ResType: Invisible and Adaptive Tablet Keyboard Leveraging Resting Fingers

Zhuojun Li
li-zj19@mails.tsinghua.edu.cn
Department of Computer Science and Technology,
Tsinghua University
Beijing, China

Chun Yu*
chunyu@mail.tsinghua.edu.cn
Department of Computer Science and Technology,
Tsinghua University
Beijing, China

Yizheng Gu
guyz17@mails.tsinghua.edu.cn
Department of Computer Science and Technology,
Tsinghua University
Beijing, China

Yuanchun Shi
shiyc@tsinghua.edu.cn
Department of Computer Science and Technology,
Tsinghua University
Beijing, China
Qinghai University
Xining, Qinghai, China

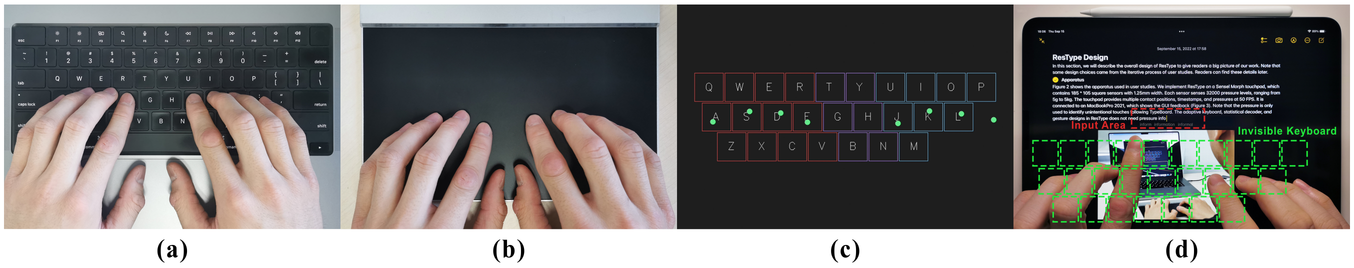


Figure 1: (a) Physical keyboard users rest their fingers on the home row keycaps to align fingers by tactile feedback. (b) ResType transfers this behavior to tablets (simulated by a Sensel Morph touchpad), allowing users to rest their fingers on the touchscreen. (c) ResType recognizes resting fingers on the home row and adapts the virtual keyboard accordingly. Users can then locate keys without visual attention and touch type efficiently. (d) ResType potential application: the user is composing a short paragraph based on the lower picture. In a traditional tablet, the soft keyboard will block the image. While using ResType, the user can touch type directly on the invisible keyboard and see the picture at the same time, without worrying about the occlusion problem.

ABSTRACT

Text entry on tablet touchscreens is a basic need nowadays. Tablet keyboards require visual attention for users to locate keys, thus not supporting efficient touch typing. They also take up a large proportion of screen space, which affects the access to information. To solve these problems, we propose ResType, an adaptive and invisible keyboard on three-state touch surfaces (e.g. tablets with unintentional touch prevention). ResType allows users to rest their hands on it and automatically adapts the keyboard to the resting fingers. Thus, users do not need visual attention to locate keys, which supports touch typing. We quantitatively explored users' resting finger patterns on ResType, based on which we proposed an augmented Bayesian decoding algorithm for ResType, with 96.3% top-1 and 99.0% top-3 accuracies. After a 5-day evaluation, ResType achieved 41.26 WPM, outperforming normal tablet keyboards by 13.5% and reaching 86.7% of physical keyboards. It solves the occlusion problem while maintaining comparable typing speed with current methods on visible tablet keyboards.

*indicates the corresponding author.

CCS CONCEPTS

• **Human-centered computing** → **Touch screens**; **Text input**.

KEYWORDS

adaptive keyboard, invisible keyboard, touch typing, text entry

ACM Reference Format:
Zhuojun Li, Chun Yu, Yizheng Gu, and Yuanchun Shi. 2023. ResType: Invisible and Adaptive Tablet Keyboard Leveraging Resting Fingers. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*

(CHI '23), April 23–28, 2023, Hamburg, Germany. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3544548.3581055

1 INTRODUCTION

Text entry is a basic task in tablet interaction - people type to write emails, surf the Internet, chat with friends, etc[13, 20, 35]. However, typing on tablets is inefficient compared with physical keyboards [13, 42, 46]. Firstly, tablet keyboard users constantly switch visual attention between the input area and the keyboard to locate keys, which affects input speed [42]. In comparison, physical keyboards provide tactile landmarks for users to align their fingers, thereby enabling efficient touch typing without visual attention [47]. Secondly, soft keyboards on tablets occupy large screen space. Users have to constantly fold and expand the keyboards to access information, which reduces typing performance in context-dependent text entry tasks, such as chatting or composing emails [45, 50]. These problems could be solved by an external physical keyboard, but it also increases the hardware weight and cost and is not convenient for mobile office. As a result, designing text entry methods on tablets that do not require visual attention and screen space is worthy of exploring.

In literature, two studies were conducted respectively to reduce tablet keyboard visual attention and occlusion. TOAST enables ten-finger touch typing on an invisible tablet keyboard [42]. But it adapts the keyboard to the touchpoint clouds of the 10 latest input words. This adaptive strategy requires a warm-up period for the decoding algorithm to reach the maximum accuracy, which cannot deal with fast and sudden hand movements (on a large surface or in multiuser scenarios). Sun et al. designed two low-occlusion keyboards by displaying only a few keys or lines [45]. The study shows users can access more information and perform less scrolling. It reduces distraction in two applications: chatting and taking notes, achieving a better interaction experience. However, users need to locate keys using the remaining landmarks, which still requires visual attention and occupies screen space.

In this paper, we propose ResType, a non-occlusion and adaptive keyboard that supports touch typing on three-state touch surfaces (i.e. surfaces that can distinguish intentional touching, resting and releasing states), like tablets that can prevent unintentional touches. ResType has a completely invisible keyboard, which does not need visual attention and solves the occlusion problem naturally. To support touch typing, ResType adapts the keyboard to users' hands, so users do not need to locate keys by visual attention. Anytime during touch typing, users can rest their hands on ResType, imagining their fingers on the home row of the virtual keyboard. This process is defined as **"calibration"** for convenience in this paper. Then, ResType recognizes the resting fingers and adapts the keyboard accordingly. It also uses an augmented Bayesian decoder to predict users' input, thus not requiring precise touch typing. Users only need to know the approximate key locations to touch type on ResType. This design leads to three research questions: **RQ1:** What is users' calibration behavior and how to detect it? **RQ2:** How to decode users' input with calibration? **RQ3:** What is the typing performance of ResType?

To support calibration, we first followed TypeBoard [20] to prevent unintentional touches, allowing users to rest their fingers on ResType without triggering unwanted responses. TypeBoard used contact data with diverse features (e.g. area, ellipticity, displacement, pressure, intensity) to train an SVM model. Each contact was labeled as intentional or not for the model to reject unintentional touches. ResType needs to recognize resting touchpoints to support calibration, so we leveraged the unintentional touch prevention feature of TypeBoard directly, and implemented ResType on the same hardware as TypeBoard (a Sensel Morph touchpad, Figure 1b). However, other core features of ResType (statistical decoding and adaptive keyboard) are independent of TypeBoard. In general, though not supported by most commercial tablet touchscreens currently (section 9), ResType can be implemented on any three-state touch surfaces. If a pratical tablet can disambiguate between resting and intentional touches as well, ResType will not need the external touchpad (Figure 1 (d)).

Then, we designed three user studies to answer the research questions respectively. To answer RQ1, we conducted study one to explore the statistical properties of calibration. Participants performed touch typing and calibration on ResType with no feedback, imagining the keyboard could adapt to resting fingers. We quantitatively analyzed the resting finger patterns and frequencies, based on which we developed calibration detection and keyboard fitting algorithms. In answering RQ2, we designed study two to collect users' typing data with calibration. Participants typed on ResType with asterisk feedback and calibration detection enabled. The typing data were used to fit the touch model under calibration and optimize the decoding algorithm. We achieved 96.3% top-1 and 99.0% top-3 accuracies at last (the top-N accuracy refers to the probability that a target word is within the most probable N words predicted by the decoder). For RQ3, we compared ResType performance with users' inherent typing ability and state-of-the-art methods (i.e. unintentional touch prevention and statistical decoding) on a fixed and visible keyboard through a 5-day within-subject evaluation. On day 5, ResType achieved 41.26 WPM, outperforming tablet keyboards by 13.5% and reaching 86.7% of physical keyboards. It solved the occlusion problem while maintaining comparable typing speed with current methods on visible tablet keyboards. We then designed a follow-up study to investigate users' gaze switching behavior on ResType. It turned out that ResType reduces gaze switching between the input area and the keyboard significantly compared with visible tablet keyboards. We also proposed several potential applications of ResType on tablets, dual-screen laptops, smart surfaces, and AR/VR, which is discussed in detail in section 8.5.

Our contribution in this paper is two-fold:

(1) We are the first to transfer calibration from physical keyboards to tablets with unintentional touch prevention or other three-state touch surfaces (i.e. users could rest their hands on the virtual keyboard to calibrate its position, similar to how they use physical keyboards). We also modeled the calibration behavior by quantitatively analyzing resting finger patterns and frequencies, based on which we proposed a new adaptive keyboard design that solved the visual attention and occlusion problems on tablets.
(2) We improved the augmented Bayesian decoding algorithm by introducing the touch model under calibration, which

achieved comparable typing speed with current methods on visible tablet keyboards.

2 RELATED WORK

We will review related work about adaptive keyboards, low-occlusion keyboards, and statistical decoding algorithms.

2.1 Adaptive Keyboard

Tablet soft keyboards are not restricted by the hardware, which can have a changeable layout, adapting to users' input languages, typing postures, hand positions, etc. Considering the language model, touchpoint distribution varies given the previous input sequence. Yin et al. and Gunawardana et al. changed key sizes in the keyboard for users to tap keys with higher frequencies more easily [22, 48]. Faraj et al. and Gkoumas et al. maintained the keyboard layout but enlarged or highlighted keys with higher probabilities dynamically for the next entry [3, 14]. Findlater et al. and Schoenleben et al. leveraged personalized touch models to decode users' input [12, 40]. Himberg et al. and Baldwin et al. further explored online personalization for key size and position adaptation [7, 23]. These strategies reduced the effort of locating keys, while still maintaining the overall keyboard position. Considering the typing posture, researchers found how users hold the device affects the touch model [6]. Based on this fact, ContextType and iGrasp changed the touch model or keyboard layout respectively to fit different typing postures [9, 16], which require posture sensing as a prerequisite [17]. Researchers also designed relative input systems, which assume the initial keyboard position is unknown and only use relative touchpoint positions to decode users' input [28, 31, 37]. These works allow users to type anywhere on the touchscreen, but lack the information of keyboard position, which leads to low decoding accuracy.

In literature, a more direct adaptive design is to make the keyboard adapt to users' hands [10, 24, 27, 39]. BrailleTouch used the combinations of 6 keys to input [39]. Hirche et al. placed 12 buttons under users' fingers and mapped each button with several characters to input [24]. LiquidKeyboard and TapBoard proposed a design that the keyboard being rearranged to fingers when users rest their hands on the tablet [10, 27] (the most similar to ResType). However, all the designs need unintentional touch prevention as a prerequisite, so that users' fingers could be placed on the tablet without unwanted responses. As a result, BrailleTouch and Hirche et al. used special gestures to avoid unintentional touches. LiquidKeyboard did not implement unintentional touch prevention and the adaptive keyboard. TapBoard used a threshold method to filter unintentional touches with an accuracy of 97%, but not implemented the adaptive keyboard either. TypeBoard [20] recognizes unintentional touches with higher accuracy (98.88%). We proposed a fully functional adaptive keyboard ResType leveraging this advantage. ResType adapts the keyboard to resting fingers after calibration, so that the keyboard position can be determined before typing, which also compensates for the artifact of relative models. Based on this design, users can type anywhere on the tablets with high decoding accuracy.

2.2 Low-Occlusion Keyboard

Tablet soft keyboards occupy large screen space, affecting the user experience of accessing information and typing [37, 45, 49]. Researchers adopted different strategies to minimize the keyboard screen space. BrailleTouch, H4-Writer, and Senorita introduced chorded typing methods to preserve only 2 to 6 keys [33, 36, 39, 44]. They encoded each character with a key combination (e.g. Huffman coding) for typing, which is hard to learn and leads to low input speed (14 to 23 WPM). Green et al., 1LineKeyboard and ThumbStroke reduced the learning burden by mapping the QWERTY layout to home row vertically [19, 30], or arranging the remaining keys in alphabetical order [29], which still changed the QWERTY layout. RearKeyboard, SandwichKeyboard, BackKeyboard, and Buschek et al. moved the keyboard to the rear of devices [4, 8, 26, 41]. This strategy successfully freed the screen space, but users need to type on the back with both hands from the left and right sides. It split the QWERTY layout and rotated it by 90 degrees to fit the finger orientations, which is still unnatural. Sun et al., Arif et al., NaviKey, and KlearKeys kept the entire QWERTY layout on the screen, while only displaying a few keys or transparentizing the key background [4, 11, 43, 45], which further reduced the learning burden, but not eliminated the occlusion completely.

Researchers further designed completely invisible keyboards while maintaining the entire QWERTY layout. Users could type on these systems by muscle memory and input decoding. BlindType and Zhu et al. developed invisible keyboards on smartphones [31, 50]. TOAST and Yoo et al. moved them to larger tablets [42, 49]. These works achieved both the lowest learning effect and keyboard occlusion by keeping the QWERTY layout and hiding the entire keyboard respectively. ResType is also based on these two strategies. The difference with prior work is that we proposed a new adaptive strategy and combine it with the decoding algorithm, thus achieving a higher decoding accuracy.

2.3 Statistical Decoding Algorithms

The Bayesian decoder [18] is widely used in text entry when it's difficult to locate keys precisely. It requires the probability distribution of input words (i.e. language model) and touchpoints (i.e. touch model), which are used to predict the most likely input words from a predefined vocabulary. The classical Bayesian decoder was first proposed by Goodman et al. [18] in 2002. It has been widely used in touchscreens, wearable devices, and ubiquitous keyboards (in-air or on any surface) [21, 38, 42].

For invisible keyboards, users cannot locate keys precisely, which brings greater variance in the touch model. A relative touch model was adopted to address this issue [31, 37, 42]. It was first proposed by Rashid et al. [37]. They assumed the keyboard position is completely unknown and used the relative model solely, which achieved only 48.5% top-3 accuracy. BlindType [31] tested the relative model on mobile phones using one thumb and reached 95.9% top-5 accuracy. TOAST [42] proposed an adaptive keyboard and used the absolute model for the first touchpoint in each word and the relative model for the other, which achieved 92% top-1 and 98% top-3 accuracies in the general-relative model (not personalized). It showed that the absolute model is useful when the keyboard position is adaptive to users.

In ResType, we combined the absolute and relative models and reached the highest prediction accuracy (96.3% top-1, 99.0% top-3, not personalized). The details of the ResType decoder are described in Section 6.

3 RESTYPE DESIGN

In this section, we will describe the overall design of ResType to give readers a big picture of our work. Note that some design choices came from the iterative process of user studies. Readers can find these details later.

3.1 Apparatus

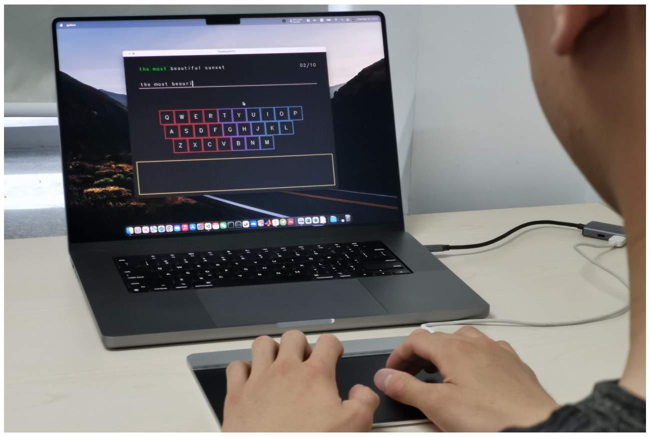


Figure 2: The apparatus used in user studies. Users type on ResType, which is implemented on a Sensel Morph touchpad. ResType is connected to a MacBookPro that shows the GUI feedback.

Figure 2 shows the apparatus used in user studies. We implement ResType on a Sensel Morph touchpad [2], which contains 185 × 105 square sensors with 1.25mm width. Each sensor senses 32000 pressure levels, ranging from 5g to 5kg. The touchpad provides multiple contact positions, timestamps, and pressures at 50 FPS. It is connected to a MacBookPro 2021, which shows the GUI feedback (Figure 3). Note that the pressure is only used to identify unintentional touches following TypeBoard [20]. The adaptive keyboard, statistical decoder, and gesture designs in ResType do not need pressure information.

3.2 Keyboard Layout and GUI Design

Figure 3 shows the keyboard layout and the general GUI design (small details are changed for different user studies). To benefit users who are not familiar with the standard touch typing method, we followed the split keyboard design as TOAST [42]. We set two virtual keyboards for two hands respectively, denoted as the "left-hand" and "right-hand" keyboards. We split 26 letters into "left, middle, and right" zones (colored with red, purple, and blue). The middle zone belongs to both keyboards. We used the same key size (19.0mm) as MacBookPro. A space key whose left, right and bottom borders are extended to infinity is placed a half key size below the

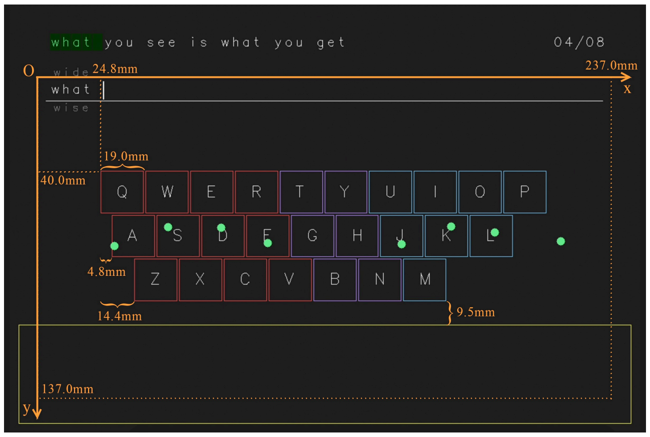


Figure 3: The keyboard layout and GUI of the final ResType. Users can only input 26 English letters and "space" in ResType. The orange annotations illustrate the keyboard range, which is linearly mapped with the touchpad sensing area and not shown to users.

third row [42]. We only considered keyboard translation, but not rotation and scaling (explained in section 8.3). Therefore, the two keyboard positions can be represented by fitting the "F" and "J" key positions.

The GUI shows the target phrase, the input area, and the keyboard layout from top to bottom. Input phrases are automatically switched to the next at the end of each one. We use green and red to indicate the correctness of input words. The keyboard layout in GUI is only useful for users who are not familiar with key positions (it was displayed for all users in studies to control this variable). Skilled touch typists could hide the layout to save screen space in real use. Besides, it shows the resting finger positions in calibration, which are moved to the home row by subtracting the offset between the actual and standard keyboard positions. It does NOT show any touchpoint positions during typing.

3.3 User Operation

Users can perform the following operations to input text using ResType:

(1) Calibration: users rest their hands on ResType while imagining their fingers are placed on the home row of the virtual keyboard. ResType updates keyboard positions to two hands immediately (with 100ms latency).
(2) Touch typing: when the keyboards are calibrated, users can touch type normally leveraging muscle memory from physical keyboards. Before finishing a word, the closest key to each touchpoint will be shown.
(3) Deletion: users can swipe left to delete a word (the continuous letter sequence at the end).
(4) Selection: ResType predicts the most likely three words when users input a complete word and tap the space key. ResType selects the top-1 candidate by default while the top-2 and top-3 candidates are shown upper and lower than it. Users

can change the selection by touching the touchpad and swiping up or down. The selected word will be displayed in real-time. And users confirm the selection by releasing their fingers from the touchpad.

ResType uses intentional touches for touch typing and filters out gestures mentioned above for calibration, deletion, and selection. Unintentional touches without any interactive intent will not cause any responses.

4 STUDY ONE: CALIBRATION BEHAVIOR

Understanding calibration behavior is the basis of ResType. To quantitatively analyze the resting finger patterns and the touchpoint geometric relationship within these patterns, we designed an imaginary text entry task to collect users' calibration behavior data. Based on the statistical properties of these data, we developed calibration detection and keyboard fitting algorithms to support the adaptive keyboard.

4.1 Design and Procedure

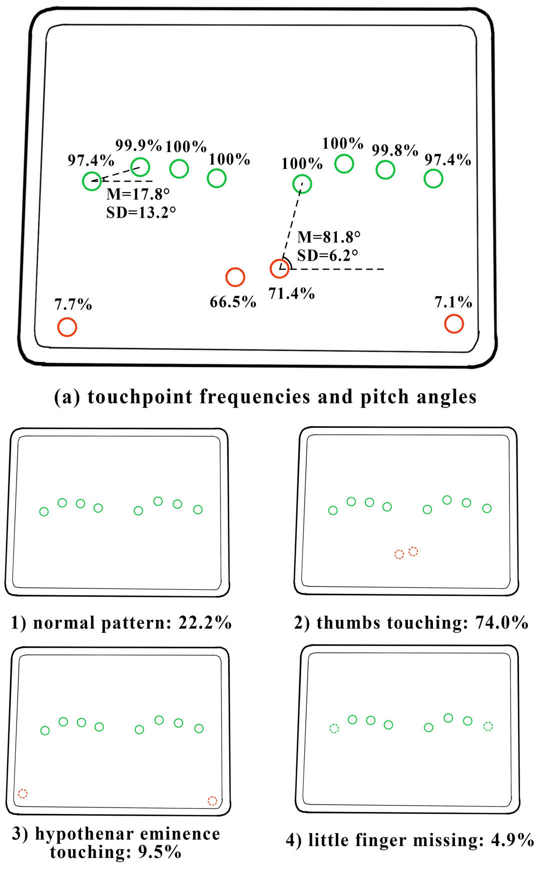
We recruited 16 participants from the campus (aged from 18 to 24, M = 21.63, SD = 1.63, 4 females). We explained all ResType features, especially unintentional touch prevention and adaptive keyboard design to them. In the imaginary task, users performed touch typing and calibration alternately. The system provided no feedback for any operations. The study for each user contained 5 blocks. Each block contained 15 phrases that were randomly sampled from the MacKenzie phrase set [32]. Each phrase was displayed for 8s, followed by 4s of black screen. Users were instructed to keep touch typing the phrases during displaying periods while performing calibration during hiding periods. This procedure basically simulated a normal touch typing and thinking scenario. All touchpad frames were recorded for analyzing the calibration patterns. We also set an RGB camera in the front of users' hands to record the fingertips as the ground truth. We further manually labeled each touchpoint in the hiding periods with its finger by checking the video. Each user spent 30 minutes and was paid $8 in the study.

4.2 Result

We collected 1329 times of calibration actions in total. All the touchpoints were generated by 10 fingertips and the hypothenar eminence. In Figure 4, all frequencies are calculated from the occurrence number of each touchpoint or pattern divided by the total number of calibrations. The calibration patterns include:

(1) Normal pattern: eight fingertips (except two thumbs) on the home row (i.e. the "ASDFJKL;" keys).
(2) Thumb touching: two thumbs resting on the touchpad.
(3) Hypothenar eminence touching: the hypothenar eminence causing unintentional touches.
(4) Little finger missing: users' little fingers not on the keyboard.
(5) The combination of the above patterns.

Since patterns b2, b3, and b4 are not independent with each other (only b1 is), the sum of pattern frequencies exceeds 100%. And the combined frequency of b2, b3, and b4 (at least two of them occur at the same time) is 10.2%.



97.4%
99.9%
100%
100%
M=17.8°
SD=13.2°
100%
100%
99.8%
97.4%
M=81.8°
SD=6.2°
7.7%
66.5%
71.4%
7.1%
(a) touchpoint frequencies and pitch angles
1) normal pattern: 22.2%
2) thumbs touching: 74.0%
3) hypothenar eminence touching: 9.5%
4) little finger missing: 4.9%

(b) pattern frequencies

Figure 4: The frequencies of calibration touchpoints and patterns. Green touchpoints are on the home row while red ones are not. Touchpoints with dotted lines are optional. The pitch angle distribution demoed in (a) is universal for all touchpoints.

4.3 Adaptive Keyboard

In ResType, the left-hand and right-hand keyboards follow users' hands after calibration. To achieve this, we first designed algorithms to filter out touchpoints on the home row based on the calibration patterns, then used these touchpoints to fit the keyboard position.

From users' calibration behavior, we have several observations: **OB1:** the index finger, middle finger, and ring finger of two hands touch the keyboard in most cases (>99.8%). **OB2:** the fingers of each hand on the home row form a flat trapezoid. We statistically explored the geometric relationship of the touchpoints and found that the pitch angle of neighboring touchpoints within the home row (in degrees, M = 17.8, SD = 13.2, MIN = 0.0, MAX = 62.0, see Figure 4 (a)) are significantly smaller than the pitch angle between an unintentional touchpoint and its adjacent ones on the home row (in degrees, M = 81.8, SD = 6.2, MIN = 60.5, MAX = 90.0). **OB3:** unintentional touches from thumbs and thenar eminence are relatively far away from the home row (minimum distance in cm within home row: M = 2.26, SD = 0.46, between home row and others: M = 5.15, SD = 1.14, both of them follow normal distributions).

Inspired by these observations, we designed our finger detection algorithm to identify fingers on the home row:

(1) For all horizontal adjacent touchpoint pairs, if the pitch angle between the two points exceeds 60.0 degrees, remove the lower one (OB2. We chose 60.0 degrees to reject all unintentional touches while recognizing 99.8% calibration), which filters out thumb and thenar eminence touches.
(2) For each touchpoint, find the closest touchpoint to it. Keep 8 touchpoints at most with the smallest distance (OB3), which filters out touches that are far away from the home row.
(3) Now only 6 to 8 touchpoints will remain. We split these touchpoints from the larger gap in the middle. Then the touchpoints of the left and right hands are found.

To prevent accidental patterns that happen to pass the algorithm (e.g. wiping the keyboard with hand palms), we set a 100ms smoothing window (i.e. 5 frames). Only when all resting touchpoints in the window have appropriately the same positions, will they be regarded as a valid calibration. Therefore, the calibration latency ranges from 100ms to 120ms.

To fit the left-hand and right-hand keyboards based on the resting fingers, we adopted the following strategy: the x coordinate of F or J was set to the x value of the index finger, while the y coordinate was set to the average y values of all fingers in each hand. While touch typing, we separate the left-hand and right-hand touchpoints by the distance to the fitted F and J keys. Then we subtract the offset between the standard and fitted F or J positions from the actual touchpoints to find the positions on the standard keyboard layout.

5 STUDY TWO: DECODER DESIGN

In this section, we first introduced ResTypeStatic for comparison with ResType. Since users' touchpoint distribution may be different, we collected users' typing data on ResType and ResTypeStatic respectively, based on which we finalized users' touch model and optimized the statistical decoding algorithms on the two keyboards.

5.1 ResTypeStatic Keyboard

To compare the typing performance between ResType and state-of-the-art tablet keyboards with a visible and fixed layout, we introduced **ResTypeStatic**, as shown in Figure 5. Our goal was to only investigate the influence of calibration and adaptive keyboard layout on typing, while excluding other factors of ResType. ResTypeStatic has exactly the same features (unintentional touch prevention, statistical decoding, gesture operations, and GUI design) as ResType, except that it has a fixed and visible layout and does not support calibration. We used the same touchpad for ResTypeStatic and drew the keyboard layout using white marker pens. To our knowledge, keyboards that both support unintentional touch prevention and statistical decoding have not been explored yet.

5.2 Design and Procedure

We recruited 16 participants from the campus (aged from 19 to 21, M = 20.38, SD = 0.62, 2 females). They had never used ResType before. We designed 4 text entry blocks for each user. In each block,

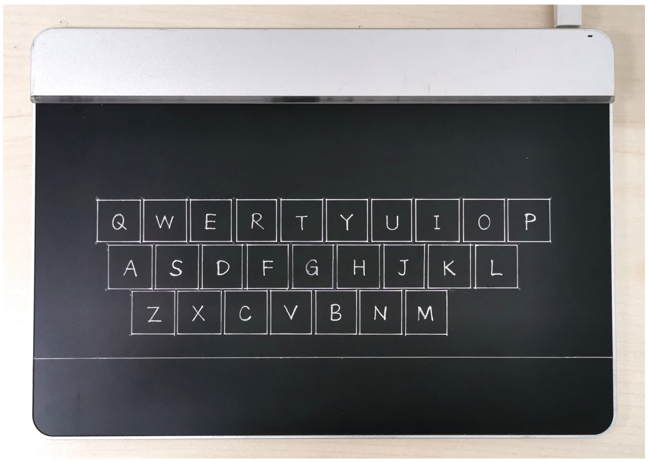


Figure 5: The ResTypeStatic keyboard layout is drawn on the Sensel Morph touchpad, whose layout is the same as the initial keyboard layout in ResType before calibration. The area below the bottom line is the space key.

users transcribed 50 phrases that were randomly sampled from the MacKenzie phrase set [32]. They first took a warm-up block with 10 phrases to get familiar with the operations and were forced to have a 2-minute break between each block to relieve fatigue.

In this study, we implemented the same unintentional touch prevention as TypeBoard [20]. We also implemented calibration detection based on study one to collect calibrated typing data on ResType. During typing, we only provided asterisk feedback (an asterisk would appear in the input area each time users triggered an intentional touch) to users. Since users might be cautious about unintentional touches using touchscreens [20], we hid each phrase in GUI by default. Users needed to perform calibration first to reveal the phrase before typing. This design encouraged users to calibrate and generated calibration at least once for each phrase. They were instructed to type as fast and as accurately as possible. We also required users to delete incorrect words if they found a mistake by themself. Each user spent an hour and was paid $15 in the study.

Then we collected the typing data on ResTypeStatic. We found users' touchpoints were significantly more concentrated on ResTypeStatic by pilot studies, so fewer data were needed to determine the touch model. We recruited 8 users (aged from 20 to 25, M = 21.25, SD = 1.91, all male) and designed 1 warm-up (10 phrases) and 1 formal block (50 phrases) for each user. All procedures are the same as on ResType.

5.3 Result

To simplify the problem, we only focus on the touchpoint distribution of the 26 letters. We finally collected 76563 touchpoints on ResType and pooled them together from all users. The touchpoint cloud of each key appropriately follows a Gaussian distribution. As shown in Figure 6, the touchpoints are considerably more concentrated on ResTypeStatic than on ResType, because users could see the visible layout during typing to locate fingers more precisely on ResTypeStatic [13]. Figure 6 (b) shows the uncalibrated clouds on

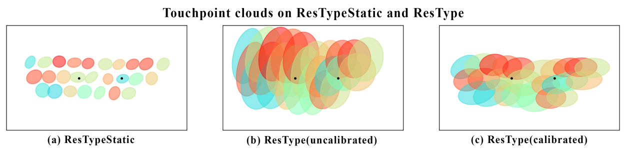


Figure 6: The 95% confidence ellipses of the touchpoint clouds on ResTypeStatic and ResType. Warmer colors indicate higher key frequencies. The two black points on each subplot represent the standard F and J centroids.

ResType, while the clouds in Figure 6 (c) are calibrated. One-way ANOVA shows both the standard deviations on x and y axis for all keys in (a) are significantly smaller than the clouds in (b) (x: $F_{1,50} = 390.27, p < 0.001$, y: $F_{1,50} = 252.62, p < 0.001$). While only the standard deviations of y axis in (c) are significantly smaller than the clouds in (b) ($F_{1,50} = 282.64, p < 0.001$). The standard deviations of x axis do not change significantly by calibration. The result shows that calibration could help users locate keys more precisely on ResType, which proves the effectiveness of the adaptive strategy.

6 RESTYPE DECODER

We adopted the augmented Bayesian method to design the ResType decoder. Classical Bayesian decoder uses a pre-trained touch model and language model to decode users' input. The touch model regards users' input as a sequence of independent touchpoints, named as the absolute model in literature [31, 42]. Researchers introduced the relative model to better decode the correlation in the input sequence. It was also used in situations when the initial keyboard position is unknown [21, 31, 37, 42]. Inspired by these algorithms, we brought the adaptive keyboard design into the touch model and developed our augmented Bayesian decoder.

6.1 Classical Bayesian Decoder

The input decoder is used to predict the input word W given the input sequence I. The classical Bayesian approach was first introduced by Goodman [18]. It predicts the most likely word in a predefined dictionary as follows:

$$P(W|I) = \frac{P(I|W) \times P(W)}{P(I)} \propto P(I|W) \times P(W) \tag{1}$$

where $P(W)$ is the language model, which describes the probability of the word W. And $P(I|W)$ is the touch model, which uses Gaussian distribution to estimate the probability of the input sequence I given the word W. For the kth input word, researchers use the n-gram language model to calculate the conditional probability:

$$P(W) = P(W_k|W_{k-n+1}...W_{k-1}) \tag{2}$$

For the input touchpoint sequence $I = i_1 i_2 ... i_n$ and a word with the same length $W = w_1 w_2 ... w_n$, classical methods treats each touchpoint as independent to each other:

$$P(I|W) = \prod_{k=1}^{n} P(i_k|w_k) \tag{3}$$

This method requires a touch model for each key and the absolute keyboard position.

6.2 Relative Touch Model

The biggest drawback of the absolute touch model is that it ignores the correlation between successive touchpoints, also known as the reference effect in literature [45]. For example, when inputting the same letter in a word like "success", the touchpoint of the second "c" would be closer to the first one, since users do not need to locate the same key twice. Thus, the conditional distribution of the second "c" will not obey the general distribution in the absolute model. The relative model uses the relative position in successive touchpoints as follows:

$$P(I|W) = \prod_{k=2}^{n} P(i_k - i_{k-1}|w_{k-1}, w_k) \tag{4}$$

This model calculates the conditional probability of a touchpoint only by the offset to the previous touchpoint. It will be invalid if the input length equals 1 and cannot deal with word collisions like "hit" and "joy" (their relative positions on the QWERTY layout are the same).

6.3 ResType Decoder Design

We used the bigram language model in ResType [25]. It was replaced by the unigram model for the first word in a phrase. The dictionary size is 10000.

For the touch model, we propose our hybrid decoding method in ResType by combining the advantages of absolute and relative models. We assume the touchpoints of each key follow Gaussian distribution. We need 5 parameters to describe the touch model for each key: the means and standard deviations of x and y, and their correlation coefficient. As a result, we totally need $26 \times 5 = 130$ parameters for the absolute model and $26 \times 26 \times 5 = 3380$ parameters for the relative model. We calculated these parameters directly from the typing data in study two.

In ResType, we define two 2-dimensional keyboard spaces, the actual space, and the standard space. The actual space refers to the actual touchpad sensing area while the standard space refers to the standard QWERTY layout. All touchpoints will be calibrated as relative to the standard space by subtracting their offset. We fit the left-hand and right-hand keyboards by the centroids of "F" and "J" keys in the actual space, denoted as $p_{a,F}$ and $p_{a,J}$. While their standard positions are $p_{s,F}$ and $p_{s,J}$. For each touchpoint in the actual space p_a, we first determine its corresponding hand by the distances to $p_{a,F}$ and $p_{a,J}$ (touchpoints closer to $p_{a,F}$ will be fit to the left-hand keyboard, similar for $p_{a,J}$). And calculate the offset to the standard space $o = p_{a,F} - p_{s,F}$ or $o = p_{a,J} - p_{s,J}$.

Based on these two spaces, the touch model is described by:

$$P(I|W) = P(i_1|w_1) + \prod_{k=2}^{n} [\lambda P(i_k|w_k) + (1-\lambda) P(i_k - i_{k-1}|w_k, w_{k-1})],$$
$$(i_k = p_{s,k} = p_{a,k} - o_k) \tag{5}$$

where we introduced λ (a constant in $[0, 1]$) as a weight parameter to combine the absolute and relative models. In the simulation, we found the hybrid model has better performance than either the pure absolute or relative model.

The ResTypeStatic decoding algorithm is the same as ResType, except that we use a different touch model and do not have the calibration operation. In other words, the actual and standard spaces on ResTypeStatic are the same.

6.4 Result

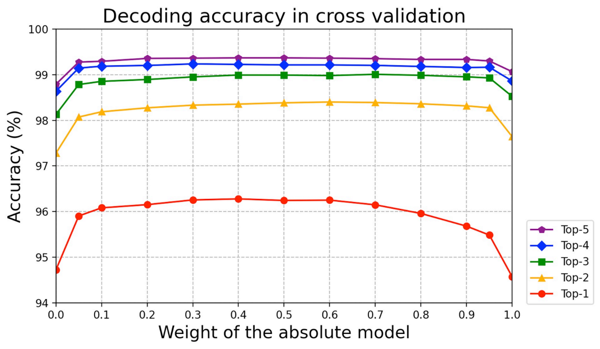


Figure 7: The top-1 to top-5 word-level decoding accuracy in 16-fold cross-validation. The top-1 accuracies of the pure absolute and relative model are 94.6% and 94.7%. $\lambda = 0.4$ reaches the highest top-1, top-3 and top-5 accuracies of 96.3%, 99.0% and 99.4%.

We implemented the ResType decoding algorithm and tested its performance on the typing data in study two by simulation. Since ResType keyboards are not either fixed or entirely unknown to users, we expected that combining the absolute and relative models could achieve better performance than the related work. In the simulation, we focused on the influence of λ (equation 6.3) on these two models. We performed 16-fold cross-validation on different λ, meaning that for each participant, we generated the touch models from the other 15 participants.

Results in Figure 7 illustrate top-1 to top-5 accuracies over λ. Both the absolute and relative models achieved satisfying accuracy. Furthermore, when combining these two models together, the accuracy increased notably (1.6% to relative and 1.7% to absolute), which was consistent with our expectation as well as the adaptive feature of ResType. We set λ to 0.4 at last, which has the highest top-1 accuracy (96.3%). We also set the candidate word number to 3, considering it is easier to select and that the top-3 accuracy (99.0%) is significantly higher than top-1, but top-5 (99.4%) is not than top-3.

We performed the same optimization for ResTypeStatic (subsection 5.1), which achieved 99.2% top-1 accuracy and 99.8% top-3 accuracy. Its decoding accuracy is higher than ResType because users can see the layout on ResTypeStatic, thus locating fingers more precisely. It is consistent with the concentrated ResTypeStatic touchpoint clouds.

6.5 Summary

By now, we have fully implemented the ResType decoder. It is beneficial to summarize how the decoder works in real use. We first collected users' typing data on ResType and ResTypeStatic to finalize the absolute and relative touch models (Section 5). Then we proposed a hybrid decoding algorithm to fuse the touch models and the adaptive keyboard with calibration (Section 6). When users touch type on ResType, they first perform calibration, after which the offset between the actual and standard spaces is determined. Then, users could touch type a letter sequence in a word. Only when users finish a word and tap the space key, will the ResType decoder use the touchpoint sequence to calculate the probability of each word in the dictionary, and returned top-3 candidates for users to confirm. By default, the top-1 candidate will be selected, and users could change the input by selection or deletion in subsection 3.3.

7 STUDY THREE: EVALUATION

In this section, we designed a 5-day user study to evaluate the typing performance and learning effect of ResType.

7.1 Design and Procedure

The study adopted a within-subject design to compare the performance of ResType with ResTypeStatic. We also measured users' typing speed on physical and tablet keyboards to indicate their typing ability. We recruited 14 participants from the campus (aged from 18 to 30, M = 21.4, SD = 2.9, 5 females). Their typing experience and touch typing levels were first collected through a questionnaire (Table 1). The result showed that all of them are familiar with physical and tablet keyboards (at least one cyear of using experience).

On each day, we designed 12 text entry blocks (10 phrases for each) - 1 for physical keyboards, 1 for tablet keyboards, 5 for ResTypeStatic, and 5 for ResType. We had fewer blocks for physical and tablet keyboards because users are already familiar with these techniques (i.e. no learning effect). All testing phrases were randomly sampled from the MacKenzie phrase set [32]. We ensured all users would type the same phrases in 5 days, the phrases within each keyboard are not repeated, and the phrases between two keyboards are the same (MacKenzie phrase set only contains 500 phrases, but we need 700 phrases for each user including warm-up blocks), but the phrase order for each user and keyboard was randomly shuffled. The keyboard order was also counterbalanced to avoid the learning effect problem. The software design and user operations in this study are described in Section 3. The detailed procedures for each day include:

(1) Typing on physical keyboard: 1 block. Participants used their own laptop keyboards to transcribe 10 phrases using the default notepad on Windows or macOS.
(2) Typing on tablet keyboard: 1 block (counterbalanced with physical keyboards). Participants transcribed 10 phrases using the "Notes" app on an 11-inch iPad Pro 2019.
(3) Typing on ResType: 5 blocks. Users first took a warm-up block and then transcribed 5 blocks of phrases using ResType. They were forced to take a one-minute break between two blocks to relieve fatigue.
(4) Typing on ResTypeStatic: 5 blocks (counterbalanced with ResType). The procedures are exactly the same with ResType except that users typed on ResTypeStatic.

Table 1: Users' typing experience.

Experience	M	SD	MIN	MAX
Tablet Keyboard (year)	6.1	3.5	1	12
Physical Keyboard (year)	12.2	3.8	3	18
Self-rated touch typing level (1-7)	5.5	1.3	2	7

(5) Subjective feedback: on day 1 and day 5, users rated on ResTypeStatic and ResType respectively, in terms of typing speed, accuracy, fatigue, cognitive burden, and general experience.

Each user spent an hour and was paid $15 each day in the study.

7.2 Result

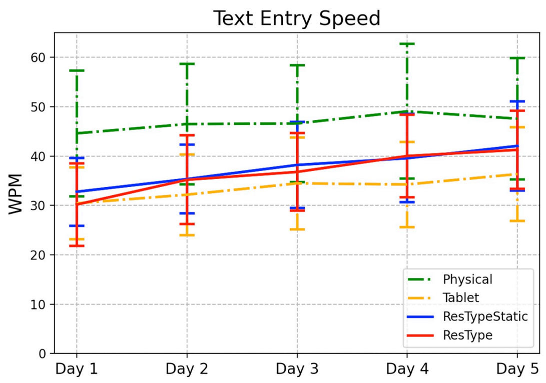


Figure 8: The text entry speed (WPM) over 5 days on ResType and ResTypeStatic. Error bars indicate the standard deviation.

7.2.1 Typing Speed. Text entry speed is measured in Words per Minute (WPM) by [5]:

$$WPM = \frac{|L| - 1}{T} \times 60 \times \frac{1}{5} \tag{6}$$

where $|L|$ is the total length of the transcribed phrases, and T is the time cost in seconds. We conducted a two-way RM ANOVA test on text entry speed with keyboard and day as two within factors. If any of the variables had a significant effect ($p < 0.05$), we adopted a multiple pairwise t-test with Bonferroni correction for comparison.

Figure 8 shows the text entry speed on ResTypeStatic and ResType. ResTypeStatic starts with 32.78 WPM (SD = 6.87) and ends with 42.07 WPM (SD = 9.07, increased 28.3%), while ResType starts with 30.18 WPM (SD = 8.37) and ends with 41.26 WPM (SD = 7.91, increased 36.7%). Keyboard has no significant effect on text entry speed ($F_{1,13} = 0.69, p = 0.42$), while day has significant effect on it ($F_{4,52} = 45.07, p < 0.001$). The interaction effect between day and keyboard is not significant ($F_{4,52} = 1.93, p = 0.12$). For the day factor, post hoc test shows significant effect between all day pairs ($p < 0.05$) except day2-day3 ($p = 0.08$) and day4-day5 ($p = 0.24$). The result shows comparable input speed and learning burden between ResType and the state-of-the-art tablet keyboard.

Furthermore, tablet keyboards start with 30.44 WPM (SD = 7.26) and end with 36.36 WPM (SD = 9.47, increased 19.4%). Physical keyboards start with 44.60 WPM (SD = 12.73) and end with 47.58 WPM (SD = 12.27, increased 6.7%). Their average input speeds are 33.54 and 46.87 WPM respectively. On day 5, text entry speed on ResType outperformed tablet keyboards by 13.5% (23.0% for the average speed) and reached 86.7% (88.0% for the average speed) of physical keyboards. The relative speed outperformed the prior work [42] (66.7% of physical keyboards).

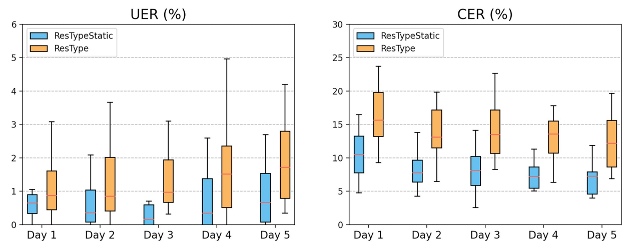


Figure 9: Uncorrected and corrected error rates (%) on ResTypeStatic and ResType over 5 days.

7.2.2 Error Rate. We measured users' word-level uncorrected error rate (UER) and corrected error rate (CER) on ResTypeStatic and ResType over 5 days (Figure 9). UER measures the input errors that remained in the transcribed phrases, which is calculated by the number of uncorrected words divided by the total word number ($|L| - 1/5$, the same as Equation 6). Two-way RM ANOVA shows that keyboard has significant effect on UER ($F_{1,13} = 12.6, p < 0.01$) but not day ($F_{4,52} = 0.97, p = 0.43$). The mean UER over 5 days is 0.84% (SD = 1.21%) on ResTypeStatic and 1.55% (SD = 1.36%) on ResType, which means users left 1 uncorrected error every 119.0 words on ResTypeStatic and every 64.5 words on ResType. CER measures the input errors corrected by users intentionally, which is calculated by the sum of errors corrected by deletion and selection divided by the total word number. Both keyboard ($F_{1,13} = 69.1, p < 0.001$) and day ($F_{4,52} = 7.0, p < 0.001$) have significant effect on CER. The mean CER over 5 days is 8.35% on ResTypeStatic and 14.09% on ResType. In all corrected errors, the proportion of errors corrected by selection is 5.70% on ResTypeStatic and 11.33% on ResType (selection and deletion sum to 100%).

The result shows that users made more mistakes on ResType than on ResTypeStatic, which is consistent with study one that it is more difficult for users to locate keys precisely on ResType due to the invisible keyboard layout. Generally, UER on both ResTypeStatic and ResType is low and acceptable.

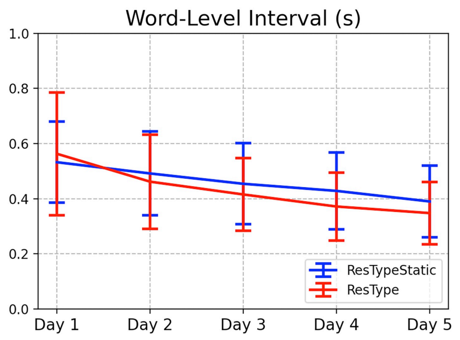


Figure 10: Word-level intervals (s) on two keyboards over 5 days. Error bars indicate standard deviation.

7.2.3 Visual Attention. On ResType, users do not need to switch visual attention between the input area and the keyboard. We first explored this effect on the time components during typing. We measured the Word-Level Interval (WLI) in each text entry block, which is defined as the average touch event intervals between every two neighboring words (e.g. the WLI of "human computer interaction" is the average time interval of "n⟶c" and "r⟶i"). WLI reflects the input time delay between words. Longer intervals could be caused by extra efforts for locating keys, switching visual attention, etc. Figure 10 shows the WLI on ResTypeStatic and ResType over 5 days. One-way RM ANOVA shows keyboard WLI has significant difference on day 4 ($F_{1,13} = 7.65, p < 0.05$) and marginal significant difference on day 5 ($F_{1,13} = 3.57, p = 0.08$). Day has significant effect on WLI for both ResType ($F_{4,52} = 26.53, p < 0.001$) and ResTypeStatic ($F_{4,52} = 19.27, p < 0.001$).

Inspired by the WLI difference, we further designed a follow-up evaluation study to collect users' gaze behavior during typing, which could reveal how users' visual attention switches when using the two keyboards. We recruited exactly the same 14 participants in the 5-day study and repeated exactly the same procedures in subsection 7.1 one more day, except that participants wore a gaze tracking device (Pupil Core [15]) to reveal the focus of their visual attention, and that the keyboard layout in the GUI was removed (discussed later in subsection 8.4). Pupil Core uses a world camera and a pupil camera to show the gaze position after the device calibration. We labeled each pair of timestamps when the users' gaze position left the input area to see the touchpad and returned back to the input area (a gaze switch). Table 2 shows the input speed, the times of gaze switches per block, the average time cost of each gaze switch, and the time proportion of gaze switches in the total input time.

This follow-up study was conducted 3 months later than the 5-day evaluation. We recruited the same participants for the reason that they were already familiar with ResTypeStatic and ResType, so the gaze switching behavior would be more reliable. Users achieved 40.00 WPM on ResTypeStatic (between day 4 and 5, 95.1% of day 5, see subsection 7.2.1) and 38.87 WPM on ResType (between day 3 and 4, 94.2% of day 5), which shows that they basically maintained the typing ability learned from the 5-day evaluation. RM ANOVA shows both the times and proportion of gaze switching on ResType are significantly lower than on ResTypeStatic (the average time cost per switch is not comparable because only 5 users had at least one gaze switch on ResType). In other words, 34.70% of users' visual attention was not in the input area when using ResTypeStatic. Leveraging the adaptive keyboard, ResType reduces this proportion to 0.07%, which could be beneficial in scenarios when the input area needs dense visual attention (e.g. shorthands).

7.2.4 Calibration Behavior. Users rested their fingers on the home row to calibrate the keyboard position. In this study, we did not force users to calibrate on ResType. Instead, they were instructed to calibrate only when they needed (e.g. after moving hands significantly). We measured the left-hand and right-hand keyboard offsets to the standard position after calibration to explore the hand drift effect. We also measured the calibration times, time cost, and time proportion to explore users' willingness to use this feature. Table 3 lists the result in each block over 5 days.

One-way ANOVA shows day has no significant effect on all data types. In general, the left-hand offset was 1.34 cm (SD = 0.64) and the right-hand offset was 1.76 cm (SD = 0.68). Their standard deviations correspond to 1.25 cm and 1.33 cm in 95% confidence intervals, which means users' hands would drift 0.66 to 0.70 of the key size (1.9 cm) from the key centers while touch typing. This offset is big enough (> 0.5 key size) to trigger unwanted touches without the adaptive keyboard. Furthermore, users calibrated 6.07 times per block (10 phrases), which means one calibration per 1.65 phrases. The average time cost per calibration is 407 ms, and calibration only takes up 2.32% of the typing time, which indicates calibration is easily adopted by users and takes little extra effort. In the user interview, participants also reported that the calibration behavior on ResType was very natural to them, as they all have the habit to align their fingers on physical keyboards using the landmarks.

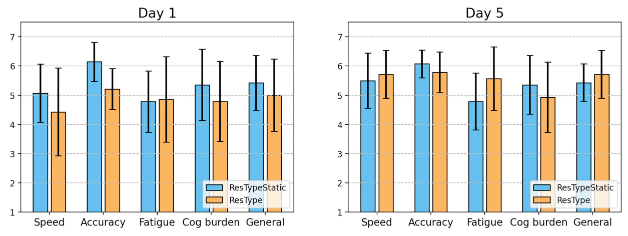


Figure 11: Subjective ratings on ResTypeStatic and ResType (integers from 1 to 7, higher is better). Error bars indicate the standard deviation.

7.2.5 Subjective Feedback. On day 1 and day 5, we collected users' subjective ratings on the following aspects:

(1) Speed: the keyboard input speed.
(2) Accuracy: the keyboard typing (locating desired keys) and decoding (predicting input words) accuracy.
(3) Fatigue: the physical demand (e.g. switching visual attention). Higher scores represent lower fatigue.
(4) Cognitive burden: the mental demand (e.g. locating keys). Higher scores represent a lower burden.
(5) General: the overall keyboard using experience.

Table 2: Users' input speed and gaze switching behavior on ResTypeStatic and ResType in the follow-up study. Parentheses show standard deviations.

Keyboard	Speed (WPM)	Times per block	Average time (ms)	Proportion (%)
ResTypeStatic	40.00 (8.90)	37.81 (17.95)	878 (168)	34.70 (14.48)
ResType	38.87 (9.32)	0.09 (0.13)	907 (178, 5 users)	0.07 (0.09)
Significance	$F_{1,13} = 0.57, p = 0.46$	$F_{1,13} = 62.36, p < 0.001$	-	$F_{1,13} = 80.57, p < 0.001$

Table 3: Users' calibration behavior on ResType over 5 days. Parentheses show standard deviations.

Day	F offset (cm)	J offset (cm)	Times per block	Average time (ms)	Time proportion (%)
Day 1	1.39 (0.72)	1.92 (0.67)	6.27 (4.61)	565 (395)	2.52 (1.85)
Day 2	1.58 (0.79)	2.01 (0.82)	5.63 (3.82)	412 (201)	2.25 (1.94)
Day 3	1.37 (0.54)	1.78 (0.75)	5.83 (3.26)	388 (203)	2.23 (1.72)
Day 4	1.23 (0.54)	1.54 (0.54)	5.84 (3.33)	338 (138)	2.15 (1.47)
Day 5	1.15 (0.59)	1.56 (0.56)	6.77 (4.55)	327 (144)	2.42 (1.79)
Overall	1.34 (0.64)	1.76 (0.68)	6.07 (3.86)	407 (245)	2.32 (1.72)

We explained the meaning of each aspect to participants in detail. Figure 11 shows the result.

A Wilcoxon signed-rank test was performed for each aspect on keyboard and day respectively. Ratings of ResType significantly improved over days on speed ($Z = -2.62, p < 0.01$), accuracy ($Z = -2.27, p < 0.05$), general ($Z = -2.00, p < 0.05$) and marginal significantly improved on fatigue ($Z = -1.78, p = 0.07$) but not on the cognitive burden ($Z = -0.51, p = 0.61$). Ratings of ResTypeStatic had no significant improvement over days on all aspects. On day 1, keyboard has a significant effect only on accuracy ($Z = -2.81, p < 0.01$), while on day 5, keyboard has a significant effect only on fatigue ($Z = -2.37, p < 0.05$). The result shows that the general ratings of ResType were lower on day 1 but improved significantly when users learned how to use it. Also, ResType can reduce physical fatigue as expected. The accuracy of ResType is lower than ResTypeStatic on day 1, but can be improved as the user learns to use it (not significant on day 5). The average general rating of ResType is higher than ResTypeStatic on day 5, which also shows the high usability of ResType.

8 DISCUSSION

8.1 The Performance and Learning Effect

On day 5 of study three, participants achieved 41.26 WPM on ResType, which is not very high for ten-finger typing. However, their inherent typing speed is 47.58 WPM on physical keyboards, which is far lower than prior work (e.g. 67 WPM in TOAST [42]). Therefore, we can conclude the low text entry speed results from participants' low typing expertise. Relatively, participants achieved 13.5% faster WPM on ResType than on tablet keyboards, and reached 86.7% of physical keyboards, which is a huge increase compared with prior work (e.g. 66.7% of physical keyboards in TOAST [42]). ResType also achieved comparable text entry speed as state-of-the-art methods on tablets, though without a visible layout. As a result, we showed the feasibility of ResType in terms of relative performance.

In our within-subject study, participants learned to type on ResTypeStatic and ResType alternately in 5 days. We adopted this design to ensure the same typing ability of ResTypeStatic users and ResType users. The typing time on these two keyboards was around 200 minutes in total. The learning curve of ResType did not converge on the last day, so we do not claim 41.26 WPM as the ceiling text entry speed. Higher speed could be achieved with more practice.

Aside from the input speed, we showed that ResType and ResTypeStatic are both efficient input methods but also have strengths and weaknesses. On ResTypeStatic, users can see the keyboard layout, thus locating keys more precisely and erring less. On ResType, users do not need to switch visual attention to locate keys, thus achieving lower WLI and being more focused on the input area. Though the two differences result in similar input speeds in our evaluation, we believe situations that require more attention but less accuracy are more suitable for ResType, like shorthands.

8.2 The Evaluation Study Setting

We compared the performance of ResType with ResTypeStatic in study three for the following considerations. First, ResType contains three main features: unintentional touch prevention, statistical decoding, and the adaptive keyboard. To evaluate the influence of the adaptive keyboard, we implemented ResTypeStatic with only the first two features to control the variables. Second, should ResTypeStatic have a visible keyboard layout? We tested the "invisible ResTypeStatic" (layout not drawn on the touchpad) in a pilot study. It turned out users could only locate keys by trial and error and can hardly finish the text entry tasks in a reasonable duration, so we did not adopt this setting at last. Actually, ResTypeStatic could be considered as a visual representation of ResType. It is also closer to the current methods on tablets. Users locate keys by vision on ResTypeStatic and by the adaptive keyboard on ResType. The "invisible ResTypeStatic" does not support either, so it is unfair for comparison.

8.3 Keyboard Fitting

The problem of keyboard fitting involves three key factors: translation, rotation, and scaling [42]. In the final ResType, we only considered keyboard translation, so why not rotation and scaling? First, the rotation of users' fingers barely changed during typing, which is restricted by the touchpad size. Second, we found the scale of users' resting fingers had no correlation with the actual scale of the touchpoint clouds. Since users cannot feel the keys on ResType, they rest their fingers on ResType just in a comfortable way so that the touchpoint patterns are distorted. Instead, since users have already formed muscle memory of physical keyboards, the scale of the touchpoint clouds matches the standard keyboard layout very well. As a result, fitting rotation and scaling by calibration did not improve the decoding accuracy in the simulation.

8.4 The Occlusion Problem

One motivation of ResType is to hide the keyboard layout to save screen space. To help users who are not familiar with touch typing, we displayed the standard QWERTY layout for all users in the GUI. After user study three, we interviewed the 14 participants whether the layout was helpful for touch typing, and only 3 users reported so. We further informally tested its influence in the follow-up evaluation (subsection 7.2.3). Without any hints on the screen, users still achieved comparable input speed on ResType compared to ResTypeStatic. For real applications, if we need to implement ResType on touchscreens, this keyboard layout could be shrunk and moved to a proper position (e.g. a small icon beside the input area), or be completely hidden for skilled touch typists, such that it would not occupy any screen space.

Aside from the GUI, users' hands would partly block the screen as well (Figure 1 (d)) in tablet interactions. This issue is beyond the scope of keyboard design, as long as users still need to perform ten-finger touch typing on tablets. However, if the information on the screen is sparse enough for users to see through the gaps between fingers, like chat history, images or graphs, users can still access these information more easily compared to traditional soft keyboards, which is also demonstrated in Sun's work [45]. In conclusion, though we have only removed the occlusion from soft keyboards, we believe the invisible keyboard design in ResType is beneficial for tablet text entry.

8.5 Applications of ResType

ResType envisions a design that the virtual keyboard on tablets adapts to users' hands, enabling efficient touch typing anywhere on the surface. It does not need a visible layout and reduces visual attention switches. We propose the following applications of ResType leveraging these benefits:

(1) Tablet computer: it can use ResType as an invisible keyboard to save screen space and reduce the cost of switching visual attention.
(2) Dual-screen laptop [1]: a laptop containing no physical keyboard but two touchscreens inside. This design faces the problem of unintentional touches and low input efficiency, which could be solved by ResType effectively.
(3) Smart surface: as ubiquitous computing develops, we are confident that much more sensors will be deployed in smart AIoT devices in the future. ResType only needs haptic signals to identify unintentional touches and adapt the keyboard to users' hands, which is inexpensive and easy to deploy in a large scale. Haptic sensors deployed on smart surfaces (e.g. table tops) could support multiuser touch typing in discussions, meetings or games, etc.
(4) AR/VR: AR/VR applications commonly leverage computer vision to support in-air typing or render virtual keyboards on table surfaces. However, these CV-based methods need to track fingers, detect touch events and map them to the keyboard layout, which is indirect and inaccurate [34, 38]. We could deploy ResType on table surfaces and use the adaptive keyboard to improve the experience of AR/VR typing.

9 LIMITATIONS AND FUTURE WORK

Though we achieved satisfying performance on ResType, our work has several limitations, which provide motivations for future work:

(1) Input limitations: we only implemented 26 English letters and "space" to input. Punctuations and other symbols could be input by gestures or separate input modes. Besides, the decoding algorithm only predicts words with the same length as the input sequence and does not support OOV (Out of Vocabulary) words.
(2) Keyboard fitting: ResType does not support rotation as discussed in subsection 8.3. Future work on large sensing surfaces or multiuser scenarios should consider keyboard rotation to achieve a more flexible input experience.
(3) Personalization: we pooled all users' typing data and implemented the general touch model in study two. For long-term use or for users with special input habits, personalization should be considered.
(4) Hardware: ResType requires recognizing resting touchpoints, which is not supported by most commercial tablets currently. Furthermore, we implemented and tested ResType only on the Sensel Morph touchpad to identify unintentional touches accurately. Since the touchpad was separated from the computer screen, it may lead to a slightly different user behavior (e.g. whether users' hands could block the screen or not). If possible, future work should test it on larger surfaces or touchscreens to explore ResType on a wider range of devices.

10 CONCLUSION

We present ResType, an invisible and adaptive keyboard that supports touch typing on three-state touch surfaces (e.g. tablets with unintentional touch prevention). Users rest their fingers on the home row of an imaginary keyboard, and the virtual keyboard will adapt to their fingers automatically. We followed TypeBoard [20] to prevent unintentional touches. We explored user behavior on ResType, including calibration and touchpoint distribution. We then designed keyboard fitting and input decoding algorithms to support adaptive keyboard and touch typing. The decoding algorithm achieved 96.3% top-1 and 99.0% top-3 accuracies in user studies. After a 5-day user study, users achieved 41.26 WPM on ResType, outperforming tablet keyboards by 13.5% and reaching

86.7% of physical keyboards. It also achieved comparable performance with state-of-the-art methods on tablets and does not need a visible layout or display devices. ResType supports efficient touch typing, which could be used in scenarios that require less or no visible keyboard layout, like tablet touchscreens, smart surfaces, and AR/VR.

ACKNOWLEDGMENTS

This work is supported by the Natural Science Foundation of China under Grant No. 62132010, and by Beijing Key Lab of Networked Multimedia, the Institute for Guo Qiang, Tsinghua University, Institute for Artificial Intelligence, Tsinghua University (THUAI), and by 2025 Key Technological Innovation Program of Ningbo City under Grant No. 2022Z080.

REFERENCES

[1] 2018. The Project Precog. https://www.asus.com/Content/Project-Precog/

[2] 2022. The Sensel Morph. https://morph.sensel.com

[3] 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.

[4] Ahmed Sabbir Arif, Benedikt Iltisberger, and Wolfgang Stuerzlinger. 2011. Extending Mobile User Ambient Awareness for Nomadic Text Entry. In *Proceedings of the 23rd Australian Computer-Human Interaction Conference* (Canberra, Australia) *(OzCHI '11)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/2071536.2071539

[5] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. 2009. Analysis of text entry performance metrics. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*. 100–105. https://doi.org/10.1109/TIC-STH.2009.5444533

[6] 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* (San Francisco, California, USA) *(MobileHCI '12)*. Association for Computing Machinery, New York, NY, USA, 251–260. https://doi.org/10.1145/2371574.2371612

[7] 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* (Lisbon, Portugal) *(IUI '12)*. Association for Computing Machinery, New York, NY, USA, 11–20. https://doi.org/10.1145/2166966.2166969

[8] Daniel Buschek, Oliver Schoenleben, and Antti Oulasvirta. 2014. Improving Accuracy in Back-of-Device Multitouch Typing: A Clustering-Based Approach to Keyboard Updating. In *Proceedings of the 19th International Conference on Intelligent User Interfaces* (Haifa, Israel) *(IUI '14)*. Association for Computing Machinery, New York, NY, USA, 57–66. https://doi.org/10.1145/2557500.2557501

[9] Lung-Pan Cheng, Hsiang-Sheng Liang, Che-Yang Wu, and Mike Y. Chen. 2013. IGrasp: Grasp-Based Adaptive Keyboard for Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3037–3046. https://doi.org/10.1145/2470654.2481422

[10] Elaine Lawrence Christian Sax, Hannes Lau. 2011. Liquid Keyboard : An Ergonomic, Adaptive QWERTY Keyboard for Touchscreens and Surfaces. http://hdl.handle.net/10453/16246

[11] Guofeng Feng, Xiao Guo, Minsi Ren, Shuoming Zhang, and Jie Liu. 2020. NaviKey: An Invisible Keyboard with Navigation Keys. In *The Eighth International Workshop of Chinese CHI* (Honolulu, HI, USA) *(Chinese CHI 2020)*. Association for Computing Machinery, New York, NY, USA, 61–64. https://doi.org/10.1145/3403676.3403684

[12] Leah Findlater and Jacob Wobbrock. 2012. Personalized Input: Improving Ten-Finger Touchscreen Typing through Automatic Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 815–824. https://doi.org/10.1145/2207676.2208520

[13] 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* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2453–2462. https://doi.org/10.1145/1978942.1979301

[14] Apostolos Gkoumas, Andreas Komninos, and John Garofalakis. 2016. Usability of Visibly Adaptive Smartphone Keyboard Layouts. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics* (Patras, Greece) *(PCI '16)*. Association for Computing Machinery, New York, NY, USA, Article 40, 6 pages. https://doi.org/10.1145/3003733.3003743

[15] Pupil Labs GmbH. 2022. The Pupil Core gaze tracking device. https://pupil-labs.com/products/core/

[16] 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* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2795–2798. https://doi.org/10.1145/2470654.2481386

[17] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. GripSense: Using Built-in Sensors to Detect Hand Posture and Pressure on Commodity Mobile Phones. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 545–554. https://doi.org/10.1145/2380116.2380184

[18] 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* (San Francisco, California, USA) *(IUI '02)*. Association for Computing Machinery, New York, NY, USA, 194–195. https://doi.org/10.1145/502716.502753

[19] Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. 2004. A Reduced QWERTY Keyboard for Mobile Text Entry. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria) *(CHI EA '04)*. Association for Computing Machinery, New York, NY, USA, 1429–1432. https://doi.org/10.1145/985921.986082

[20] Yizheng Gu, Chun Yu, Xuanzhong Chen, ZHUOJUN LI, and Yuanchun Shi. 2021. TypeBoard: Identifying Unintentional Touch on Pressure-Sensitive Touchscreen Keyboards. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 568–581. https://doi.org/10.1145/3472749.3474770

[21] Yizheng Gu, Chun Yu, Zhipeng Li, Zhaoheng Li, Xiaoying Wei, and Yuanchun Shi. 2020. QwertyRing: Text Entry on Physical Surfaces Using a Ring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4, Article 128 (dec 2020), 29 pages. https://doi.org/10.1145/3432204

[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* (Hong Kong, China) *(IUI '10)*. Association for Computing Machinery, New York, NY, USA, 111–118. https://doi.org/10.1145/1719970.1719986

[23] Johan Himberg, Jonna Häkkilä, Petri Kangas, and Jani Mäntyjärvi. 2003. On-Line Personalization of a Touch Screen Based Keyboard. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (Miami, Florida, USA) *(IUI '03)*. Association for Computing Machinery, New York, NY, USA, 77–84. https://doi.org/10.1145/604045.604061

[24] Johannes Hirche, Peter Bomark, Mikael Bauer, and Pawel Solyga. 2008. Adaptive interface for text input on large-scale interactive surfaces. In *2008 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*. 153–156. https://doi.org/10.1109/TABLETOP.2008.4660198

[25] Nancy Ide. 2012. The American National Corpus: Then, Now, and Tomorrow. (03 2012).

[26] Hwan Kim, Yea-kyung Row, and Geehyuk Lee. 2012. Back Keyboard: A Physical Keyboard on Backside of Mobile Phone Using Qwerty. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI EA '12)*. Association for Computing Machinery, New York, NY, USA, 1583–1588. https://doi.org/10.1145/2212776.2223676

[27] Sunjun Kim, Jeongmin Son, Geehyuk Lee, Hwan Kim, and Woohun Lee. 2013. TapBoard: Making a Touch Screen Keyboard More Touchable. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 553–562. https://doi.org/10.1145/2470654.2470733

[28] Ue-Hwan Kim, Sahng-Min Yoo, and Jong-Hwan Kim. 2021. I-Keyboard: Fully Imaginary Keyboard on Touch Devices Empowered by Deep Neural Decoder. *IEEE Transactions on Cybernetics* 51, 9 (2021), 4528–4539. https://doi.org/10.1109/TCYB.2019.2952391

[29] Jianwei Lai, Dongsong Zhang, Sen Wang, Isil Doga Yakut Kilic, and Lina Zhou. 2019. ThumbStroke: A Virtual Keyboard in Support of Sight-Free and One-Handed Text Entry on Touchscreen Mobile Devices. *ACM Trans. Manage. Inf. Syst.* 10, 3, Article 11 (sep 2019), 19 pages. https://doi.org/10.1145/3343858

[30] Frank Chun Yat Li, Richard T. Guy, Koji Yatani, and Khai N. Truong. 2011. The 1line Keyboard: A QWERTY Layout in a Single Line. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 461–470. https://doi.org/10.1145/2047196.2047257

[31] 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

(jun 2017), 24 pages. https://doi.org/10.1145/3090083

[32] 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* (Ft. Lauderdale, Florida, USA) *(CHI EA '03)*. Association for Computing Machinery, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[33] I. Scott MacKenzie, R. William Soukoreff, and Joanna Helga. 2011. 1 Thumb, 4 Buttons, 20 Words per Minute: Design and Evaluation of H4-Writer. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 471–480. https://doi.org/10.1145/2047196.2047258

[34] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: A Mid-Air Word-Gesture Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1073–1082. https://doi.org/10.1145/2556288.2556964

[35] Kseniia Palin, Anna Maria Feit, Sunjun Kim, Per Ola Kristensson, and Antti Oulasvirta. 2019. How Do People Type on Mobile Devices? Observations from a Study with 37,000 Volunteers. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services* (Taipei, Taiwan) *(MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, Article 9, 12 pages. https://doi.org/10.1145/3338286.3340120

[36] Gulnar Rakhmetulla and Ahmed Sabbir Arif. 2020. Senorita: A Chorded Keyboard for Sighted, Low Vision, and Blind Mobile Users. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376576

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

[38] Mark Richardson, Matt Durasoff, and Robert Wang. 2020. *Decoding Surface Touch Typing from Hand-Tracking*. Association for Computing Machinery, New York, NY, USA, 686–696. https://doi.org/10.1145/3379337.3415816

[39] Mario Romero, Brian Frey, Caleb Southern, and Gregory D. Abowd. 2011. BrailleTouch: Designing a Mobile Eyes-Free Soft Keyboard. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (Stockholm, Sweden) *(MobileHCI '11)*. Association for Computing Machinery, New York, NY, USA, 707–709. https://doi.org/10.1145/2037373.2037491

[40] 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* (Munich, Germany) *(MobileHCI '13)*. Association for Computing Machinery, New York, NY, USA, 175–178. https://doi.org/10.1145/2493190.2493233

[41] James Scott, Shahram Izadi, Leila Sadat Rezai, Dominika Ruszkowski, Xiaojun Bi, and Ravin Balakrishnan. 2010. RearType: Text Entry Using Keys on the Back of a Device. In *Proceedings of the 12th International Conference on Human Computer Interaction with Mobile Devices and Services* (Lisbon, Portugal) *(MobileHCI '10)*. Association for Computing Machinery, New York, NY, USA, 171–180. https://doi.org/10.1145/1851600.1851630

[42] 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 (mar 2018), 23 pages. https://doi.org/10.1145/3191765

[43] Touchscape Software. 2020. KlearKeys: Transparent Keyboard Phone. https://play.google.com/store/apps/details?id=com.touchscapesoftware.klearkeys.phone

[44] 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* (San Francisco, California, USA) *(MobileHCI '12)*. Association for Computing Machinery, New York, NY, USA, 317–326. https://doi.org/10.1145/2371574.2371623

[45] Ke Sun, Chun Yu, and Yuanchun Shi. 2019. Exploring Low-Occlusion Qwerty Soft Keyboard Using Spatial Landmarks. *ACM Trans. Comput.-Hum. Interact.* 26, 4, Article 20 (jun 2019), 33 pages. https://doi.org/10.1145/3318141

[46] 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* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2307–2316. https://doi.org/10.1145/2556288.2557412

[47] Wikipedia. 2022. Touch typing. https://en.wikipedia.org/wiki/Touch_typing

[48] Ying Yin, Tom Yu Ouyang, Kurt Partridge, and Shumin Zhai. 2013. Making Touchscreen Keyboards Adaptive to Keys, Hand Postures, and Individuals: A Hierarchical Spatial Backoff Model Approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2775–2784. https://doi.org/10.1145/2470654.2481384

[49] Sahng-Min Yoo, Ue-Hwan Kim, Yewon Hwang, and Jong-Hwan Kim. 2021. Type Anywhere You Want: An Introduction to Invisible Mobile Keyboard. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1757–1764. https://doi.org/10.24963/ijcai.2021/242 Main Track.

[50] 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* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174013