to select. By this means, we expect a typing speed close to that obtained in normal direct touch condition with visual feedback.

We recognize three research questions for eyes-free typing:

(1) Can a user transfer his/her typing ability derived from normal condition to eyes-free use?

(2) If so, what new typing pattern (in terms of the distribution of touch endpoints [3, 8, 10]) will emerge?

(3) Further, can we leverage the existing text entry algorithm to decode users' eyes-free input? Or do we need a new algorithm that more effectively accounts for eyes-free input?

To answer these questions, we conducted a series of studies and explorations. We first collected eyes-free typing data from real users and examined the shape, size and location of the "imaginary keyboard" emerging from touch endpoints. Specially, we formed a new hypothesis of human eyes-free typing behavior, the relative input model, which assumes individual finger tapping action is not independent of each other; it is programmed relative to the last tapping. We obtained quantitative evidence for the validity of the relative model.

Based on the findings, we characterized two statistical decoding algorithms for text entry. The absolute algorithm, following the classical method, treats individual tap operations independently and predicts user input based on the absolute location of touch endpoints. The relative algorithm, a novel approach, predicts based on the relative position (the vector) between successive endpoints. In addition, both algorithms leverage a transformed keyboard model reflecting users' eyes-free typing pattern.

Finally, we conducted two studies to examine and compare the performance of different algorithm designs (*Absolute* vs. *Relative*, *General* vs. *Personalized*). Results showed eyes-free typing could achieve 17-23 WPM (depending on the algorithm used) with low uncorrected errors after users practiced only a few words, which was 2.25-2.97 times of the speed of a baseline cursor-based technique.

The contribution of this work is as follows:

(1) We propose a regression approach to derive a keyboard model from users' eyes-free typing data, which characterizes the spatial parameters (i.e. the keyboard location and size) of the "imaginary keyboard" in eyes-free typing.

(2) We empirically studied users' eyes-free typing behavior and reported detailed parameters of keyboard model, including its size and location as well as the standard deviation of touch points during users' eyes-free tapping.

(3) Based on the empirical results, we propose two text decoding algorithms (the absolute algorithm and the relative algorithm) that can effectively decode users' eyes-free input. The achieved input speed is close to that of normal typing on a visible keyboard.

2 RELATED WORK

We first review different approaches to realize eyes-free text entry in literature. We then discuss the most widely used algorithm that interprets users' inaccurate input on touchscreens.

2.1 Eyes-Free Text Entry Techniques

Eyes-free text entry is most familiar on physical keyboards [7, 17], where key boundaries provide tactile feedback for fingers to navigate without visual feedback. In this paper, we focus on touch-based eyes-free interaction. We also focus on one-handed interaction as it is often more desirable than two-handed interaction for handheld devices [13].

Unfortunately, eyes-free touch is inaccurate on a flat input surface. Researchers have investigated this issue in the context of absolute indirect-touch pointing. Wang et al. [29] found that the accuracy of eyes-free thumb-based touch was only 85% when target size was 12.5 mm. Gilliot et al. [9] reported the minimal target size (with 95% acquisition accuracy) of eyes-free touch to be 22.3 mm when inputting with the index finger. Since key size is much smaller (4mm - 6mm) on commercial software keyboards, eyes-free input is conceivably even more inaccurate.

Character-level gesture input is an important approach to realize eyes-free text entry. Previous work [26] showed eyes-free unistroke input could achieve 10 WPM with word-level correction. Escape-Keyboard [4] allowed users to type letters with a flick gesture starting from different areas of the screen. Users achieved 14.7 WPM with an error rate of 4.36% after 16 twenty-minute training sessions. No-look Notes [6] was specially designed for blind people, using TTS as feedback. The interaction was bi-manual: one finger specified a group of characters, and the other specified a character within the chosen group. Evaluation results showed the text entry rate was 1.67 WPM.

Word-level gesture keyboard (e.g. SHARK [14]) is also a potential candidate for eyes-free typing. Users type by performing a gesture that traverses individual keys constituting the desired word in sequence. Empirical studies showed that users could learn about 15 gestures per hour of practice on a visible keyboard. Kristensson et al. [15] later extended the elastic matching algorithm of gesture keyboard to tapping-based stylus typing. Recently, Markussen et al. [20] investigated the possibility of eyes-free use of the gesture keyboard in mid-air condition. Unfortunately, results showed visual tracing was indispensable even after users practiced the same word repeatedly.

Eyes-free typing with multiple fingers has also been researched. Findlater et al. [8] examined the typing patterns of expert users performing ten-finger typing on a flat surface. The keyboard visibility was manipulated. However, no specific algorithm was proposed to realize eyes-free typing. Azenkot et al. [2] studied input finger detection and proposed Perkinput, a 6-bit Braille text entry on touchscreen, which could achieve 17.56 WPM for one hand and 38.0 WPM for two hands after training. Later Southern et al. [25] studied the performance of BrailleTouch, which adapted 6-key chorded braille input to a touchscreen phone. Results showed expert blind users could achieve 23.2 WPM after 165 twenty-minute sessions of training. Sandwich Keyboard [24] leverages an adaptive algorithm to enable users to type with ten fingers on the back side of a tablet device. Evaluation results showed experienced typists reached 46.2 WPM (error rate = 9.8%) after eight hours of training. Vertanen et al. [27] asked participants to tap out in a completely blind condition with two fingers. They then devised a sentence-level prediction algorithm to interpret the input. Preliminary results showed a character-level error rate of 18.4% and a word-level error rate of 30.1%.

To the best of our knowledge, eyes-free typing with a single thumb has not yet been realized in literature.

2.2 Statistical Decoding Algorithm

When typing on touchscreens, the major problem is the tap variability that arises from various factors, such as fat finger [28], hand postures [3], target size [22], mobility [10], etc. To address this, the most commonly used approach is Bayesian decoding, first described by Goodman et al. [11]. The basic idea is to compute a word with maximum a posteriori of input sequence given a language model. To simplify computation, no correlation is usually assumed between individual key-press actions; a user performs each key press (touch, stylus tapping, or mouse click) independently. The mathematical derivation of Bayesian decoding is described in section 5 (see page 12). However, such a simplification does not consider the possible dependency between key-press actions.

Rashid and Smith [23] first described the concept of relative input, which sought to enable ten-finger touch typing anywhere on a flat surface. Their relative algorithm predicted the target word based on offsets of all following endpoints from the first one. The hypothesis described keyboard location as unknown to the system. Preliminary results showed Top-3 accuracy was only 45.6% in offline simulation. In contrast, our relative method is designed for single-thumb input and based on the vector between successive touch points. Our hypothesis conveys users perform a touch relative to the last touch in eyes-free input mode. In addition, our algorithm incorporates a keyboard model that is derived from empirical data of users' eyes-free typing behavior, which is also a key component for the success of our technique.

3 USER STUDY 1: COLLECTING USERS' EYES-FREE TYPING DATA

This experiment has two goals: First, we want to examine whether a user is subjectively able and willing to perform eyes-free typing, given he/she is familiar with typing on a visible keyboard. The second goal is to obtain users' eyes-free typing data (touch point data), based on which we can perform further analysis. To our knowledge, several works [3, 8, 10, 22, 29] have been conducted to collect and analyze typing data on a visible keyboard. However, none has been conducted for eyes-free text entry.

3.1 Participants

12 participants (11 males and 1 female) were recruited from the university campus. They were aged from 20-32, all right-handed. They used a QWERTY keyboard to enter text on their personal mobile phones on daily basis. None had experienced eyes-free text entry before. Participants were paid 20 dollars to compensate their time.

3.2 Apparatus and Experimental System

We used a 4.3-inch Android phone to emulate a remote control. The size of the touch area was 53.6 mm $\times$ 95.2 mm, with resolution 720 $\times$ 1080. We used a 50-inch smart TV to display the text feedback. Participants sat on a chair 1.5 meters away from the smart TV (Fig. 1). We asked the participants to type with their right thumb on the touchscreen of the phone without looking at it. Instead, we instructed they should focus their attention on text feedback on the smart TV. In addition, an image of a QWERTY keyboard (without any feedback) was displayed on the smart TV for the user to refer to in case he/she was not sure about the positions of key (which seldom happened during experiment according to our observations).

We developed an Android application to collect users' touch input on the phone and forward the touch events to a computer through Wi-Fi. The display and record software on the computer was developed with C# WPF .NET Framework 4.5 running in Microsoft Windows 8.

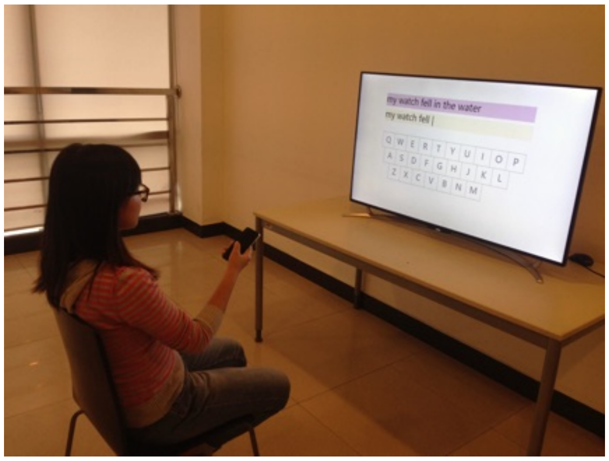


Fig. 1. Experimental setting of Study 1: the user typed text in front of a smart TV with a touchscreen in an eyes-free condition. During the tapping, user's eye attention was on the TV instead of the touchscreen where nothing was displayed.

The purpose of this study was to collect users' natural eyes-free typing data that could be automatically labelled by software. However, since we did not have a concrete prediction algorithm at this stage, we designed the experiment system to display an always correct character (according to the desired phrase) no matter where the participant tapped on the touchscreen. We told participants that the future algorithm would be intelligent enough to interpret the input. A similar experimental design was leveraged to collect ideal data in previous text entry research [3, 8].

In addition, we designed a left-swipe gesture for deleting a just-entered letter (to substitute for the "backspace" key), and a right-swipe gesture for entering a space (to substitute the "space" key). Each time a space was entered, the experimental system checked whether the number of touch endpoints (separated by spaces) and the number of letters in the desired word were equal. If not, the system displayed asterisks to indicate the mismatch. In that case, participants were then required to delete the involved characters and input again.

3.3 Design and Procedure

Before the experiment, we introduced the goal of the experiment to the participants, explained the task, and demonstrated how to perform the task.

The experiment had two sessions. In the training session, we asked participants to familiarize themselves with eyes-free typing and the task. We did not instruct participants on how to handle the phone and where the keyboard was, participants just assumed the keyboard was in a typical position and performed their typing as spontaneously as in their daily use. In the test session, each participant was required to type 150 phrases from Mackenzie's phrases set [19]. The current phrase to be inputted was displayed on the TV screen. Participants were instructed to input phrases "as fast and accurate as possible" and encouraged to take a break after inputting each set of 10 phrases. The experiment took about 50 minutes.

We employed two mechanisms to ensure the participants performed the typing eyes-freely. First, the phone screen was blacked out during the whole experiment, which provided no feedback. Second, the experimenter sat beside the participant to supervise the process, so that the participant kept focus on the TV screen during the whole experiment.

3.4 Data Processing

We collected typing data of 9,664 words with 40,509 letters in total. For each key, we removed outliers that were more than three times standard deviation away from the collected centroid in either X or Y dimension. These accounted for 0.82% of letter-point pairs in our data.

3.5 Results

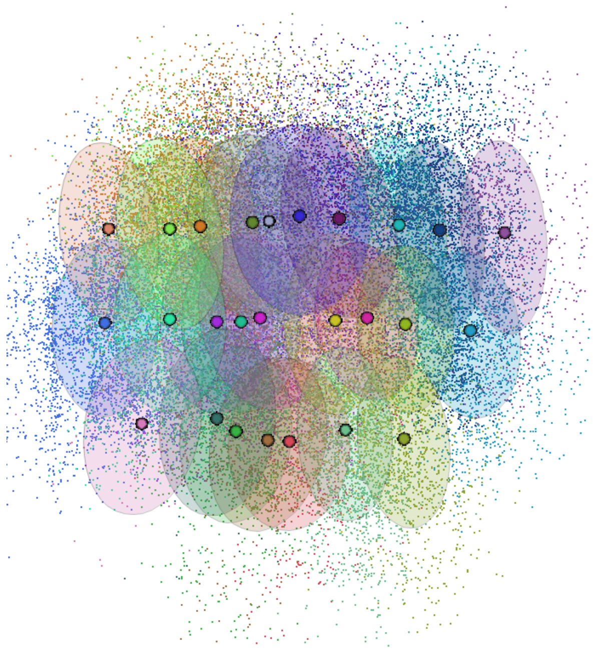


Fig. 2. All users' touch endpoints after a filtering standard deviation three times. The ellipses cover 50% of touch points corresponding to individual keys. Dots label the centroid of touch points for each key.

Fig. 2 illustrates the distribution of touch points collected from all 12 participants in this study. Touch points corresponding to individual keys are rendered in different colors. As well, we computed the centroids of touch points for each key and labeled them in the figure.

As shown, touch points yielded during eyes-free typing were quite noisy: Even if we look at the confidence areas of 50% coverage (the ellipses for individual keys), there was considerable overlap between them. This shows that without visual feedback, the accuracy of tapping keys solely based on spatial and muscle memory was significantly affected. On the other hand, a promising finding was that the centroids of touch points for individual keys formed a keyboard layout that was similar to a standard QWERTY layout. The only difference is that the emergent keyboard was taller than an ordinary one on mobile phones. This suggests that participants were able to transfer their spatial and muscle memory developed on visible QWERTY keyboards to eyes-free use.

Moreover, according to our observation, no more than five phrases were needed before most participants got familiar with eyes-free typing in the training session. All participants reported they could remember the QWERTY keyboard layout when typing eyes-freely. These results suggested that it was not difficult for participants to adapt their typing skill to eyes-free conditions.

In summary, both the distribution of touch points and the participants' subjective feedback supported the feasibility of eyes-free typing. Meanwhile, the lack of tapping accuracy as well as the deformation of keyboard layout pointed to the necessity of improved text decoding algorithms.

4 KEYBOARD MODEL FOR EYES-FREE TYPING

When typing on a visible keyboard, users control the movement of the tapping finger by referring to the location of required key that can be seen. However, in eyes-free typing, there is no keyboard model to refer to. Instead, users type on an "imaginary keyboard" on the touchscreen according to muscle memory and proprioception of the thumb. The location and the size of the "imaginary keyboard" is the comprehensive result of users' typing experience, size of the hand, size of the touchpad, etc.

In this section, we introduce the approach to fit a keyboard model from users' typing data, i.e. determining the spatial parameters of the "imaginary keyboard". We applied the approach to the data collected in Study 1. For clarity, we refer to the derived keyboard model as a **user keyboard**. Based on the user keyboard, we further investigated users' eyes-free typing behavior. The ultimate goal was to provide input to design effective text decoding algorithms for supporting eyes-free typing.

Before we introduce the approach utilized to derive a user keyboard, we first discuss two candidate mental models of eyes-free typing. The two models describe how the "imaginary keyboard" is represented in users' mind, and the strategy of how each key tapping action is performed.

4.1 Mental Models of Eyes-free Typing

We hypothesize two mental models of eyes-free typing (Fig. 3). The two models imply different ways to research users' eyes-free typing behavior and designs of the text decoding algorithm.

- **Absolute Model**: The user taps the required key only according to its absolute position on the keyboard. That is, the user executes individual tapping actions independently; there is no correlation between successive tapping actions. In text entry research [3, 8, 10], the absolute model is by default used to account for users' typing behavior on visible keyboards.
- **Relative Model**: The user taps the current key according to the tapping location of the previous key. For example, when inputting the word "open", it is likely that touch endpoint of 'p' always falls one key right to that of 'o', no matter where the user taps to input 'o'. Therefore, in the relative model, there is correlation between successive tapping actions. Its physical interpretation is that a user moves his/her thumb from the previous location to the current location to reflect the relative distance and direction between two keys on the keyboard. We propose the relative model to specially account for users' eyes-free typing behavior.

4.2 Deriving the User Keyboard from Typing Data

This subsection details how to derive the user keyboard (in both absolute and relative model) from users' typing data. Our assumption is that a user keyboard should have the same layout (arrangement of keys) as a standard QWERTY keyboard; however, the specific keyboard location on the touchscreen and the keyboard size (in both X- and Y-axes) should be further determined.

For the sake of convenience, we normalized the coordinate of a **standard keyboard** by defining the origin to be at the centroid of 'Q' and the key to be a 1×1 square (as shown in Fig. 4).

Supposing we collect n labeled pairs of touch endpoints and the corresponding keys, we thus denote coordinates of the observed touch endpoints as $\{(x_i, y_i)\}_{i=1}^{n}$, and coordinates of the corresponding keys on a standard keyboard as $\{(X_i, Y_i)\}_{i=1}^{n}$. We then apply a linear regression to model the relationship between the coordinates as follows

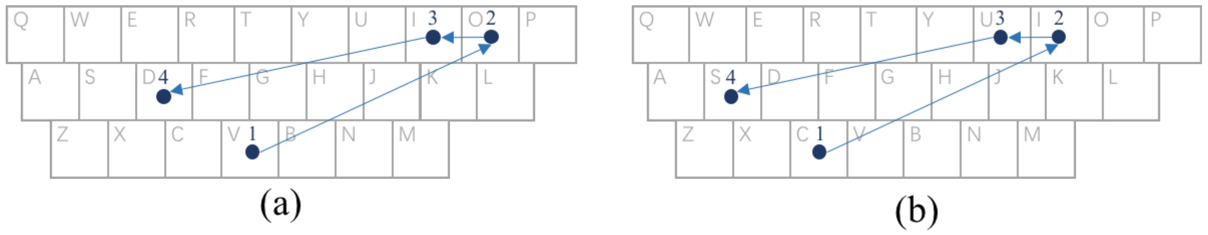


Fig. 3. This figure describes the concept of the two mental models for eyes-free typing. Take inputting “void" for example. (a) In the absolute model, each tapping action is programmed to hit the center of the required key. (b) In the relative model, a tapping action is programmed relative to the previous one. In this example, the first tapping has a 1-key offset to the desired key (“v"). Then all the following tappings also have a 1-key offset to their desired keys. Note that in the two models, the relative vectors between successive tappings are equivalent.

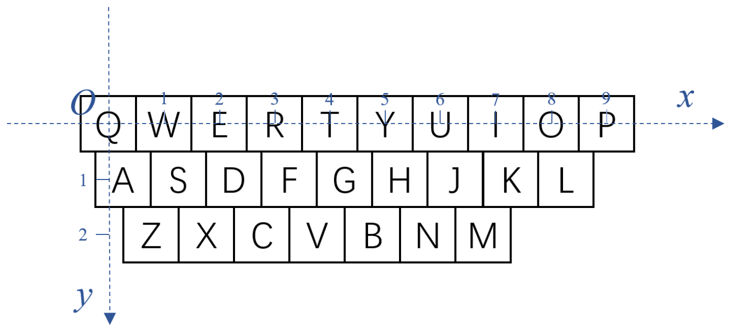


Fig. 4. The standard keyboard. Each key of the keyboard is a 1×1 square. The origin is at the centroid of the key ‘Q’. For example, the centric coordinates of ‘Q’, ‘P’, ‘A’, ‘L’, ‘Z’ and ‘M’ are (0, 0), (9, 0), (0.25, 1), (8.25, 1), (0.75, 2) and (6.75, 2), respectively.

$$\begin{cases} x_i = X_i * a_x + b_x \\ y_i = Y_i * a_y + b_y \end{cases} \tag{1}$$

The shape of a user keyboard (rectangle or arc-shaped) can be reflected by the goodness of fit, i.e. the R^2 value. $R^2 = 1$ means the user keyboard has an ideal rectangle shape, in that individual keys sit at the standard position on the keyboard. The higher the R^2 value is, the more similar the user keyboard is to a standard keyboard. In addition, a_x and a_y are interpreted as key size on X- and Y-axes respectively; (b_x, b_y) measures the keyboard location, defined as the offset between the centric coordinate of the key ‘Q’ and the upper left corner on the touchscreen. In total, we obtain four parameters a_x, a_y, b_x, b_y from the fitting and use them to characterize the user keyboard.

4.2.1 Deriving the user keyboard in absolute model. For the absolute model, we can directly perform the above linear regression to obtain the parameters of a user keyboard. We have

$$\begin{cases} x_i = X_i * size_x^{(abs)} + offset_x^{(abs)} \\ y_i = Y_i * size_y^{(abs)} + offset_y^{(abs)} \end{cases} \tag{2}$$

where $size_x^{(abs)}$ and $size_y^{(abs)}$ substitute a_x and a_y in Equation (1) as the key size respectively; $offset_x^{(abs)}$ and $offset_y^{(rel)}$ substitute b_x and b_y in Equation (1) as the offset of the keyboard location respectively.

4.2.2 Deriving the user keyboard in relative model. For the relative model, we can also fit a keyboard model by replacing (x_i, y_i) and (X_i, Y_i) with $(\Delta x_i, \Delta y_i)$ and $(\Delta X_i, \Delta Y_i)$ respectively for each $i \geq 2$

$$\begin{cases} \Delta x_i = \Delta X_i * size_x^{(rel)} + offset_x^{(rel)} \\ \Delta y_i = \Delta Y_i * size_y^{(rel)} + offset_y^{(rel)} \end{cases} \tag{3}$$

where Δ denotes the vector between successive touch endpoints or the corresponding standard key positions, which $\Delta x_i = x_i - x_{i-1}$ and $\Delta X_i = X_i - X_{i-1}$ for instance. Here, $size_x^{(rel)}$ and $size_y^{(rel)}$ measure the size of key in relative sense; $offset_x^{(rel)}$ and $offset_y^{(rel)}$ are interpreted as the distance between two successive touch endpoints performed on the same key. We hypothesize their expected value to be zero.

4.3 Fitting User Keyboard from Typing Data of User Study 1

We applied the linear regression described above to derive user keyboards from the data of Study 1. In addition to fitting user keyboards in the absolute and relative models, we also fit personalized keyboards and general keyboards: A personalized keyboard was fitted based on data of an individual participant; a general keyboard was fitted based on the pooled data of all 12 participants.

	Individual User				**Pooled User**			
Model	**Absolute**		**Relative**		**Absolute**		**Relative**	
Axis	**X**	**Y**	**X**	**Y**	**X**	**Y**	**X**	**Y**
Key Size (mm)	3.77 (0.57)	9.62 (2.04)	3.92 (0.60)	9.30 (1.98)	3.77	9.57	3.91	9.24
Offset (mm)	9.89 (3.86)	56.91 (6.49)	-0.17 (0.18)	0.07 (0.21)	9.90	57.10	-0.18	-0.08
R^2 value	0.963 (0.023)	0.994 (0.005)	0.986 (0.007)	0.999 (0.001)	0.982	0.998	0.992	0.999

Table 1. User keyboard parameters derived from users' actual typing data in Study 1. Standard deviations are given in parentheses. Note the offset has different meaning in the absolute model and in the relative model.

Results are summarized in Table 1. We next provide more detailed discussion on the results on the keyboard shape, key size and keyboard location, as well as the comparison between the absolute model and the relative model.

4.3.1 Keyboard Shape. In the absolute model, the mean R^2 value for individual users was 0.963 on X-axis and 0.994 on Y-axis. The R^2 value was even higher if we pooled all user data together, 0.982 on X-axis and 0.998 on Y-axis. In the relative model, the mean R^2 value was 0.986 on X-axis and 0.999 on Y-axis for individual users, and reached 0.992 on X-axis and 0.999 on Y-axis when merging all user data.

The above results showed that in both the absolute model and relative model, the shape of user keyboards was very similar to a standard keyboard. This quantitative measure again suggested that users could transfer their spatial memory of the keyboard layout from visible use to eyes-free. It also provided support for the rationality of our analysis using the fitted keyboard.

4.3.2 Key Size. For individual users, the mean key size on user keyboards was 3.77mm (SD=0.57, range: 3.05-5.04) on X-axis, and 9.62mm (SD=2.04, range: 6.42-12.56) on Y-axis. That is, in the eyes-free condition, the mean key size on Y-axis (height) was about 2.6 times of the mean key size on X-axis (width). Compared to a standard keyboard on a 4.3" phone screen where the key size is 4mm to 5mm on X-axis and 6mm on Y-axis, our results suggested that users were more likely to use a slightly narrower and much taller key in the eyes-free condition (see Fig. 6). On a general user keyboard fitted from pooled data, we have the same finding.

In addition, the standard deviation of the key size on Y-axis was 3.58 times of that on X-axis for individual users. Even if we divided the standard deviation by the mean height/width, key size on Y-axis was still 40.3% more variant than on X-axis. One possible explanation is that since there are significantly more keys on X-axis than on Y-axis (7-10 vs. 3), users' typing behavior was more constraint by the width of the touchscreen than the height. Meanwhile, the key height on user keyboards was more dependent on users' comfortable range for the thumb touch, which varied among individual users.

4.3.3 Keyboard Location. In the absolute model, we could derive the centric position of key 'Q' from the fitting offset, which determined the keyboard location. So that we could also estimate how far the user keyboard boundary (left, right, top and bottom) from the edge of the touchscreen by calculating with the keyboard location and key size.

The mean left and right boundary locations for individual users were 8.01mm (SD=4.09mm, range: 1.97-13.18) and 8.29mm (SD=3.35, range: 1.59-14.25) respectively. These results indicated that although our participants were all right-handed, they tended to tap around the middle area of the X-axis. Also the average standard deviation of left/right boundary location was as large as the size of key width (3.77mm).

The mean top and bottom boundary locations for individual users were 52.09mm (SD=7.03, range: 41.93-64.74) and 80.97mm (SD=6.07, range: 68.65-92.52) respectively. The average standard deviation of top/bottom boundary locations was larger than 60% of key height (9.62mm), which was still much larger than on X-axis due to the arbitrariness of holding the touchscreen on Y-axis.

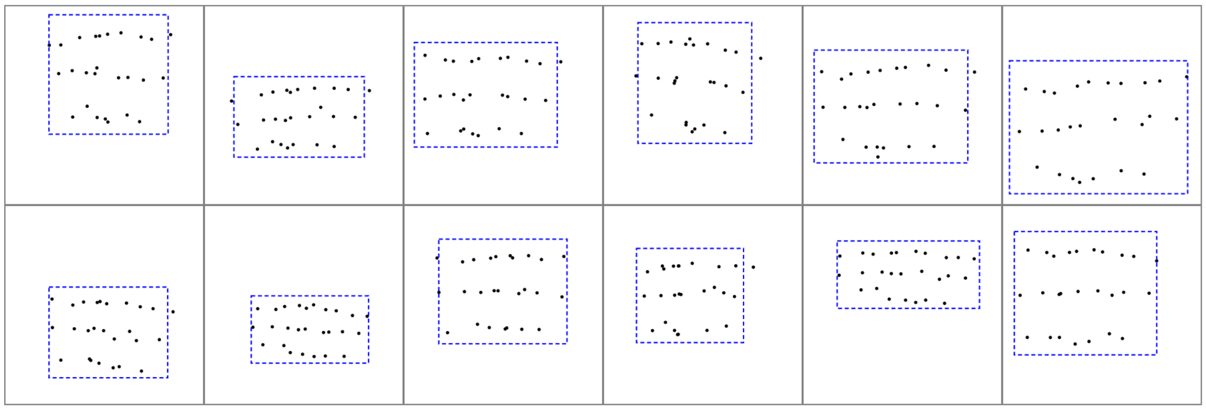


Fig. 5. Visualization of user keyboards for all 12 participants on the same area of the touchscreen. Blue dashed rectangles show the fitted keyboard boundaries in the absolute model. Black dots show the average key centroids of 26 letters from each user's endpoint data not fitted. Each keyboard was close to regularity but had different keyboard size and location.

From the above results and Fig. 5, we can see individual user keyboards were positioned on different locations on the touchscreen. This variance as well as that of the keyboard size, increases the difficulty for high-accuracy prediction of the text entry algorithm.

4.3.4 Endpoint Deviation. Endpoint deviation reflects the accuracy of user control of the finger movement or the stability of the user keyboard. It is an important measure that indicates the confidence of interpreting user input intention based on the observed touch endpoints.

In the absolute model, the mean standard deviation of endpoints (x, y) per key for individual users was 3.03mm (SD=0.43, range: 2.28-3.53) on X-axis, and 3.66mm (SD=1.17, range: 2.34-6.93) on Y-axis. In the relative model, the mean standard deviation of vectors $(\Delta x, \Delta y)$ per vector for individual users was 3.43mm (SD=0.42, range: 2.75-3.95) on X-axis, and 2.77mm (SD=0.42, range: 2.23-3.62) on Y-axis. That is, endpoint deviations on X- and Y-axis in the relative model were 13.2% higher and 24.3% lower than those in the absolute model respectively.

When pooling all user data, the mean endpoint deviations were 4.31mm on X-axis and 7.01mm on Y-axis in the absolute model, while the mean endpoint deviations of the vector were 4.52mm on X-axis and 3.80mm on Y-axis. In this instance, endpoint deviations on X- and Y-axes were 4.9% higher and 45.8% lower in the relative model than those in the absolute model. The reason for this is that merging all user data incorporates the difference of keyboard location between individual users into the endpoint deviation (Fig. 2). As shown previously, such keyboard location diversity cannot be ignored. Therefore, using the relative model which considers vectors will effectively reflect the characteristics of typing data on Y-axis in general case.

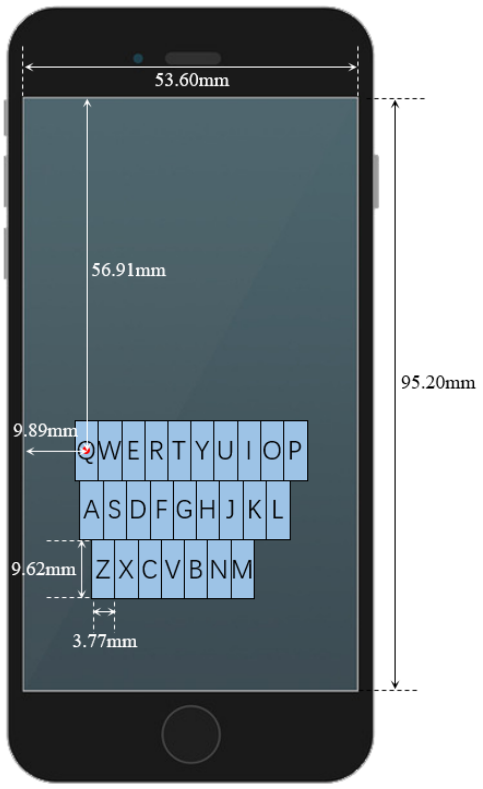


Fig. 6. The general user keyboard (in absolute model) fitted from pooled data of the 12 users in Study 1. The red dot represents the centroid of the key 'Q'.

4.3.5 Further comparison between the absolute model and relative model. Given two independent Gaussian variables with the same standard deviation σ, the sum or difference of the two variables should have a standard deviation of $\sqrt{2}\sigma$. Therefore, if the user completely follows an absolute model for eyes-free input, the standard deviation of the the vector ($SD(\Delta x)$ and $SD(\Delta y)$) between two successive endpoints should be be $\sqrt{2} \approx 1.41$ times the standard deviation of the absolute touch endpoint ($SD(x)$ and $SD(y)$), showing no correlation between endpoints. However, our data showed this was not true. The ratio of $SD(\Delta x)/SD(x)$ was 1.13 and 1.05 for individual and pooled user data respectively. On Y-axis, the result was even reversed. The ratio of $SD(\Delta y)/SD(y)$ was 0.76 and 0.54 for individual and pooled user data respectively. Therefore, there must be some degree of dependency between successive touch in eyes-free input. These results provide another element of support for the relative model.

5 THE ALGORITHM

In this section, we describe our word-level text decoding algorithm for eyes-free typing, which predicts the most probable word users attempt to type in the eyes-free condition. The algorithm was built based on the user keyboard derived from user typing data. We integrated the previous empirical results, including the size of key, keyboard location and endpoint deviation into the algorithm as parameters.

Let $I = \{(x_i, y_i)\}_{i=1}^{n}$ denote the sequence of touch input, $w = c_1c_2...c_n$ denotes a candidate word consisted of n characters belongs to a predefined set D (we only considered the word whose length is identical with the input sequence). The coordinate of the centroid of key c_i in the standard keyboard is (X_i, Y_i). The objective is to compute a word w^* with the highest probability using maximum a posteriori

$$w^* = \arg\max_{w \in D} p(w|I) \tag{4}$$

According to the Bayesian rule,

$$p(w|I) = p(I|w) \times p(w)/p(I) \tag{5}$$

where $p(w)$ represents the possibility of occurrence of word w in our unigram language model, and $p(I)$ is constant for all candidates. We rewrite $p(I|w)$ as

$$p(I|w) = p(\{(x_i, y_i)\}_{i=1}^{n} | \{c_i\}_{i=1}^{n}) \tag{6}$$

We then replace c_i by its 2D coordinate

$$p(I|w) = p(\{(x_i, y_i)\}_{i=1}^{n} | \{(X_i, Y_i)\}_{i=1}^{n}) \tag{7}$$

Further, we assume X-axis and Y-axis are independent

$$p(I|w) = p(\{x_i\}_{i=1}^{n} | \{X_i\}_{i=1}^{n}) \cdot p(\{y_i\}_{i=1}^{n} | \{Y_i\}_{i=1}^{n}) \tag{8}$$

5.1 The Absolute Algorithm

In the absolute model, individual tapping actions are independent to each other. We can then derive Equation (9) from Equation (8). It shares the same spirit with Goodman et al.'s method [11].

$$p(I|w) = \prod_{i=1}^{n} p(x_i|X_i) \cdot p(y_i|Y_i) \tag{9}$$

We use a normal distribution $N(\mu_i, \sigma_i)$ to calculate $p(x_i|X_i)$, where μ_i is the mean value of the coordinate of the key on X-axis, which can be estimated using key size, keyboard location and X_i by linear regression with Equation (2), and σ_i is standard deviation of the key from the result of Study 1 as well. The same way for $p(y_i|Y_i)$.

5.2 The Relative Algorithm

In the relative model, we use the vector between successive endpoints to predict the words. However, because the first endpoint does not have a predecessor, we still use the absolute model to interpret it:

$$p(I|w) = p(x_1|X_1) \cdot p(y_1|Y_1) \cdot \prod_{i=2}^{n} p(\Delta x_i|\Delta X_i) \cdot \prod_{i=2}^{n} p(\Delta y_i|\Delta Y_i) \tag{10}$$

where $\Delta x_i = x_i - x_{i-1}$, $\Delta y_i = y_i - y_{i-1}$, $\Delta X_i = X_i - X_{i-1}$ and $\Delta Y_i = Y_i - Y_{i-1}$. Note that for a word that has only one letter, the relative algorithm will degenerate into the absolute algorithm.

$p(\Delta x_i|\Delta X_i)$ and $p(\Delta y_i|\Delta Y_i)$ can be also calculated with the normal distribution with parameters from the linear regression (Equation (3)). Note the size and the offset of fitting as well as standard deviation of vectors are all in the relative condition.

5.3 Simulation of Performance

Before conducting a study involving real users, we ran a simulation to have a first look at the prediction power of the algorithms using data from Study 1, comparing the performance of different algorithms. When utilizing the data, key presses whose letters were subsequently backspaced were considered errors and removed. We simulated the performance of four algorithms:

- **General-Absolute (GA)**: The absolute algorithm with a general user keyboard. General keyboard was derived by fitting with parameters of pooled user data in Study 1. GA requires 6 parameters: key size, keyboard offset and key's standard deviation on both X- and Y-axes in the absolute model.
- **General-Relative (RA)**: The relative algorithm with a general user keyboard. RA also requires 6 parameters, which are all in the relative model.
- **Personalized-Absolute (PA)**: The absolute algorithm with a personalized user keyboard. Each user had his/her own personalized keyboard, which was derived by fitting with his/her own typing data. PA requires 6 parameters for each user.
- **Personalized-Relative (PR)**: The relative algorithm with a personalized user keyboard. PR also requires 6 parameters for each user.

User-dependency (general vs. personalized) not only affects the accuracy of the algorithm, but also has practical significance for interaction. For example, public display scenarios (like interacting with a smart TV) often involve multiple users. A user-dependent (personalized) algorithm requires training for each newly registered user, so a user-independent (general) algorithm seems more suitable for these scenarios. A user-dependent algorithm is considerable in scenarios which are more private (like interacting in VR environment or typing secretly on a personal smart phone).

For each algorithm, we performed a 10-fold cross validation. For the two general algorithms (GA and GR), we first pooled all data together and divided them into 10 folds; we then validated each fold with a trained user keyboard from the other 9 folds. Meanwhile, we guaranteed that in each fold, there was an equal amount of data sampled from each participant. For the two personalized algorithms (PA and PR), training and testing were done for each individual participant. In other words we divided each user's typing data into 10 folds and then validated each fold with a trained user keyboard from the other 9 folds. In the end, we averaged all user results.

We used the top 50,000 words in the ANC [1] with frequency data as the unigram language model. We simulated the Top-1, Top-5 and Top-25 accuracy for each tested algorithm, where Top-X means the target word is within the first X candidates ranked by the algorithm according to the probability.

Fig. 7 shows the results. The GA algorithm performed relatively worse than the other three algorithms in all Top-1, Top-5 and Top-25 conditions. As well, the GR algorithm was slightly worse (3.5%) than the two personalized

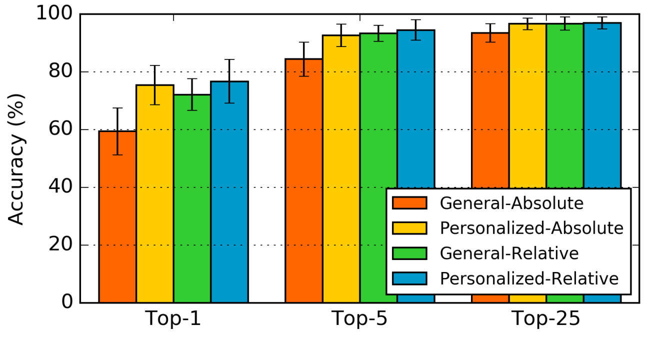


Fig. 7. Simulation of Top-1, Top-5 and Top-25 accuracies of different algorithms. Error bars represent standard deviations.

algorithms (PA and PR) in Top-1 condition. Their Top-5 and Top-25 accuracies were not distinguishable. Note that Top-1 accuracy is of great significance because it means users can input the word without a selection operation.

Now, we provide an explanation for the simulation results. For a general algorithm, a challenge is the variance of the key size and the keyboard location among individual users. In contrast, a personalized algorithm does not suffer from this problem, and naturally perform better. On the other hand, the GR algorithm predicts based on vectors rather than absolute locations. Thus, it eliminates the impact of variance of keyboard location. Fig. 8 illustrates the advantage of a relative algorithm over an absolute algorithm.

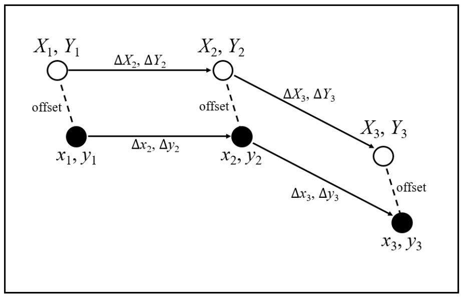


Fig. 8. An example of the relative algorithm being superior to the absolute algorithm. The user inputs three touch endpoints $\{(x_i, y_i)\}_{i=1}^3$ to enter a word of three letters. The actual positions of the three letters on the touchscreen are $\{(X_i, Y_i)\}_{i=1}^3$. There is a consistent offset between each pair of letter and touch endpoint in eyes-free condition. Each offset will decrease the predictive probability of the correct word in the absolute algorithm, but only one (the first offset) will affect the probability in the relative algorithm (vectors $(\Delta X_2, \Delta Y_2)$ and $(\Delta X_3, \Delta Y_3)$ are the same as $(\Delta x_2, \Delta y_2)$ and $(\Delta x_3, \Delta y_3)$, respectively, which will not bring the probability loss).

5.4 Impact of the Size of Training Data

To investigate how much training data is needed for achieve an acceptable accuracy, we conducted an additional simulation. We simulated Top-1, Top-5 and Top-25 accuracies of each algorithm by changing the size of training data, which was defined as the number of words. To do this, we first extracted 1/10 of the data as the validation data, and then randomly extracted different portions from the rest 9/10 of the data as the training data. Fig. 9

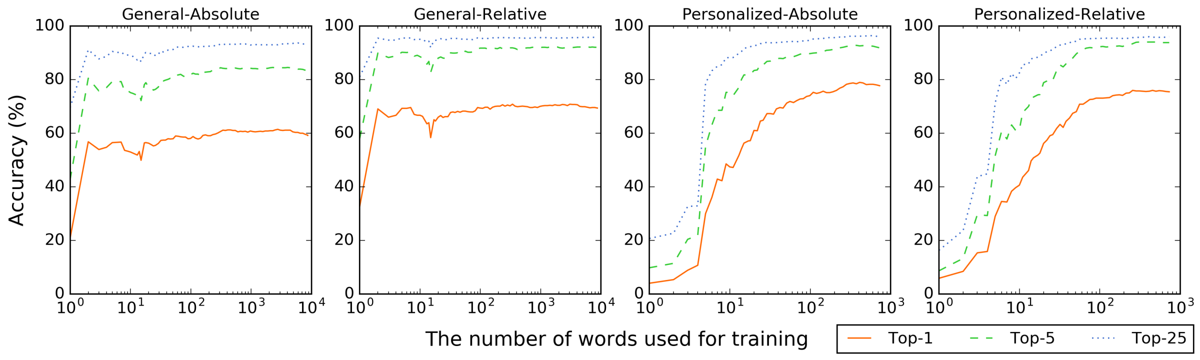


Fig. 9. Simulation of Top-1, Top-5 and Top-25 accuracies by the number of words used for training.

shows the result: For the two general algorithms, less than 10 words were needed to train a keyboard model with a Top-1 accuracy over 50%, and about 100 words to reach a convergence with an accepted accuracy. For the two personalized algorithms, more than 30 words were needed to achieve a keyboard model with a Top-1 accuracy over 50%, and the accuracy was still increasing after training data size exceeded 100.

6 USER STUDY 2: EVALUATING THE PERFORMANCE OF THE GENERAL ALGORITHM

In this study, we tested the performance of the General-Absolute (GA) algorithm and General-Relative (GR) algorithm in a user-independent setting. Parameters of the user keyboard we used were also derived from previous results. We used a smart TV as the interaction platform, since it is a public device which can be accessed by multiple users.

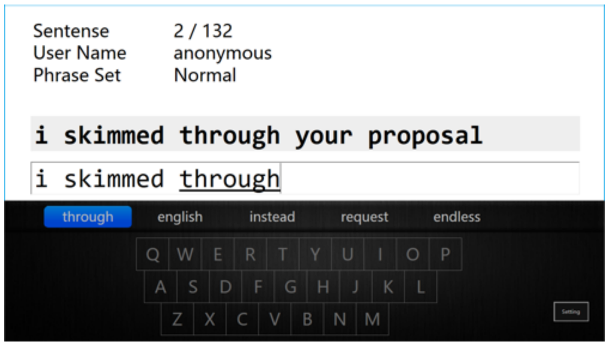
In addition, we implemented a **Baseline** technique for comparison: a user utilized the touchscreen to move a cursor on the TV screen and performed a single tap to select the key, which is a current text entry method on commercial smart TVs [5].

6.1 Apparatus and Participants

Another 16 skilled QWERTY keyboard users were recruited from the university campus (11 males and 5 females, aged 18-24). All of them were right-handed and could type fluently with one thumb. Participants were compensated 20 dollars for their time.

We used the same apparatus as in Study 1. Fig. 10 shows the software interface of the experimental system. The user was still required to type on the touchscreen, but he/she would see the real-time prediction on the TV this time. According to the prediction, the system displayed the Top-1 word at the end of the current input phrase, the Top-5 words below in probability order. We displayed a static image of a QWERTY keyboard on the interface for the participant to refer to in case he/she was not sure about the keyboard layout.

The user selected the Top-1 word with a quick right swipe gesture shorter than 500ms. Holding the thumb on the touchscreen for 500ms would trigger a panel containing an additional 20 words (25 in total) in lexicographical order. Meanwhile, the user entered the selecting mode and dragged on the touchscreen to select a Top-25 word which was highlighted along with the dragging, while releasing the thumb from the touchscreen to finish the selection. We determined through our pilot study that 500ms was a safe threshold. Also, the user swiped left to erase a letter, and swiped down to erase the word just entered.



Sentense 2 / 132
User Name anonymous
Phrase Set Normal

i skimmed through your proposal

i skimmed through

Fig. 10. The software interface of the experimental system in Study 2. A static keyboard image was displayed for users to refer to in case of forgetting or being unsure about key location. It was replaced by a panel with an additional 20 candidate words when the user triggered the selecting mode. A blue highlighted patch moved along with the dragging during the selecting mode, and the word in the highlighted patch would be selected after releasing the thumb from the touchscreen.

6.2 Design

We used a single-factor within-subjects design. The independent factor was *Technique* (GA, GR and Baseline). The tested phrases were randomly generated from Mackenzie's phrase set [19] for each technique independently, and were the same for all participants. For GA and GR, participants were required to type 40 phrases for each technique. For Baseline, participants typed 15 phrases, because according to our pilot study, the input speed of Baseline was about one-third of that for the two eyes-free techniques. For each technique, participants completed the phrases in one session, which was divided into 5 blocks. Each block contained 8 phrases for the two eyes-free techniques, or 3 phrases for Baseline. The presentation order of the three techniques was randomized for each participant, while the presentation order of the blocks was the same for each participant. In eyes-free condition, it was possible that some words could not be correctly entered with several trials. Therefore, we told the participants to give up and move on after a word had been attempted more than three times. This resulted in uncorrected errors in the final result.

6.3 Procedure

Before the experiment, we introduced the interaction method (eyes-free typing, functional gestures and selection method) and software for the participants, and asked them to familiarize themselves with the interface, input method and gestures. We did not tell them the difference between algorithms (GA and GR), or which algorithm was being used. We then asked them to type "as fast and accurate as possible". During the experiment, participants took part a short warm-up for each technique. They were forced to take a break between each block and fill out the NASA-TLX questionnaire after each technique. We also had the mechanisms as Study 1 to ensure participants performed typing leveraging their muscle memory only. Each session took 5-12 minutes, and the experiment took 20-30 minutes in total.

6.4 Results

6.4.1 Input Speed. RM-ANOVA found there was a significant effect of *Technique* on text entry speeds ($F_{2,30}$ = 306.82, $p < .0001$). The average input speeds for GA, GR and Baseline across blocks were 17.23 (SD=2.29) WPM, 20.97 (SD=3.08) WPM and 7.66 (SD=0.99) WPM, respectively. GA and GR were 1.25 and 1.74 times faster than Baseline, respectively. The difference between GA and GR was also significant ($F_{1,15} = 79.50, p < .0001$).

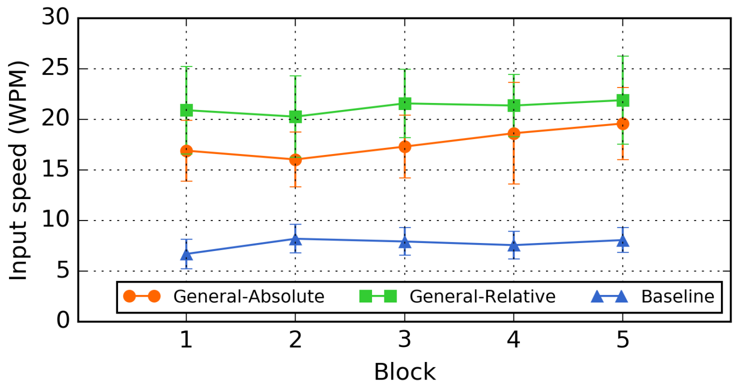


Fig. 11. The input speeds of GA, GR and Baseline across 5 blocks. Error bars represent standard deviations.

In addition, we observed a significant effect ($F_{4,60} = 3.44$, $p < .05$) of *Block* on text entry rate for GA. However, in a later study (Study 3), we found such an effect to be not significant. Currently, we cannot provide a compelling reason for this result. One potential reason might be that the phrases in blocks 3-5 for GA happened to contain less difficult words. This is because, due to the random sampling, individual blocks might contain phrases having different difficulty for eyes-free input. We acknowledge it as a limitation of our experimental design. This also suggested a more careful control of the phrase set for evaluation to be required.

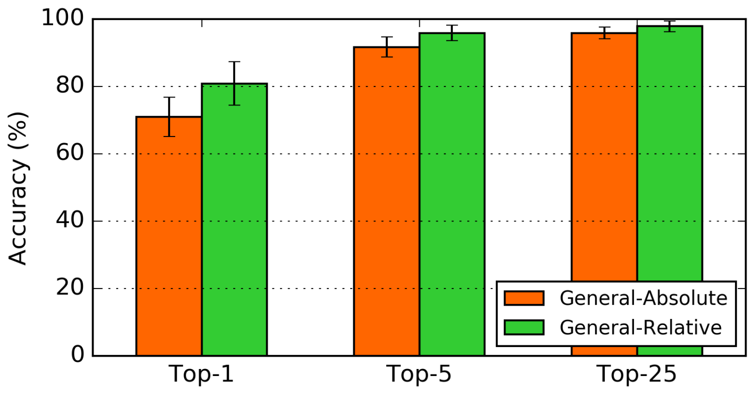


Fig. 12. Top-1, Top-5 and Top-25 accuracies of GA and GR. Error bars represent standard deviations.

6.4.2 Prediction Accuracy. As Baseline does not have input prediction ability, we only report results of GA and GR. Fig. 12 shows that GR outperformed GA in all three conditions. For GR, the Top-1, Top-5 and Top-25 accuracies were 80.9%, 95.9% and 97.9%, respectively. For GA, the Top-1, Top-5 and Top-25 accuracies were 71.0%, 91.7% and 95.9%, respectively. We found significant difference between GA and GR in terms of Top-1 ($F_{1,15} = 36.56$, $p < .0001$), Top-5 ($F_{1,15} = 27.35$, $p < .0001$) and Top-25 ($F_{1,15} = 20.04$, $p < .0005$) accuracy. And noticeably, the Top-1 accuracy of GR was 10% higher than GA. These results were consistent with simulation results (see Fig. 7).

6.4.3 Input Errors. We presented word error rate (WER) as the measurement, including corrected error rate and uncorrected error rate [30]. Consistent with previous researches [30], uncorrected error rates were low for all three techniques (GA: 1.69%; GR: 0.51%; Baseline: 0.21%). Corrected error rates for GA, GR and Baseline were 6.11%, 5.13% and 3.11%. Participants corrected 16% less errors with GR than GA. There were significant effects of *Technique* on corrected error rate ($F_{2,30} = 8.01, p < .005$) and uncorrected error rate ($F_{2,30} = 17.93, p < .0001$).

Post hoc analysis showed that there were significant differences between GA and GR on corrected error rate ($F_{1,15} = 17.68, p < .001$) and uncorrected error rate ($F_{1,15} = 14.50, p < .005$).

In addition, we defined a **prediction error** as one that occurred when the algorithm failed to predict the target word as the Top-K candidates. In our study, K was 25 according to the interface design. Results demonstrated a prediction error rate for GA of 4.09% and GR of 2.09%. That is, GR reduced prediction errors by half.

Finally, nearly all the tested words could be successfully entered with eyes-free input techniques. With GA, 9.20% of words were typed twice, 1.47% of words were typed three or more times. With GR, 7.17% of words were typed twice, 1.43% of words were typed three or more times. Words that participants **could not enter** (attempt to input for more than three times without success) accounted 0.3% of the total input words for GA and 0.06% for GR, and accounted for 17.7% and 11.8% of uncorrected errors for GA and GR, respectively.

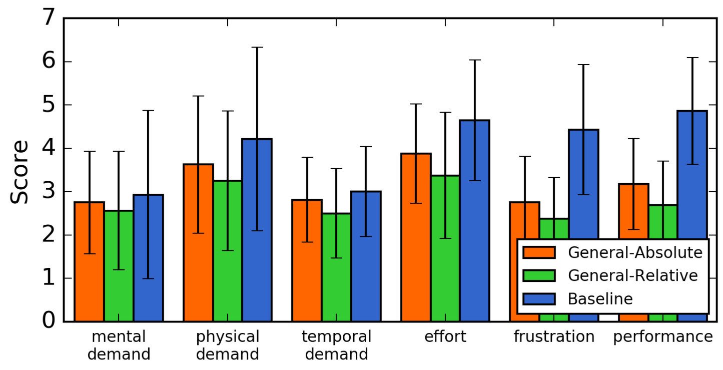


Fig. 13. Comparison of subjective feedback for GR, GA and Baseline. Error bars represent standard deviations. The higher the score, the more demanding a person believed the task to be.

6.4.4 Subjective Feedback. In terms of subjective feedback, the ratings for all dimensions were consistent. GR was the most preferred technique in terms of all dimensions, while Baseline was the least preferred. Friedman test showed that there was a main effect of *Technique* on physical demand ($\chi^2(2) = 11.8, p < .01$), effort ($\chi^2(2) = 8.05$, $p < .05$), frustration ($\chi^2(2) = 10.5, p < .01$) and performance ($\chi^2(2) = 19.4, p < .001$). No significant effects were found on mental ($p = 0.78$) and temporal ($p = 0.10$) demands.

For GA and GR, Wilcoxon signed-rank test showed that there was no significant difference on all dimensions except effort ($Z = -2.12, p < .05$).

In the post-experiment interview, we specially asked participants whether they look at the keyboard during typing. Participants reported that their visual attention was for the most time focused on the text feedback and candidate list. Two participants reported they occasionally referred to the keyboard when they were unsure of the keyboard layout.

7 USER STUDY 3: EVALUATING THE PERFORMANCE OF THE PERSONALIZED ALGORITHM

The main goal of this study was to examine whether a personalized keyboard could further improve eyes-free typing performance. We chose a head-mounted display (HMD) as the interaction platform. Moreover, the performance of current text entry techniques on HMD is still unsatisfying (only about 10 WPM [12, 32]).

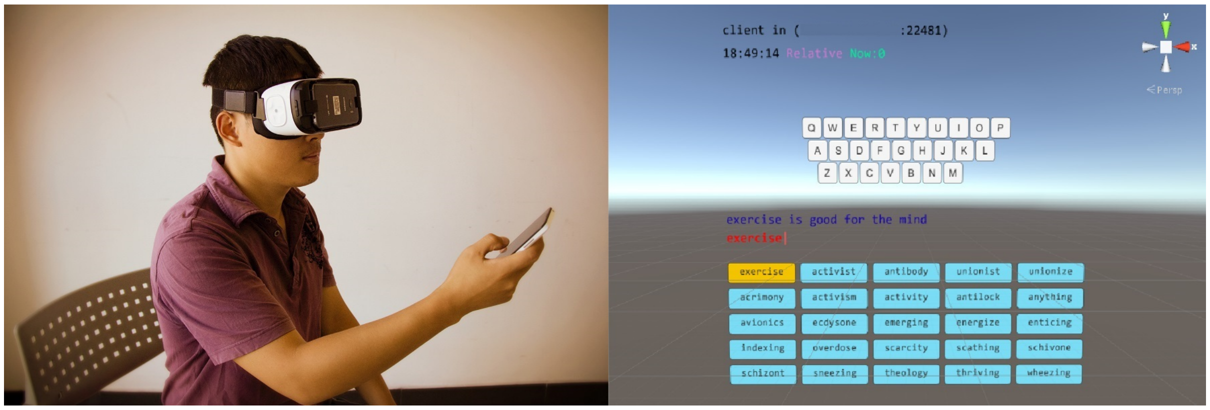

client in ([redacted] :22481)
18:49:14 Relative Now:0
Persp
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
exercise is good for the mind
exercise
exercise activist antibody unionist unionize
acrimony activism activity antilock anything
avionics ecdysone emerging energize enticing
indexing overdose scarcity scathing schivone
schizont sneezing theology thriving wheezing

Fig. 14. Experimental setting in Study 3: the user types on a head-mounted device (HMD) with a touchscreen (left), and the software interface (right).

7.1 Apparatus and Participants

We used the same Android phone of Study 1 as a remote control, and a Samsung Gear VR and a Samsung Galaxy Note 7 as the HMD. We developed the experimental software in Unity 5, which shares the same look-and-feel as that in Study 2. The remote control was connected to the HMD phone through Wi-Fi (Fig. 14).

Another 16 skilled right-handed QWERTY keyboard users were recruited from the university campus, 14 males and 2 females, aged 19-23. Half of them had experience with the HMD. Participants were compensated 10 dollars for their time.

7.2 Design and Procedure

We used a two-factor within-subjects design. The two independent factors were *Technique* (Absolute vs. Relative) and *Dependency* (General vs. Personalized).

The experiment had two sessions, each corresponding one algorithm with two dependencies, either the absolute (GA + PA) or the relative (GR + PR). The presentation order of sessions was counterbalanced. In each session, participants completed a total of 6 blocks, with each block containing 7 phrases randomly sampled from the Mackenzie and Soukoreff phrase set [19]. In this study, the random sampling of phrase set was done independently for each technique and for each participant. In the first 3 blocks, a user entered the phrases using the technique with a general keyboard that was the same as Study 2. We then trained his/her personalized keyboard using the data in these blocks. In the following 3 blocks, he/she entered text using the techniques with his/her personalized keyboard. To avoid potential subjective bias, we did not tell participants the model replacement during the task. A 3-minute break was enforced between blocks. Each session took 15-25 minutes, and the experiment took 30-50 minutes in total.

7.3 Result

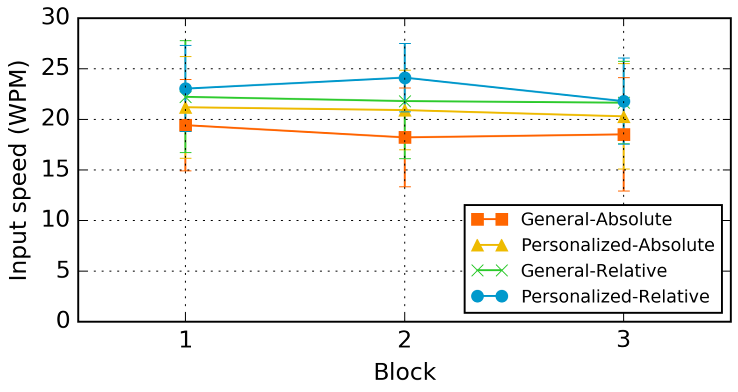


Fig. 15. The input speeds of all techniques over blocks. Error bars represent standard deviations.

7.3.1 Input Speed. Fig. 15 shows the input speed of participants for the four techniques across blocks. The average speed for GA, PA, GR and PR was 18.33 (SD=4.29) WPM, 20.59 (SD=4.18) WPM, 21.68 (SD=4.57) WPM and 22.77 (SD=3.50) WPM, respectively. Personalization improved input speed by 9.3% for the absolute algorithm and 5.0% for the relative algorithm. A significant effect of *Dependency* on input speed was found between GA and PA ($F_{1,15} = 10.21$, $p < .01$), but not found between GR and PR ($p = 0.28$).Moreover, the relative algorithm yielded a slightly higher text entry speed than the absolute algorithms. Significant effects of *Technique* were found between GA and GR ($F_{1,15} = 11.96$, $p < .005$), as well as between PA and PR ($F_{1,15} = 10.06$, $p < .01$).

In addition, we found no significant effects of *Block* on typing speed for all techniques (GA: $F_{2,30} = 0.61$, $p = 0.55$; PA: $F_{2,30} = 0.47$, $p = 0.63$; GR: $F_{2,30} = 0.18$, $p = 0.84$; PR: $F_{2,30} = 3.39$, $p = 0.06$). This suggested the learning cost was low, and users could adapt to eyes-free typing very quickly and reach a high typing speed without long-term training.

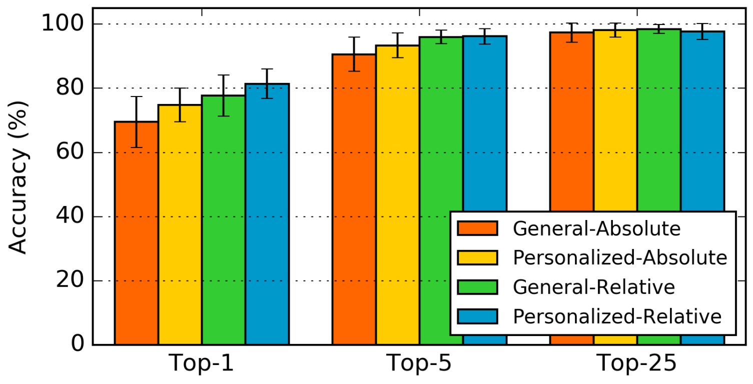


Fig. 16. Top-1, Top-5 and Top-25 accuracies of all techniques. Error bars represent standard deviations.

7.3.2 Prediction Accuracy. Fig. 16 shows the Top-1, Top-5 and Top-25 prediction accuracies of all the techniques, which was consistent with the simulation results in the previous section (see Fig. 7). Again, the relative algorithm outperformed the absolute algorithm for both general keyboard and personalized keyboard in terms of Top-1 ($F_{1,15} = 6.60, p < .05$ for general and $F_{1,15} = 18.96, p < .001$ for personalized) and Top-5 ($F_{1,15} = 12.17, p < .005$ for general and $F_{1,15} = 7.94, p < .05$ for personalized).

In line with the results of text entry speed, personalized was shown to increase the prediction accuracy for the absolute algorithm. A significant effect of *Dependency* was found between GA and PA on Top-1 ($F_{1,15} = 7.97$, $p < .05$). No significant difference was found on Top-5 and Top-25. In contrast, personalization seemed not to affect the prediction accuracy for the relative algorithm. No significant effects were found between GR and PR on Top-1, Top-5 or Top-25 accuracy.

7.3.3 Input Errors. We reported the same four word-level input errors as Study 2 (see Table 2): corrected error, uncorrected error, prediction error and ratio of words that could not enter. We found significant differences between GA and GR for prediction error ($F_{1,15} = 7.28, p < .05$) and ratio of words that could not enter ($F_{1,15} = 5.24$, $p < .05$). All other comparisons were not significant.

	Word Error Rate (%)			
	GA	**PA**	**GR**	**PR**
Corrected error	8.03 (4.24)	8.02 (4.71)	6.11 (4.90)	5.96 (4.07)
Uncorrected error	2.42 (2.76)	1.71 (2.01)	1.41 (1.34)	2.19 (2.42)
Prediction error	4.76 (2.88)	3.67 (2.28)	2.44 (2.40)	2.26 (1.90)
Could not enter	0.93 (1.40)	0.62 (0.81)	0.17 (0.37)	0.32 (0.43)

Table 2. Error rates of all tested techniques. Standard deviations are given in parentheses.

7.3.4 Subjective Feedback. In the post-experiment interview, we asked participants if they could identify the difference between the two techniques (absolute vs. relative) in the two sessions. 5/16 participants commented that they could perceive the relative technique was more accurate at predicting the target word as the Top-1 candidate, so that they did not have to perform further selection of candidate words. We also asked if they identified a switch of the technique during a session. None of the participants perceived his/her user keyboard had been changed from general to personalized in either absolute or relative session.

In addition, most (14/16) participants reported that it was surprising to see themselves be able to type without looking at the keyboard, and they would like to use our eyes-free technique for future HMD interaction.

8 LIMITATION

There are a number of limitations with the current research.

First, our keyboard model only considers global features such as the keyboard size, the keyboard offset and the average standard deviation of touch endpoints. Previous research [3, 8] has also studied key-based models, where the endpoint distribution for individual keys are explicitly modeled. We acknowledge that leveraging key-based model may further improve the prediction power of the algorithm, but may also require more training data compared to our approach.

Second, our current algorithm requires supervised training data to derive the parameters of the keyboard model. It thus cannot deal with sudden change of the keyboard model. For example, a user may reorient their hand grip on the phone. Therefore, it is important to further research adaptive algorithms that can dynamically update the keyboard model according to users' on-line input.

Third, we only test a unigram language model in this research. Probably, using a more powerful language model, such as n-gram, should further improve the performance. Meanwhile, it may also reduce the relative advantage of the personalized and relative touch models over the general and absolute models. This is also valuable to be explored in the future.

Finally, in Study 2 and Study 3, we displayed a static keyboard image on the screen for the participant to refer in case he/she was not sure about the keyboard layout. We did not specially study the impact of the keyboard image on the performance. However, enabling eyes-free typing with no keyboard image might have practical use, e.g. when the display area is limited. Therefore, this issue deserves to be investigated in the future.

9 CONCLUSION AND FUTURE WORK

In this paper, we followed an iterative approach to explore the possibility of eyes-free typing using one thumb. We first investigated users' eyes-free typing pattern and compared different mental models of eyes-free typing. We then described two algorithms: the absolute algorithm and the relative algorithm, both reflecting eyes-free typing patterns. We finally evaluated the actual performance of the two algorithm designs in both general and personalized conditions.

Through exploration, we obtained the answers to the research questions we raised at the beginning of this paper. First, users can indeed transfer their typing skills from normal condition to eyes-free use, after practicing only a few words. Second, the keyboard layout that emerged from users' eyes-free typing data was of a similar shape to an ordinary keyboard, but the tap variability was large. Additionally, to some degree, users followed a relative strategy during serial tapping. Third, the classical decoding algorithm (absolute algorithm) could parse the input relatively well; and our new algorithm, the relative algorithm, was even more accurate in inference and was preferred by most participants. The final result of this research is an eyes-free typing technique that allows immediate use. The obtained text entry rate was close to that reported for a visible keyboard [18], and faster than other eyes-free typing techniques [4, 26].

Future work has two directions: First, we need to examine the effect of touchpad size on users' eyes-free typing behavior and tapping accuracy. In our experiment, the width of the touchpad was 54mm; the average width of the emerging keyboard from users' touch endpoints was 37.7mm. It is important to know the limit of the smallest keyboard that users can perform eyes-free input with acceptable accuracy. Second, users may not have similar patterns in different contexts, including walking, riding, etc. We plan to apply our findings and techniques to other ubiquitous computing scenarios to examine if our models should be improved.

REFERENCES

[1] 2011. The Open American National Corpus (OANC). Website. (7 November 2011). http://www.anc.org/.

[2] 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

[3] 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

[4] Nikola Banovic, Koji Yatani, and Khai N Truong. 2013. Escape-keyboard: A sight-free one-handed text entry method for mobile touch-screen devices. *International Journal of Mobile Human Computer Interaction (IJMHCI)* 5, 3 (2013), 42–61.

[5] Aurora Barrero, David Melendi, Xabiel G Pañeda, Roberto García, and Sergio Cabrero. 2014. An empirical investigation into text input methods for interactive digital television applications. *International Journal of Human-Computer Interaction* 30, 4 (2014), 321–341.

[6] 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 *International Conference on Pervasive Computing*. Springer, 409–426.

[7] James Clawson, Kent Lyons, Thad Starner, and Edward Clarkson. 2005. The impacts of limited visual feedback on mobile text entry for the twiddler and mini-qwerty keyboards. In *Ninth IEEE International Symposium on Wearable Computers (ISWC'05)*. IEEE, 170–177.

[8] Leah Findlater, Jacob O. Wobbrock, and Daniel Wigdor. 2011. Typing on Flat Glass: Examining Ten-finger Expert Typing Patterns on Touch Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2453–2462. https://doi.org/10.1145/1978942.1979301

[9] Jérémie Gilliot, Géry Casiez, and Nicolas Roussel. 2014. Impact of Form Factors and Input Conditions on Absolute Indirect-touch Pointing Tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 723–732. https://doi.org/10.1145/2556288.2556997

[10] 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

[11] 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

[12] Tovi Grossman, Xiang Anthony Chen, and George Fitzmaurice. 2015. Typing on Glasses: Adapting Text Entry to Smart Eyewear. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '15)*. ACM, New York, NY, USA, 144–152. https://doi.org/10.1145/2785830.2785867

[13] Amy K Karlson and Benjamin B Bederson. 2007. ThumbSpace: generalized one-handed input for touchscreen-based mobile devices. In *IFIP Conference on Human-Computer Interaction*. Springer, 324–338.

[14] Per-Ola Kristensson and Shumin Zhai. 2004. SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04)*. ACM, New York, NY, USA, 43–52. https://doi.org/10.1145/1029632.1029640

[15] 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

[16] Kevin A. Li, Patrick Baudisch, and Ken Hinckley. 2008. Blindsight: Eyes-free Access to Mobile Phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 1389–1398. https://doi.org/10.1145/1357054.1357273

[17] Kent Lyons, Thad Starner, Daniel Plaisted, James Fusia, Amanda Lyons, Aaron Drew, and E. W. Looney. 2004. Twiddler Typing: One-handed Chording Text Entry for Mobile Phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 671–678. https://doi.org/10.1145/985692.985777

[18] I Scott MacKenzie, Mauricio H Lopez, and Steven Castelluci. 2009. Text entry with the Apple iPhone and the Nintendo Wii. *Proceedings of CHI2009* (2009).

[19] I. Scott MacKenzie and R. William Soukoreff. 2003. Phrase Sets for Evaluating Text Entry Techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems (CHI EA '03)*. ACM, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[20] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: A Mid-air Word-gesture Keyboard. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1073–1082. https://doi.org/10.1145/2556288.2556964

[21] Antti Oulasvirta, Sakari Tamminen, Virpi Roto, and Jaana Kuorelahti. 2005. Interaction in 4-second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 919–928. https://doi.org/10.1145/1054972.1055101

[22] Pekka Parhi, Amy K. Karlson, and Benjamin B. Bederson. 2006. Target Size Study for One-handed Thumb Use on Small Touchscreen Devices. In *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '06)*. ACM, New York, NY, USA, 203–210. https://doi.org/10.1145/1152215.1152260

[23] Daniel R. Rashid and Noah A. Smith. 2008. Relative Keyboard Input System. In *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI '08)*. ACM, New York, NY, USA, 397–400. https://doi.org/10.1145/1378773.1378839

[24] Oliver Schoenleben and Antti Oulasvirta. 2013. Sandwich Keyboard: Fast Ten-finger Typing on a Mobile Device with Adaptive Touch Sensing on the Back Side. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 175–178. https://doi.org/10.1145/2493190.2493233

[25] Caleb Southern, James Clawson, Brian Frey, Gregory Abowd, and Mario Romero. 2012. An Evaluation of BrailleTouch: Mobile Touchscreen Text Entry for the Visually Impaired. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 317–326. https://doi.org/10.1145/2371574.2371623

[26] Hussain Tinwala and I. Scott MacKenzie. 2010. Eyes-free Text Entry with Error Correction on Touchscreen Mobile Devices. In *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI '10)*. ACM, New York, NY, USA, 511–520. https://doi.org/10.1145/1868914.1868972

[27] Keith Vertanen, Haythem Memmi, and Per Ola Kristensson. 2013. The Feasibility of Eyes-free Touchscreen Keyboard Typing. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, Article 69, 2 pages. https://doi.org/10.1145/2513383.2513399

[28] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 657–666. https://doi.org/10.1145/1240624.1240727

[29] Yuntao Wang, Chun Yu, Jie Liu, and Yuanchun Shi. 2013. Understanding Performance of Eyes-free, Absolute Position Control on Touchable Mobile Phones. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 79–88. https://doi.org/10.1145/2493190.2493215

[30] 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

[31] Bo Yi, Xiang Cao, Morten Fjeld, and Shengdong Zhao. 2012. Exploring User Motivations for Eyes-free Interaction on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2789–2792. https://doi.org/10.1145/2207676.2208678

[32] Chun Yu, Ke Sun, Mingyuan Zhong, Xincheng Li, Peijun Zhao, and Yuanchun Shi. 2016. One-Dimensional Handwriting: Inputting Letters and Words on Smart Glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 71–82. https://doi.org/10.1145/2858036.2858542

[33] Shengdong Zhao, Pierre Dragicevic, Mark Chignell, Ravin Balakrishnan, and Patrick Baudisch. 2007. Earpod: Eyes-free Menu Selection Using Touch Input and Reactive Audio Feedback. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07)*. ACM, New York, NY, USA, 1395–1404. https://doi.org/10.1145/1240624.1240836

Received February 2017; revised April 2017; accepted April 2017