Published in final edited form as:

Proc ACM Symp User Interface Softw Tech. 2024 ; 2024: . doi:10.1145/3654777.3676447.

Accessible Gesture Typing on Smartphones for People with Low Vision

Dan Zhang,
Department of Computer Science, Stony Brook University New York, USA

William H Seiple,
Lighthouse Guild New York, USA

Zhi Li,
Department of Computer Science, Stony Brook University New York, USA

IV Ramakrishnan,
Department of Computer Science, Stony Brook University New York, USA

Vikas Ashok,
Department of Computer Science, Old Dominion University Virginia, USA

Xiaojun Bi
Department of Computer Science, Stony Brook University New York, USA

Abstract

While gesture typing is widely adopted on touchscreen keyboards, its support for low vision users is limited. We have designed and implemented two keyboard prototypes, layout-magnified and key-magnified keyboards, to enable gesture typing for people with low vision. Both keyboards facilitate uninterrupted access to all keys while the screen magnifier is active, allowing people with low vision to input text with one continuous stroke. Furthermore, we have created a kinematics-based decoding algorithm to accommodate the typing behavior of people with low vision. This algorithm can decode the gesture input even if the gesture trace deviates from a pre-defined word template, and the starting position of the gesture is far from the starting letter of the target word. Our user study showed that the key-magnified keyboard achieved 5.28 words per minute, 27.5% faster than a conventional gesture typing keyboard with voice feedback.

Keywords

low vision; accessibility; smartphone keyboard; text input; gesture input; word gesture

zhang64@cs.stonybrook.edu .

1 INTRODUCTION

The text entry technology in smartphones has advanced rapidly. Gesture typing [19, 46–48] allows users to enter text by gliding a finger over the letters of a word. This method suits touch-based interaction and supports word-level input. It is available on all major touchscreen keyboards, including Google's Gboard, Microsoft's SwiftKey, and iOS's built-in keyboard. Despite its popularity, many who could benefit from gesture typing, notably people with low vision, are excluded. Low vision is vision loss uncorrectable by refractive optics or surgery [35], encompassing conditions like limited vision field, light sensitivity, blurry vision, or contrast issues. This paper focuses on low vision individuals who can see keys after magnifying the keyboard. They often use screen magnifiers to interact with smartphones. To type a word, they magnify the keyboard, pan to find each letter, and tap to select it, repeating this for every letter. This process is slow and laborious. Gesture typing could help by eliminating repetitive actions for each letter.

Existing gesture typing keyboards poorly support low vision users. Two challenges stand out: (1) With the screen magnifier on, some keys on the soft keyboard become invisible if they are outside the viewport. Consequently, words containing these letters cannot be entered in one stroke. (2) Low vision users exhibit distinct input behaviors. The landing position of the input finger may be far from the starting letter, and users often wiggle the input finger due to delayed audio feedback and limited visual feedback. Current gesture keyboards, like the SHARK^2 decoding algorithm [19], match the input gesture's shape with pre-defined templates, assuming it will approximate the intended word's shape—this does not account for low vision users' behaviors. Our user study I with 10 low vision users showed that half could not use an existing gesture keyboard.

In this paper, we designed and implemented the prototypes of two gesture typing keyboards for low vision users: layout-magnified and key-magnified keyboards (Figure 1). We have made innovations in both the interface and decoding algorithm design. On the interface front, the layout-magnified keyboard magnifies the entire keyboard layout and supports a user to enter any letters in one continuous stroke; the key-magnified keyboard magnifies the key underneath the finger only thus all the keys are visible and accessible during the gesture input. On the algorithm front, we designed and implemented a kinematics-based decoding algorithm which decodes a gesture input based on its kinematic features such as the distance between the turning points and key centers, and the speed of the gesture as it crosses a letter. This algorithm accommodates the unique input behavior of low vision people whose gesture often deviates a great deal from the ideal template of entering a word (i.e., the polyline prototype connecting the centers of keys).

We conducted a user study II to investigate the effectiveness of the two novel gesture typing keyboards designed for users with low vision: layout-magnified and key-magnified keyboards. The results showed the key-magnified keyboard improved typing speed for low vision users compared to a conventional gesture typing keyboard with integrated voice feedback. Participants achieved typing speeds of 5.28 words per minute (WPM) with key-magnified keyboard, representing increases of 27.5%. Furthermore, these keyboards outperformed a typical commercial implementation of tap typing keyboard with a TalkBack

default method. Typing speeds increased by 54.2% and 77.6% for layout-magnified and key-magnified keyboards, respectively, with error rate reductions of 70.3% and 78.6%. These findings demonstrate that the key-magnified keyboard has the potential to expand the accessibility of gesture typing technology for users with low vision.

2 RELATED WORK

2.1 Text Entry Accessibility Tools on Smartphones

People with low vision rely on accessibility tools to interact with smartphones. The commonly used accessibility features on a smartphone are magnification, text size, audio feedback, contrast enhancement, black/white reverse [13, 49]. Among these, magnification and audio feedback are the most widely adopted methods. Magnification allows people to temporarily zoom into the screen and adjust the zoom level freely according to their preference. Numerous studies have leveraged the magnifier effect to enhance text entry such as ZoomBoard [29] and Swipeboard [8]. Audio feedback is an eyes-free technique that guides users through on-screen content using simple gestures by audibly describing interface elements, such as VoiceOver [39] on iPhone and TalkBack [36] on Android. Additionally, color inversion and text size settings offer customization options to improve visual clarity and reduce glare for users with low vision. Integrating these accessibility features into smartphones is crucial to empower people with low vision to fully engage in mobile technology.

Voice input is a popular method to input text on the phone. Early methods such as Dragon Naturally Speaking [26] were for desktops. VoiceTyping [20] supports dictation using speech recognition and works on mobile devices. Despite its popularity, researchers found that incorrectly recognized voice input can be a major reason to impede the speech input speed [1, 17].

2.2 Gesture Typing

Gesture typing is a word-level text entry method where users swipe on a keyboard to connect the letters of a word, which is first proposed by Zhai and Kristensson [19, 46–48] and has been adopted in various commercial keyboards. Compared to traditional tap typing, gesture typing requires less precise motor control and allows for continuous text input, potentially offering enhanced accessibility for low vision people.

Researchers have investigated the application of gesture typing across a spectrum of input modalities and diverse usage scenarios. Bi et.al [3] broadened the gesture keyboard from single-finger to multiple-finger input. Zhu et al. [51] proposed an eyes-free gesture typing method using a touch-enabled remote control. Yeo et. al [42] implemented a tilt-based gesture keyboard for single-handed text entry. Vulture [24] proposed mid-air gesture typing by projecting hand movement onto a display. Yu et.al [45] explored head-based gesture typing with head-mounted displays. Lin et al. showed that older adults typed 15% faster and with 27% fewer errors using gesture typing on QWERTY keyboards on the smartphone [21]. While gesture typing is becoming more and more popular [9–11, 21, 46–48], low vision people still face considerable barriers in using gesture typing on smartphones.

2.3 Text Entry Techniques for Low Vision People

Existing text entry techniques for users with low vision primarily focus on letter-level input. Rakhmetulla et al. designed Senorita [30], a two-thumb virtual chorded keyboard prioritizing comfort over speed for mobile devices. Lu et al. proposed a split keyboard where users focus on the output text displayed in their remaining vision while keeping the keyboard in their peripheral view [22]. VIP-Board [33] supports letter-level auto-correction by improving the keyboard decoder. Zhu et. al. [50] explored the potential of invisible keyboards on smartphones. Samanta et al. [31] proposed a text entry mechanism using directional movement gestures. Various eyes-free techniques have been developed to support text entry on a touch screen for low vision and blind people [6, 14, 25, 27, 28, 32, 37, 43]. These techniques include No-Look Notes by Bonner et al., allowing users to select letters through two virtual keys [6], NavTouch by Oliveira et al., utilizing gestures for letter navigation before selection [27], and Sánchez et al.'s method based on virtual key selection [32]. Guerreiro et al. explored bimanual interaction and stereo speech to enhance QWERTY keyboard speed, but this approach yielded limited benefits [16]. In addition, several recent studies [15, 18, 34] identified promising avenues for future research in text entry design for people with low vision.

AGTex [5] is the only well-established work that explored word-level text entry for people with visual impairments. It utilizes audio feedback to compensate for the absence of visual feedback. However, it does not fully utilize the user's potential and accessibility features such as the screen magnifier. To address this gap, this paper proposes a novel gesture input method that uses both auditory feedback and the magnifier functionality. Additionally, we introduce a customized decoder specifically designed to account for the unique input behaviors of low vision users.

3 USER STUDY I: UNDERSTANDING GESTURE TYPING FOR LOW VISION PEOPLE

Unlike previous researcher-centric approaches [15, 18], this paper adopts a user-informed design methodology. We first conducted a formative user study to evaluate how low vision participants use existing gesture typing keyboards. The insights from this user study guided the development of gesture typing keyboards aimed at improving accessibility.

3.1 Participants and Apparatus

We recruited 10 low vision participants (6 females, 4 males) aged 36 to 75 years (average 57.9 ± 15.2). Participants were from a non-profit vision and healthcare organization. The study was approved by the Institutional Review Board (IRB) and all participants provided their informed consent. Table 1 presents participant demographics. All participants used the index finger for text input. On a five-level Likert scale (1:never heard of - 5:expert), participants had a median familiarity of 1 with gesture typing and 3 with the QWERTY layout.

We used a Google Pixel 2 smartphone with a 5-inch 1920 × 1080 display running Android 11 for the study. We developed a web-based gesture typing keyboard resembling a modern

keyboard interface (see Figure 2) because commercial keyboards like Gboard or iPhone's built-in keyboard are incompatible with TalkBack and VoiceOver in gesture typing mode. Our keyboard supports accessibility features such as screen magnifier, color inversion, and voice feedback. It uses a SHARK^2 decoder [19, 47] and a bigram language model (10k unique words) to decode the user gesture.

3.2 Experiment Design

We adopted a text transcription task, where participants transcribe a set of phrases using our web-based keyboard. The task consists of a warm-up session with five phrases and a formal test session with 10 phrases. The warm-up phrases were randomly picked from a versatile phrase set for evaluating typing task [38]. The test phrases were randomly picked from the T-40 dataset [44]. All phrases were the same for each participant but presented in a random order. Figure 2 shows a screenshot of the transcription task. We recorded the following information for further analysis: timestamp, gesture trace, and current texts while typing.

3.3 Procedure

After greeting the participants, we introduced the purpose and obtained informed consent. We conducted a pre-study interview to collect demographic information, visual condition, typing posture, expertise, and QWERTY familiarity. Before the formal study, we configured the smartphone based on accessibility preferences (including the screen reader and color inversion), provided a 5-minute tutorial on gesture typing, and held a warm-up session with five phrases. During the formal study, participants were instructed to complete the transcription task as accurately and fast as possible in a comfortable posture. They may skip a word after five input attempts. They need to tap the submit button to complete a phrase. The formal study was limited to 40 minutes to account for typing variations, even if they unable to complete all phrases within this time. After the transcription task, participants were invited to participate in a post-study survey. This survey aimed to collect valuable feedback on potential areas for improvement in the gesture typing method. The user study lasted about an hour.

3.4 Results

3.4.1 Completion Rate.—

Table 2 presents the number of phrases completed by the 10 participants in the text transcription task. We can see that half of the participants were unable to complete the entire task. Among them, four participants encountered significant challenges during the warm-up session and were unable to transcribe any phrase. We identified several potential factors: (1) Screen size: The user study used a 5-inch phone, smaller than their usual phones (no less than 6 inches). This smaller screen size might have made accurate gesture typing difficult. (2) False spelling: Gesture typing involves swiping across multiple keys in sequence which requires users to coordinate finger movements. Learning, memorizing and consistently executing these gestures can be challenging. (3) Limited practice: Some participants (P4 and P7) mainly used voice input instead of on-screen typing. Learning a new typing method like gesture typing requires dedicated practice time.

3.4.2 Input Speed.—We measured the input speed of the participants based on their completed phrases by Words Per Minute (WPM) [41]. The mean speed (SD) across the 10 participants is 3.1 WPM (SD=2.22). We observed that the participants spent a large amount of time searching for letters.

3.4.3 Error Rate.—We looked into the Word Error Rate (WER) based on the completed phrases. The WER was calculated based on the Minimum Word Distance (MWD) [2]. The mean WER (SD) for all participants is 5.7% (SD = 11.16%).

3.4.4 Gesture Visualization.—To better understand how the typing error occurred, we visualized some representative gesture traces in Figure 3. In Figure. 3a the trace wiggled around the letter 'Z' which indicates that the participant was exploring the keyboard to find the 'X' key. Near the 'M' key in Figure 3b and the 'K' key in Figure 3c, we can observe a pattern of changing gliding directions. This pattern can be utilized to improve gesture decoding performance.

3.4.5 User Feedback.—Some participants (ID: P2, P3, P8, P9) expressed a positive reception for gesture typing and agreed that they could type faster after more practice. We identified three main reasons based on their comments: First, gliding fingers on the screen is natural and saves effort lifting the finger (P3: "I like the way sliding the finger to type"). Second, there is less pressure for the eyes to recognize the keys and locate the finger on the keyboard for they have the information about the relative positions of keys (P9: "Gliding the fingers is easier for me and my eyes feel relieved"). Lastly, the practice of gesture typing instills knowledge of word shapes in the participants' minds, thus it becomes easier for them to correct the errors (P8: "I can build Confidence in the gesture typing process"). There is one interesting comment from P10 that gesture typing can improve the ability of spelling. These comments highlight the potential importance of gesture typing as an option for text entry for individuals with low vision.

3.5 Implications for Designing Keyboard for Low Vision People

The existing decoding algorithm (e.g. $SHARK^2$) assumes that the gesture input of a user approximates the predefined template of the intended word (i.e., the shape formed by connecting centers of keys of the intended word with straight lines), while low vision people's gesture often deviates from such a template. We identified the following major characteristics of gesture input for low vision people, and explained how we could improve the gesture typing technique to accommodate low vision people:

- *Inaccurate starting position*. We observed that it is hard for low vision users to accurately land the finger on the first character of a word to start the gesture. The decoding algorithm should relax the constraint that the gesture needs to start from a position near the first letter.
- *Wiggling in the gesture*. With a lack of visual feedback of where the finger lands on the screen, low vision people tend to wiggle the input finger near the intended key: they explore the region to confirm that the intended key was reached. Because of these wiggling behaviors, the gesture input from low vision people

deviates from the pre-defined template of the intended word, which causes poor gesture recognition results. We suggest leveraging such wiggling behavior to improve the gesture recognition accuracy when designing the decoding algorithm for low vision people.

- *Pauses on intended keys*. Low vision users usually dwell longer at an intended key. Some possible reasons for such pause behavior are: (1) They need to wait for the auditory feedback which indicates the gesture reached the desired key. (2) They need to figure out in which direction to move based on their memory of the keyboard layout.
- *Cannot use the magnifier in gesture typing*. As previously explained, gesture typing requires a user to enter one word with one continuous gesture, and it does not work with a screen magnifier because some keys will be displayed outside the current viewport once the keyboard is magnified. The keyboard should provide a mechanism that allows users to reach these inaccessible keys in the middle of gesture typing.
- *Suggestion bar is under-utilized*. Participants rarely used the suggestion bar of the keyboard in the study. They commented that the current keyboard interface required them to switch the input focus to the suggestion bar to use it, which broke their interaction flow so they rarely used it.

4 ACCESSIBLE GESTURE TYPING KEYBOARDS FOR PEOPLE WITH LOW VISION

Based on the results obtained from user study I, we created two accessible gesture typing keyboards for people with low vision. Our keyboards contribute to both interface design and decoding algorithm design. Our keyboards allow users to access all keys when the magnifier is turned on. we designed and implemented a new decoder (kinematics-based decoder) that decodes the gesture based on kinematic features of gesture input of low vision people.

4.1 Keyboard Interface Design

We developed two keyboards, a layout-magnified keyboard and a key-magnified keyboard, to address a key issue identified in our formative user study: the current on-screen magnifier sometimes pushes keys out of the magnified viewport, making them inaccessible. The layout-magnified keyboard automatically shifts the keyboard position to display keys that are outside the viewport. In the key-magnified keyboard, all keys remain accessible because the layout is not magnified. Both designs aim to enhance accessibility, with the key-magnified keyboard providing a consistent layout that prevents any key from becoming unreachable.

4.1.1 Layout-magnified Keyboard.—We first designed a keyboard with a magnified layout that follows the movement of the typing finger. As shown in Figure 4b, a touchdown event anywhere on the screen initiates a zoom-in effect of the keyboard centered at the touch point. The keyboard continuously translates its location and moves towards the movement direction of the user’s finger. The zoom-in effect disappears when the user lifts the finger up.

4.1.2 Key-magnified Keyboard.—

This keyboard magnified the key underneath the input finger. As shown in Figure 4c, this keyboard has a static layout and provides an overlay magnifier following the movement of the gliding finger. The magnifier is located at the upper-left of the intended letter, intentionally avoiding any overlap. If the input finger reaches the left edge of the keyboard, the magnifier's placement shifts to the upper-right region.

4.1.3 Suggestions.—

Picking the intended word from the suggestions can save effort from retyping it. The participants in user study I hardly used the suggestion bar due to the lack of accessibility. To address this problem, we show the top suggestion with an enlarged font (Figure 4d) and three suggestion words below it. The magnifier also works for the suggestion bar.

We also show the just typed word at the same position with the top suggestion so that users could check what has been inputted.

4.1.4 Voice Feedback.—

Our keyboards also support voice feedback. Specifically, the keyboard reads out every key underneath the finger. When the user lifts the finger from the screen to complete a gesture typing, it also provides voice feedback for what is just typed. When the user explores the suggestion bar, the keyboard reads out the word underneath the finger as well.

4.2 Kinematics-based Decoding Algorithm for Low Vision People

We propose a kinematics-based decoding algorithm that determines the probability of entering a letter by examining kinematic features of gesture such as whether the gesture wiggles near a letter, whether the gesture pauses near a letter, and how long the pause is, etc. Even though there are some widely used decoders such as SHARK^2 [19], they are unsuitable for our particular scenario. They overlook certain distinctive input patterns exhibited by low vision users, as enumerated in Section 3.5: (1) ambiguous gesture starting position, (2) wiggling in the gesture, (3) pause in the gesture. Our design also drew inspiration from the GlanceWriter [9] which used probabilistic methods to infer the typed letter from the trajectory of the gaze. Our algorithm stands apart by specifically addressing the finger-based gesture input behavior (as mentioned above) exhibited by low vision users, in contrast to the approaches in GlanceWriter that were tailored for gaze-based input.

4.2.1 Data Structure.—

We used a Trie [40] (prefix tree) as the data structure to store all the words in the lexicon with a size of $10k$. The root of Trie is an empty node, and each of the other nodes in the Trie represents a character. Each node has no more than 26 children as there are 26 English letters. Each leaf node within the structure retains both the word and its concluding character. In instances where a word includes successive identical characters, these characters are symbolized using a solitary node. The children of a given node share a common prefix, encompassing the path from the root node to their immediate parent node.

In the Trie structure, nodes possess two states: RELEASE and HOLD. RELEASE means that the finger did not engage the corresponding key of the node, while HOLD signifies engagement. The initial state of the root node is HOLD. When a touch point hypothetically

occurs within the key i, the decoder checks all the HOLD nodes' children nodes: If the character stored in a HOLD node's child node is identical to key i, the HOLD node's child node is also transitioned to HOLD. The decoder computes the key score for all nodes in the HOLD state and incorporates the words stored within such nodes, if present, into the output candidate set.

4.2.2 Key Score.—Each node in Trie is associated with a key score, which considers the pausing and wiggling behavior. Let $K(i)$ denote the key score of key i, representing how likely i is the desired key. To obtain $K(i)$, we define $k(i, p)$ as the key score for a given touch point p at key i, where the touch point is sampled from the gesture trajectory at a certain frequency by the smartphone operating system. In our algorithm, $k(i, p)$ contains three components: (1) distance score $D(i, p)$, (2) pause score $P(i, p)$, (3) wiggle score $W(i, p)$. Specifically, they are computed as follows:

Distance score $D(i, p)$.: The distance score captures the space likelihood of a key i being the target given a touch point p. We assume that the distance d from p to i's key center follows a Gaussian distribution. Then, we define the distance score using the Gaussian probability density function as follows:

$$D(i, p) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(d-\mu)^2/2\sigma^2}, \tag{1}$$

where we assume μ to be 0 and we empirically set σ to 0.4.

Pause score $P(i, p)$.: It measures how long a user spends on the key i till the touch point p.

$$P(i, p) = \frac{t_p - t_0}{N}, \tag{2}$$

where t_p denotes the timestamp for touch point p and t_0 represents the timestamp the finger left the most recent key distinct from i, measured in milliseconds. We empirically set N as 100 to adjust the metric. A prolonged pause suggests a higher likelihood that i is the target key.

Wiggle score $W(i, p)$.: It quantifies the extent to which a user's gesture path wiggles around key i at touch point p. We define $W(i, p)$ as the number of occurrences that a user's finger enters i in a L-size window (measured in the number of touch points) right before touch point p. In order to count the occurrences $W(i, p)$, we need to determine the time range, i.e. the L. We set L to

$$L = f \cdot \Delta t, \tag{3}$$

where f denotes the frequency at which touchpoints are sampled, Δt is the expected time that the finger stays on each key given the input phrases. Let S be the phrases (without space), and j be a character, we compute the Δt as follow:

$$\Delta t = \frac{\sum_{j \in S}\left(t_j^l - t_j^e\right)}{|S|} \tag{4}$$

where t_j^e represents the first time point that the user's finger enters the key j, and t_j^l indicates the time that the user's finger leaves the key. $|S|$ is the number of characters that the user inputted.

Once we have the three scores, we could compute $k(i, p)$ as the product of them, That is,

$$k(i, p) = D(i, p) \cdot P(i, p) \cdot W(i, p). \tag{5}$$

The key score for a key i is defined as the maximum value of $k(i, p)$ among all touch points within the boundary of (i).

$$K(i) = \max_{p \in P} k(i, p), \tag{6}$$

where P contains all touch points that are located inside the boundary of key i.

4.2.3 Word Score.—The word score $S(w)$ (represents how likely w is the intended word given the input gesture path P. It is the sum of key scores for w along its path in the Trie:

$$S(w) = \sum_{i \in w} K(i). \tag{7}$$

4.2.4 Incorporating Langauge Model.—We followed the basic principle of word-gesture decoding [19] to incorporate a language model into decoding. For a word w in the candidate set, the principle combines the probability estimated from the input gesture path, denoted as $c(w)$, with the language model probability, denoted as $l(w)$, to obtain the probability that w being the intended word given input P, denoted by $Score(w)$:

$$Score(w) = \frac{l(w)c(w)}{\sum_{i \in W} l(i)c(i)}, \tag{8}$$

where W is a lexicon that contains i words and $c(w)$ is approximated using $S(w)$. Let C consist of top t ($t = X$) words with the highest word score in the lexicon, then

$$c(w) = \frac{S(w)}{\sum_{k \in C} S(k)}. \tag{9}$$

We employed a trigram language model to obtain $l(w)$, which is trained on the Corpus of Contemporary American English (COCA) [12] (2012 to 2017). Our decoding algorithm computes $Score(w)$ and outputs the top N words with the highest word score.

5 USER STUDY II: EVALUATING LAYOUT-MAGNIFIED AND KEY-MAGNIFIED KEYBOARDS

We conducted an IRB-approved study to evaluate the performance of our layout-magnified and key-magnified keyboards, powered by the kinematics-based decoding algorithm, in transcription tasks. We compared their performance to conventional keyboards in tap and gesture typing modes.

5.1 Participants and Apparatus

Twelve participants (8 females, 4 males) with low vision diagnoses were recruited from a non-profit vision and healthcare organization. Their ages ranged from 35 to 76 years old (average 55.1 ± 14.9). Among them, eight participants were new and four participants (P1, P2, P5, P7) were from user study I. The study adhered to ethical guidelines with IRB approval and informed consent from all participants. Table 3 shows the demographic information of the participants.

In the user study, 10 users used their index finger to perform the gesture input, while two users used their thumb. Based on their self-reported typing skills, 2 were experts and 10 were intermediates. In terms of self-reported familiarity with the QWERTY layout, 10 were experts and two were intermediates. Thus, all participants had prior knowledge of tap typing the QWERTY layout.

We used a Google Pixel 6 smartphone running Android 14 with a 6.4-inch 2400×1080 pixels display in this study. We developed a web-based text entry system with accessibility features, and the participants accessed the system using the Chrome browser on the provided smartphone.

5.2 Experiment Design

We employed a within-subjects design with one independent variable: keyboard type. This variable had four levels: layout-magnified keyboard, key-magnified keyboard, conventional gesture typing keyboard, and conventional tap typing keyboard. Similarly to user study I, we modified the conventional gesture typing keyboard by integrating voice feedback. It is implemented with a SHARK^2 decoder and the same trigram language model as our kinematics-based algorithm described in Section 4.2.4. The conventional tap typing keyboard uses the TalkBack default method and has two-step executions for text input. The

first tap or swipe explores the keyboard to locate the intended letter, and the second step requires the participant to double-tap anywhere on the screen to confirm the letter.

Similarly to user study I, this study also used a text transcription task. We only sampled two phrases for the warm-up session and five phrases for the formal session. The typing interface is shown in Figure 4a. To control for order effects, participants were evenly divided into two conditions: starting with the tap typing keyboard or the gesture typing keyboards. The order of the gesture typing keyboards was further counterbalanced within each group using a Latin Square design [4].

The default magnification levels of the layout-magnification keyboard and key-magnified keyboard are 2X and 70px respectively. Users may adjust that based on their preferences. All participants used the default values, except that one user (P10) adjusted the value for the key-magnified keyboard to 90px. Our system logged the following information: timestamp, gesture trace, and current texts while typing.

5.3 Procedure

The user study procedure is similar to the user study I and contains a warm-up session, where we introduced the keyboard and participants practiced typing with two phrases, and a formal session, where users typed five phrases using each keyboard.

Participants could have a 5-minute or longer break if needed between keyboards. After each keyboard task, a post-study interview using the System Usability Scale (SUS) questionnaire [7] evaluated usability perceptions. The user study lasts around 2 hours.

5.4 Results

All 12 participants successfully completed typing five phrases on each of the four keyboards during the formal study session. We observed that 10 participants used the color inversion option, while all participants enabled voice feedback throughout the study.

5.4.1 Input Speed.—We first investigated input speed measured in WPM [23]. As shown in Figure 6, the key-magnified keyboard yielded the fastest typing speed, while the tap typing keyboard resulted in the slowest WPM.

A repeated-measures ANOVA revealed a significant main effect of keyboard layout on typing speed ($F_{3, 33} = 16.24$, $p < 0.001$). Pairwise mean comparisons with Holm's adjustment showed significant differences in WPM between key-magnified keyboard vs. conventional gesture typing keyboard ($p < 0.001$), layout-magnified keyboard vs. tap typing keyboard ($p < 0.001$), key-magnified keyboard vs. tap typing keyboard ($p < 0.001$), and conventional gesture typing keyboard vs. tap typing keyboard ($p = 0.03$). However, no significant differences were found between layout-magnified keyboard vs. key-magnified keyboard ($p = 0.13$), layout-magnified keyboard vs. conventional gesture typing keyboard ($p = 0.24$). These results suggest that the keyboard layout significantly impacted typing speed, with the key-magnified keyboard leading to faster performance than others. This highlights the importance of considering visual acuity limitations when designing gesture typing interfaces for low vision users. Interestingly, the WPM between the layout-magnified

and the conventional gesture typing keyboard was not statistically significant. This could indicate that for some users with low vision, the familiarity and established motor skills associated with the static layout of the conventional gesture typing keyboard might be comparable to the benefits of a magnified layout.

5.4.2 Error Rate.—We investigated the WER [2] using Minimum Word Distance (MWD) for each keyboard condition. As shown in Figure 7, the layout-magnified keyboard and the key-magnified keyboard resulted in lower WER compared to the conventional gesture typing keyboard and the tap typing keyboard. This suggests that the magnified keyboard layouts might have contributed to fewer errors for participants with low vision.

A repeated measures ANOVA showed a significant main effect for the keyboard layout ($F_{3, 33} = 11.7$, $p < 0.001$ on WER. Pairwise mean comparisons with Holm adjustment showed) the differences were significant for layout-magnified keyboard vs. tap typing keyboard ($p = 0.007$), key-magnified keyboard vs. tap typing keyboard ($p = 0.003$). However, results are not significant for key-magnified keyboard vs. conventional gesture typing keyboard ($p = 0.11$), key-magnified keyboard vs. layout-magnified keyboard ($p = 0.17$), layout-magnified keyboard vs. conventional gesture typing keyboard ($p = 0.22$), and conventional gesture typing keyboard vs. tap typing keyboard ($p = 0.11$). It suggests that the magnified gesture typing keyboards(layout-magnified and key-magnified) with our kinematics-based decoding algorithm can improve typing accuracy by reducing errors over the tap typing keyboard. We can observe that the error rates between the magnified layouts (key-magnified and layout-magnified) and the conventional gesture typing keyboard were not statistically significant. This indicates that for some users, gesture-based interaction with a standard keyboard layout might be just as accurate as using magnified versions.

5.4.3 Deletes Per Word.—We then analyzed the backspace usage using the metric of Deletes Per Word (DPW). As shown in Figure 8, the key-magnified keyboard has the lowest DPW. Conversely, the conventional gesture typing keyboard showed the highest DPW.

A repeated measures ANOVA showed significant main effects for the keyboard layout ($F_{2, 22} = 6.1$, $p < 0.001$) on the DPM. Pairwise mean comparisons with Holm adjustment showed a significant reduction in DPW for the key-magnified keyboard vs. the conventional gesture typing keyboard ($p < 0.05$). However, no statistically significant differences in DPW were found between others: layout-magnified keyboard vs. conventional gesture typing keyboard ($p = 0.11$), key-magnified keyboard ($p = 0.15$) vs. layout-magnified keyboard ($p = 0.20$), conventional gesture typing keyboard vs. tap typing keyboard ($p = 0.05$), layout-magnified keyboard vs. tap typing keyboard ($p = 0.87$), and key-magnified keyboard vs. tap typing keyboard ($p = 0.87$).

5.4.4 Usability.—As shown in Figure 9, the key-magnified keyboard received the highest SUS score and the tap typing keyboard received the lowest. While the results suggest a trend where participants favored the key-magnified and layout-magnified keyboards, a Friedman test revealed no statistically significant main effect of keyboard layout on SUS scores (χ^2 (2) = 7.25, $p = 0.06$).

5.4.5 Evaluation by Expertise and Prior Participation.—

We analyzed the typing performance based on users' expertise of typing (Expert vs. Non-expert) and prior participation in the formative study (New Users vs. Old Users) respectively. As shown in Table 4, the two magnification-based keyboards benefited both types of user, especially for non-expert users. New users tend to have slower typing speeds and higher error rates compared to old users who participated in our first study. This suggests that experience over time leads to improved typing performance.

6 DISCUSSION

The comparison between the key-magnified keyboard and the conventional gesture typing keyboard showed that the key-magnified keyboard outperforms the conventional gesture typing keyboard for low vision people. The key-magnified keyboard achieved a higher input speed in terms of WPM. Specifically, the key-magnified keyboard improved the input speed of the conventional gesture typing keyboard by 1.14 WPM (27.5%). While the layout-magnified keyboard did not demonstrate a statistically significant improvement in WPM compared to the conventional option, it may still offer benefits for a specific subset of low vision users. Users who prefer a deliberate typing style may find the layout-magnified keyboard advantageous, while those accustomed to swift gliding gestures may require additional adaptation time.

Our study also showed that low vision people also performed better on the gesture typing keyboards than in tap typing keyboard. The input speed of the tap typing keyboard is 2.97 WPM (SD=1.12) and the WER is 0.24 (SD=0.13). Our layout-magnified and key-magnified keyboards demonstrated notable enhancements in input speed when compared to the tap typing keyboard, with improvements of 1.61 WPM (54.2%) and 2.31 WPM (77.6%) respectively. Additionally, our layout-magnified and key-magnified keyboards exhibited enhancements in WER for the tap typing keyboard by 0.17 (70.3%) and 0.19 (78.6%) respectively.

Furthermore, SUS scores showed a trend in which layout-magnified and key-magnified keyboards received higher ratings compared to conventional gesture typing and tap typing keyboards. In line with this observation, some participants from user study II commented "It is helpful that I can actually see what I'm typing" (P1), "It is faster to type with gesture typing and the keyboards are easy to use"(P11). It is important to note that user experiences varied. There are some participants who had difficulty adapting to the proposed keyboards. Participant P10 in Table 3 expressed his preference for the tap typing keyboard, "I'm used to using tap typing and I feel more confident to type the letters one by one". These findings highlight the need to consider user experience and existing typing habits when designing keyboards for low vision users. Although magnified keyboards showed promise, some users might prefer established methods such as tapping typing.

User study II revealed some potential issues with the current keyboard designs. Participant P6's comment suggests that the layout-magnified keyboard might lead to occasional difficulties in maintaining spatial context on the magnified layout. This could be because users might lose track of their finger's position relative to the broader keyboard layout while

focusing on the magnified area. On the other hand, especially for long words, low vision users sometimes lift their finger before completing the word gesture because of fatigue or context loss. Then they need to delete the inputted words and re-input the expected word, which is time consuming. We believe that a decoder that supports inputting text using multi-segment of gestures would solve this problem and would be an interesting direction for future work.

7 LIMITATION AND FUTURE WORK

Inclusion Criteria

Low vision encompasses a wide range of conditions and there can be considerable variations in the severity and characteristics of the visual condition. Our study focuses on low vision people who can utilize the screen magnifier on smart-phones. This criterion ensures that participants can see the keys and content after triggering the screen magnifiers. Consequently, this excludes users with more severe vision impairments who may rely on alternative text entry methods such as voice input or Braille input. The proposed techniques may benefit other people with visual impairments because it can handle gestures deviating from pre-defined templates. However, further research is required to fully understand how to accommodate other low-vision users.

Heterogeneity of Low Vision

Low vision people have different challenges when writing on smartphone keyboards based on their specific condition. For example, wet macular degeneration can cause significant field loss and the vision may appear distorted, while dry macular degeneration might lead to a blurry or dark spot in the center of the vision field. To address this diversity, our research definition of “low vision” prioritizes functional limitations over specific diagnoses. It is important to investigate how to account for the unique challenges of individuals in keyboard interface and decoding algorithm design. For example, if a user can only see the center of the view field, the interface may use a permanently centered magnifier to keep crucial information in view. This exemplifies the potential of personalization, a promising direction for future research to improve user experience.

Learning Effects

In the user study, participants practiced with each keyboard using only two phrases. Since all participants practiced the same number of phrases, the impact on the conclusion of the input speed during the formal study was minimal. Further longitudinal studies are needed to fully understand and account for the effects of learning.

8 CONCLUSION

We have designed and implemented layout-magnified and key-magnified keyboard prototypes to enable gesture typing for people with low vision. The layout-magnified keyboard will automatically pan the keyboard layout to reveal hidden keys once the input finger reaches the edge of the viewport, and the key-magnified keyboard magnifies the key underneath the input finger only. Both keyboards make all the keys accessible during the

input, thus making the magnifier compatible with gesture typing. Furthermore, a kinematics-based decoding algorithm was developed to accommodate the unique typing patterns of users with low vision, even when gestures deviate from predefined templates or starting positions. User study results demonstrate that both keyboards led to improvements in typing speed and error reduction compared to the conventional tap typing keyboard. moreover, The key-magnified keyboard increased typing speed by 27.5% compared to a conventional gesture typing keyboard.

ACKNOWLEDGMENTS

We sincerely thank our anonymous reviewers for their insightful comments and suggestions. This work was supported by Google Inclusion Research Award, NSF Awards 2153056 and 2113485, and NIH Awards R01EY030085 and R01EY035688.

REFERENCES

[1]. Azenkot Shiri and Nicole B Lee. 2013. Exploring the use of speech input by blind people on mobile devices. In Proceedings of the 15th international ACM SIGACCESS conference on computers and accessibility. 1–8.

[2]. Bi Xiaojun, Azenkot Shiri, Partridge Kurt, and Zhai Shumin. 2013. Octopus: evaluating touchscreen keyboard correction and recognition algorithms via “Remulation”. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 543–552.

[3]. Bi Xiaojun, Chelba Ciprian, Ouyang Tom, Partridge Kurt, and Zhai Shumin. 2012. Bimanual gesture keyboard. In Proceedings of the 25th annual ACM symposium on User interface software and technology. 137–146.

[4]. Bi Xiaojun, Smith Barton A, and Zhai Shumin. 2012. Multilingual touchscreen keyboard design and optimization. Human–Computer Interaction 27, 4 (2012), 352–382.

[5]. Syed Masum Billah Yu-Jung Ko, Ashok Vikas, Bi Xiaojun, and Ramakrishnan IV. 2019. Accessible gesture typing for non-visual text entry on smartphones. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 1–12.

[6]. Bonner Matthew N, Brudvik Jeremy T, Abowd Gregory D, and Edwards W Keith. 2010. No-look notes: accessible eyes-free multi-touch text entry. In International Conference on Pervasive Computing. Springer, 409–426.

[7]. Brooke John. 1996. Sus: a “quick and dirty’usability. Usability evaluation in industry 189, 3 (1996).

[8]. Chen Xiang’Anthony’, Grossman Tovi, and Fitzmaurice George. 2014. Swipeboard: a text entry technique for ultra-small interfaces that supports novice to expert transitions. In Proceedings of the 27th annual ACM symposium on User interface software and technology. 615–620.

[9]. Cui Wenzhe, Liu Rui, Li Zhi, Wang Yifan, Wang Andrew, Zhao Xia, Rachidian Sina, Baig Furqan, Ramakrishnan IV, Wang Fusheng, and Bi Xiaojun. 2023. GlanceWriter: Writing Text by Glancing Over Letters. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. 1–13.

[10]. Cui Wenzhe, Zheng Jingjie, Lewis Blaine, Vogel Daniel, and Bi Xiaojun. 2019. HotStrokes: Word-gesture shortcuts on a trackpad. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 1–13.

[11]. Cui Wenzhe, Zhu Suwen, Li Zhi, Xu Zheer, Yang Xing-Dong, Ramakrishnan IV, and Bi Xiaojun. 2021. BackSwipe: Back-of-device Word-Gesture Interaction on Smartphones. In In Proceedings of CHI 2021 - the SIGCHI Conference on Human Factors in Computing Systems.

[12]. Davies Mark. 2008. The corpus of contemporary American English: 1990-present.

[13]. Franz Rachel L, Wobbrock Jacob O, Cheng Yi, and Findlater Leah. 2019. Perception and adoption of mobile accessibility features by older adults experiencing ability changes. In Proceedings of the 21st International ACM SIGACCESS Conference on Computers and Accessibility. 267–278.

[14]. Frey Brian, Southern Caleb, and Romero Mario. 2011. Brailletouch: mobile texting for the visually impaired. In International Conference on Universal Access in Human-Computer Interaction. Springer, 19–25.

[15]. Grussenmeyer William and Folmer Eelke. 2017. Accessible touchscreen technology for people with visual impairments: a survey. ACM Transactions on Accessible Computing (TACCESS) 9, 2 (2017), 1–31.

[16]. Guerreiro João, Rodrigues André, Montague Kyle, Guerreiro Tiago, Nicolau Hugo, and Daniel Gonçalves. 2015. TabLETS get physical: non-visual text entry on tablet devices. In Proceedings of the 33rd annual acm conference on human factors in computing systems. 39–42.

[17]. Kane Shaun K, Wobbrock Jacob O, and Ladner Richard E. 2011. Usable gestures for blind people: understanding preference and performance. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 413–422.

[18]. Komninos Andreas, Stefanis Vassilios, and Garofalakis John. 2023. A Review of Design and Evaluation Practices in Mobile Text Entry for Visually Impaired and Blind Persons. Multimodal Technologies and Interaction 7, 2 (2023), 22.

[19]. Kristensson Per-Ola and Zhai Shumin. 2004. SHARK2: a large vocabulary short-hand writing system for pen-based computers. In Proceedings of the 17th annual ACM symposium on User interface software and technology. 43–52.

[20]. Kumar Anuj, Paek Tim, and Lee Bongshin. 2012. Voice typing: a new speech interaction model for dictation on touchscreen devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2277–2286.

[21]. Lin Yu-Hao, Zhu Suwen, Ko Yu-Jung, Cui Wenzhe, and Bi Xiaojun. 2018. Why is gesture typing promising for older adults? comparing gesture and tap typing behavior of older with young adults. In Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility. 271–281.

[22]. Lu Yiqin, Yu Chun, Fan Shuyi, Bi Xiaojun, and Shi Yuanchun. 2019. Typing on Split Keyboards with Peripheral Vision. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 1–12.

[23]. Scott MacKenzie I. 2002. A Note on Calculating Text Entry Speed. http://www.yorku.ca/mack/RN-TextEntrySpeed.html.

[24]. Markussen Anders, Jakobsen Mikkel Rønne, and Hornbæk Kasper. 2014. Vulture: a mid-air word-gesture keyboard. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 1073–1082.

[25]. Mascetti Sergio, Bernareggi Cristian, and Belotti Matteo. 2011. TypeInBraille: a braille-based typing application for touchscreen devices. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility. 295–296.

[26]. Nuance. 2023. Dragon Naturally Speaking Software. https://www.nuance.com/dragon.html. [Online; accessed April 05, 2023].

[27]. Oliveira João, Guerreiro Tiago, Nicolau Hugo, Jorge Joaquim, and Gonçalves Daniel. 2011. Blind people and mobile touch-based text-entry: acknowledging the need for different favors. In The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility. 179–186.

[28]. Oliveira João, Guerreiro Tiago, Nicolau Hugo, Jorge Joaquim, and Daniel Gonçalves. 2011. BrailleType: unleashing braille over touch screen mobile phones. In IFIP Conference on Human-Computer Interaction. Springer, 100–107.

[29]. Oney Stephen, Harrison Chris, Ogan Amy, and Wiese Jason. 2013. ZoomBoard: a diminutive qwerty soft keyboard using iterative zooming for ultra-small devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2799–2802.

[30]. Rakhmetulla Gulnar and Arif Ahmed Sabbir. 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. 1–13.

[31]. Samanta Debasis and Chakraborty Tuhin. 2020. VectorEntry: Text Entry Mechanism Using Handheld Touch-Enabled Mobile Devices for People with Visual Impairments. ACM Transactions on Accessible Computing (TACCESS) 13, 3 (2020), 1–29.

[32]. Sánchez Jaimeand Aguayo Fernando. 2007. Mobile messenger for the blind. In Universal access in ambient intelligence environments. Springer, 369–385.

[33]. Shi Weinan, Yu Chun, Fan Shuyi, Wang Feng, Wang Tong, Yi Xin, Bi Xiaojun, and Shi Yuanchun. 2019. VIPBoard: Improving Screen-Reader Keyboard for Visually Impaired People with Character-Level Auto Correction. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 1–12.

[34]. Stefanis Vassilios, Komninos Andreas, and Garofalakis John. 2020. Challenges in Mobile Text Entry using Virtual Keyboards for Low-Vision Users. In 19th International Conference on Mobile and Ubiquitous Multimedia. 42–46.

[35]. Szpiro Sarit Felicia Anais, Hashash Shafeka, Zhao Yuhang, and Azenkot Shiri. 2016. How people with low vision access computing devices: Understanding challenges and opportunities. In Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility. 171–180.

[36]. TalkBack. 2009 [n.d.]. TalkBack: An Open Source Screenreader For Android. https://opensource.googleblog.com/2009/10/talkback-open-source-screenreader-for.html [Online; accessed April 05, 2023].

[37]. Tinwala Hussain and MacKenzie I Scott. 2009. Eyes-free text entry on a touchscreen phone. In 2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH). IEEE, 83–88.

[38]. Vertanen Keith and Kristensson Per Ola. 2011. A versatile dataset for text entry evaluations based on genuine mobile emails. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services. 295–298.

[39]. VoiceOver. 2009 [n.d.]. Screen reader from Apple. https://www.apple.com/accessibility/vision [Online; accessed April 05, 2023].

[40]. Wikipedia contributors. 2023. Trie — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Trie [Online; accessed 05-April-2023].

[41]. Wobbrock Jacob O 2007. Measures of text entry performance. Text entry systems: Mobility, accessibility, universality (2007), 47–74.

[42]. Yeo Hui-Shyong, Phang Xiao-Shen, Steven J Castellucci Per Ola Kristensson, and Quigley Aaron. 2017. Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. 4194–4202.

[43]. Yfantidis Georgios and Evreinov Grigori. 2006. Adaptive blind interaction technique for touchscreens. Universal Access in the Information Society 4, 4 (2006), 328–337.

[44]. Yi Xin, Yu Chun, Shi Weinan, Bi Xiaojun, and Shi Yuanchun. 2017. Word clarity as a metric in sampling keyboard test sets. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. 4216–4228.

[45]. Yu Chun, Gu Yizheng, Yang Zhican, Yi Xin, Luo Hengliang, and Shi Yuanchun. 2017. Tap, dwell or gesture? exploring head-based text entry techniques for hmds. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. 4479–4488.

[46]. Zhai Shumin and Kristensson Per-Ola. 2003. Shorthand writing on stylus keyboard. In Proceedings of the SIGCHI conference on Human factors in computing systems. 97–104.

[47]. Zhai Shumin and Kristensson Per Ola. 2012. The word-gesture keyboard: reimagining keyboard interaction. Commun. ACM 55, 9 (2012), 91–101.

[48]. Zhai Shumin, Per Ola Kristensson Pengjun Gong, Greiner Michael, Shilei Allen Peng Liang Mico Liu, and Dunnigan Anthony. 2009. Shapewriter on the iPhone: from the laboratory to the real world. In CHI'09 Extended Abstracts on Human Factors in Computing Systems. 2667–2670.

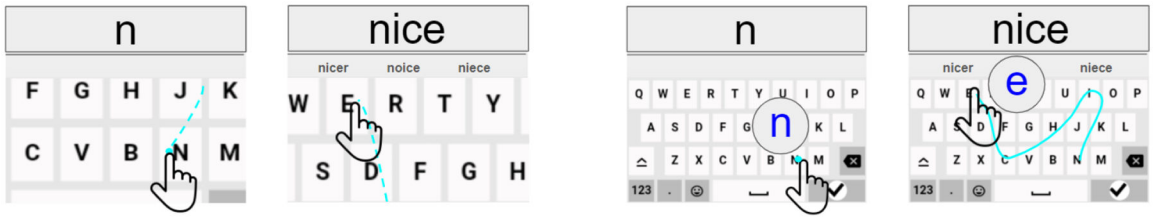
[49]. Zhao Yuhang, Szpiro Sarit, and Azenkot Shiri. 2015. Foresee: A customizable head-mounted vision enhancement system for people with low vision. In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility. 239–249.

[50]. Zhu Suwen, Luo Tianyao, Bi Xiaojun, and Zhai Shumin. 2018. Typing on an invisible keyboard. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. 1–13.

[51]. Zhu Suwen, Zheng Jingjie, Zhai Shumin, and Bi Xiaojun. 2019. i'sFree: Eyes-Free Gesture Typing via a Touch-Enabled Remote Control. In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. 1–12.

CCS CONCEPTS

- Human-centered computing → Accessibility technologies; Text input; Gestural input.



(a) (b) (c) (d)

Figure 1:

Typing “nice” with gesture on our layout-magnified and key-magnified keyboards. On the layout-magnified keyboard, the user (a) presses a finger near ‘N’ to initiate a zoom-in effect and swipes towards the subsequent letters, (b) finishes at the letter ‘E’ and lifts the finger. With the key magnifier keyboard, the user (c) starts near ‘N’, and then (d) swipes towards ‘I’, ‘C’, and ‘E’, with a magnifier following the finger. The top choice from the real-time decoder appears at the top of the keyboard in an enlarged font, followed by three suggestions in the suggestion bar. The light blue dots and traces denote touchpoints and gestures. Note that both keyboards do not require the user to start the gesture at the first letter of a word.

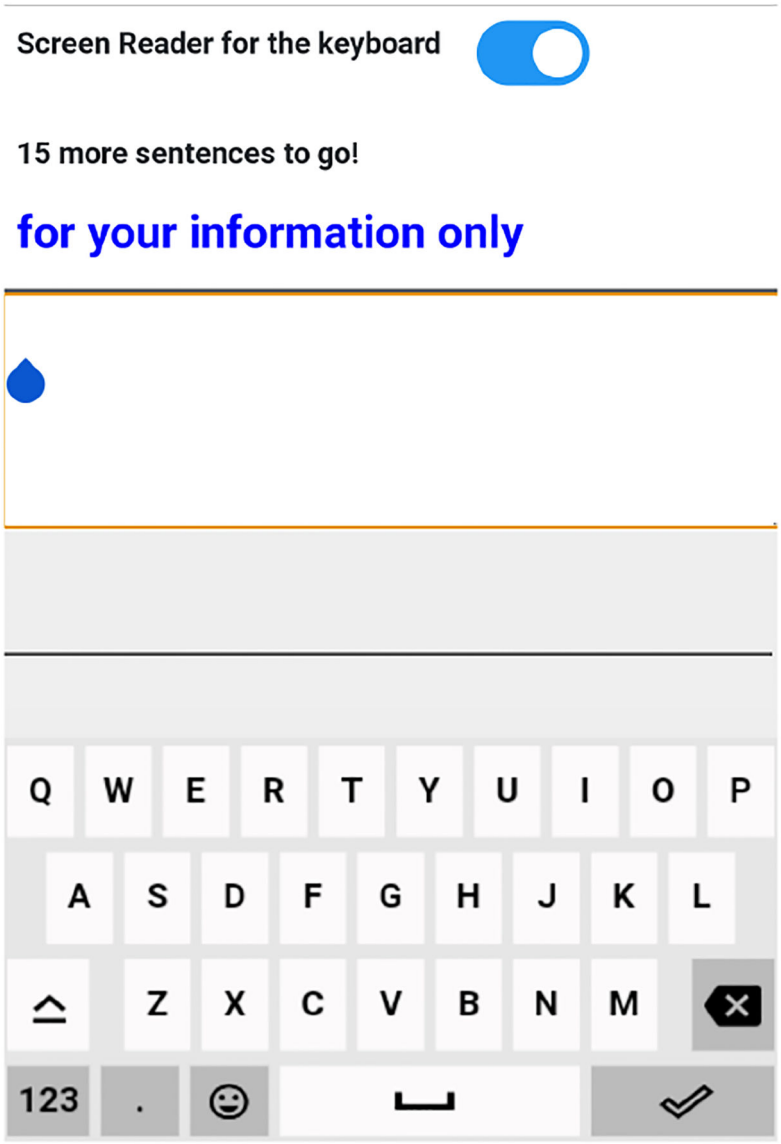
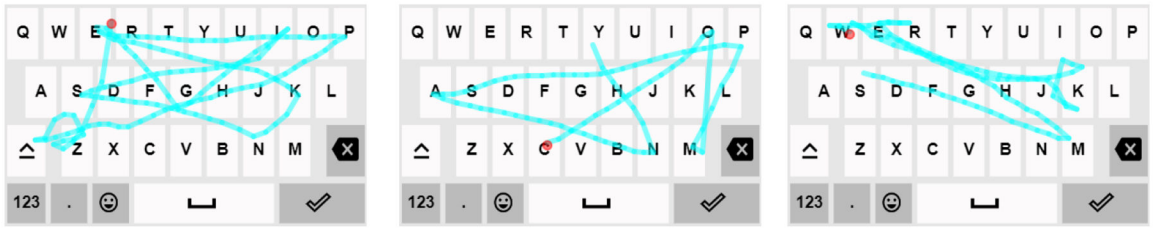


Figure 2:
The interface of the keyboard in User Study I.



(a) (b) (c)

Figure 3:
Sample gestures of low vision participants. The red dot indicates the starting point. (a) The intended word “expensive” was recognized as “expose”. The input finger wiggled around *z* when the user was searching for *x*. (b) The intended word “company” was successfully recognized. (c) The intended word “weekend” was recognized as “walgreens”. The input finger wiggled around *k* as the user was searching for this key.

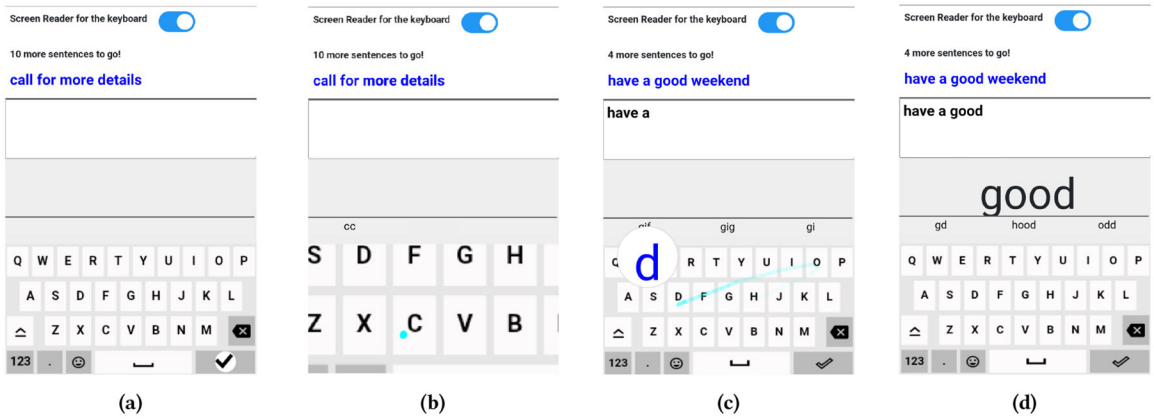


Figure 4:

Screenshots of our keyboards. (a) The interface of our text entry system. (b) The layout-magnified keyboard. When the user lands the finger on the key ‘C’, the keyboard initiates a zoom-in effect. (c) The key-magnified keyboard with the magnified key underneath the finger. (d) The top suggestion / just typed word is displayed with an enlarged font size.

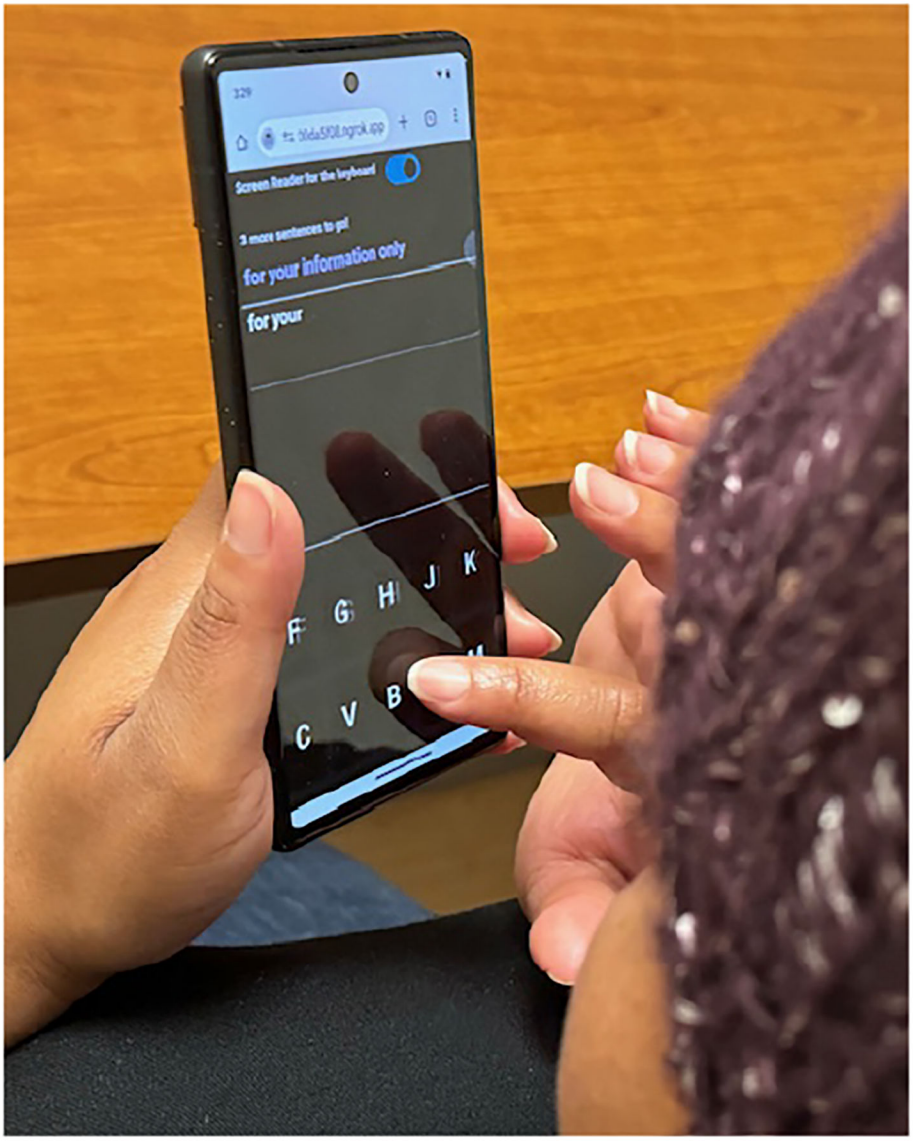


Figure 5:
A user is inputting text using the layout-magnified keyboard with color inversion on.

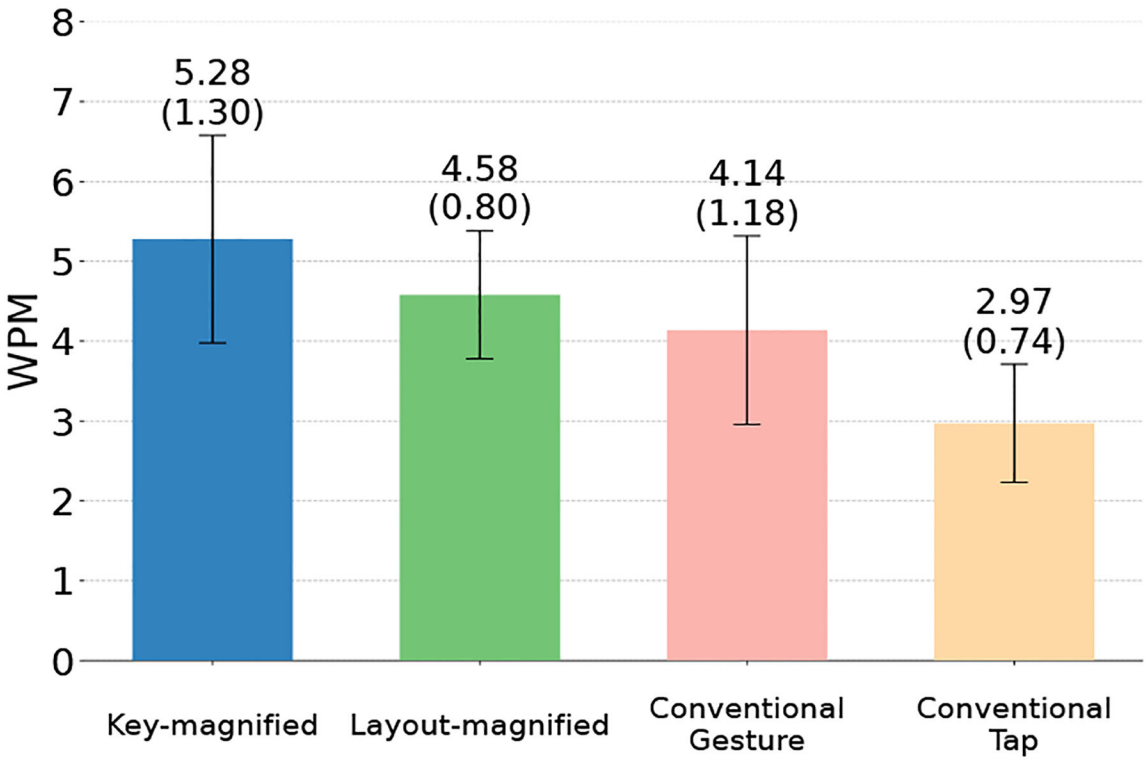


Figure 6:
Average input speed (95% Confidence Interval) of the four keyboards.

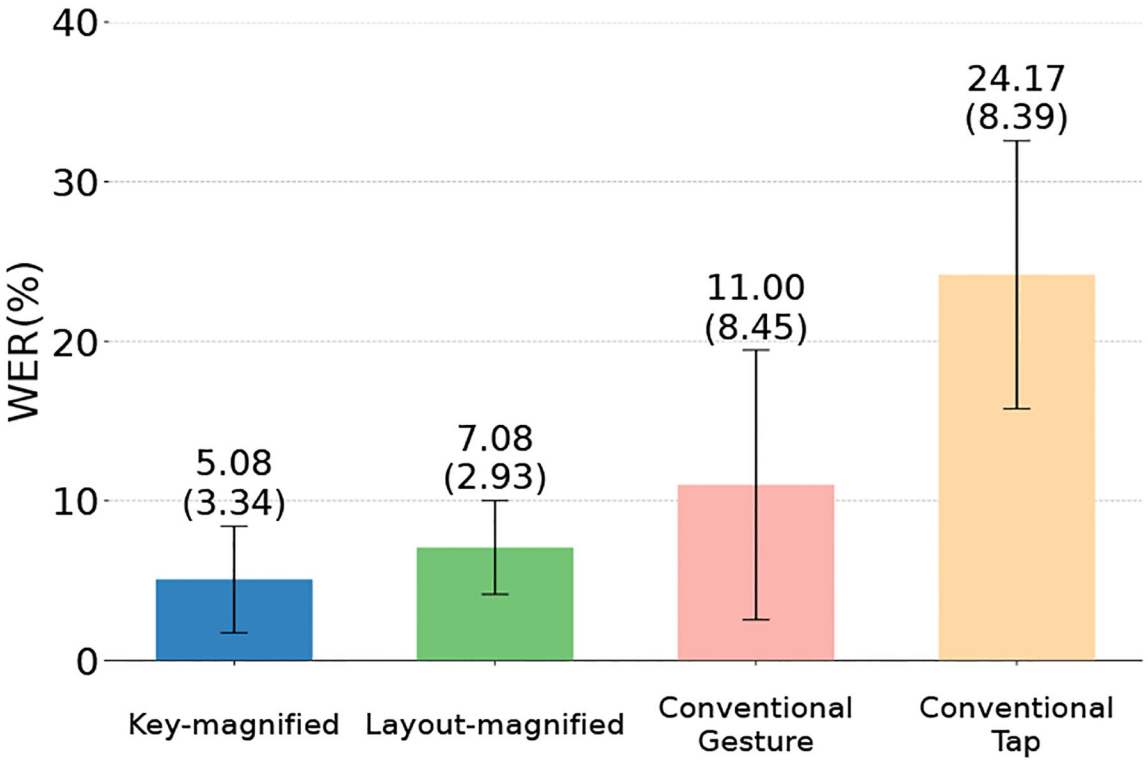


Figure 7:
Average WER (95% CI) of the four keyboards.

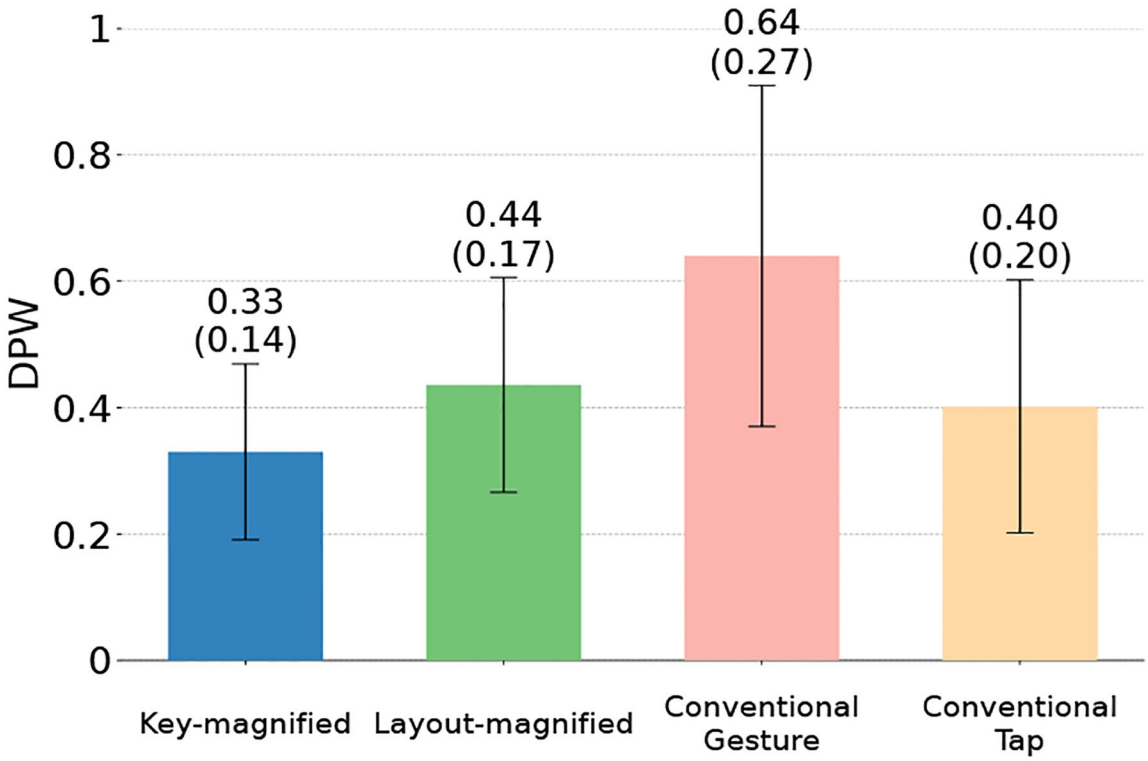


Figure 8:
Average Deletes Per Word (95% CI) of the four keyboards.

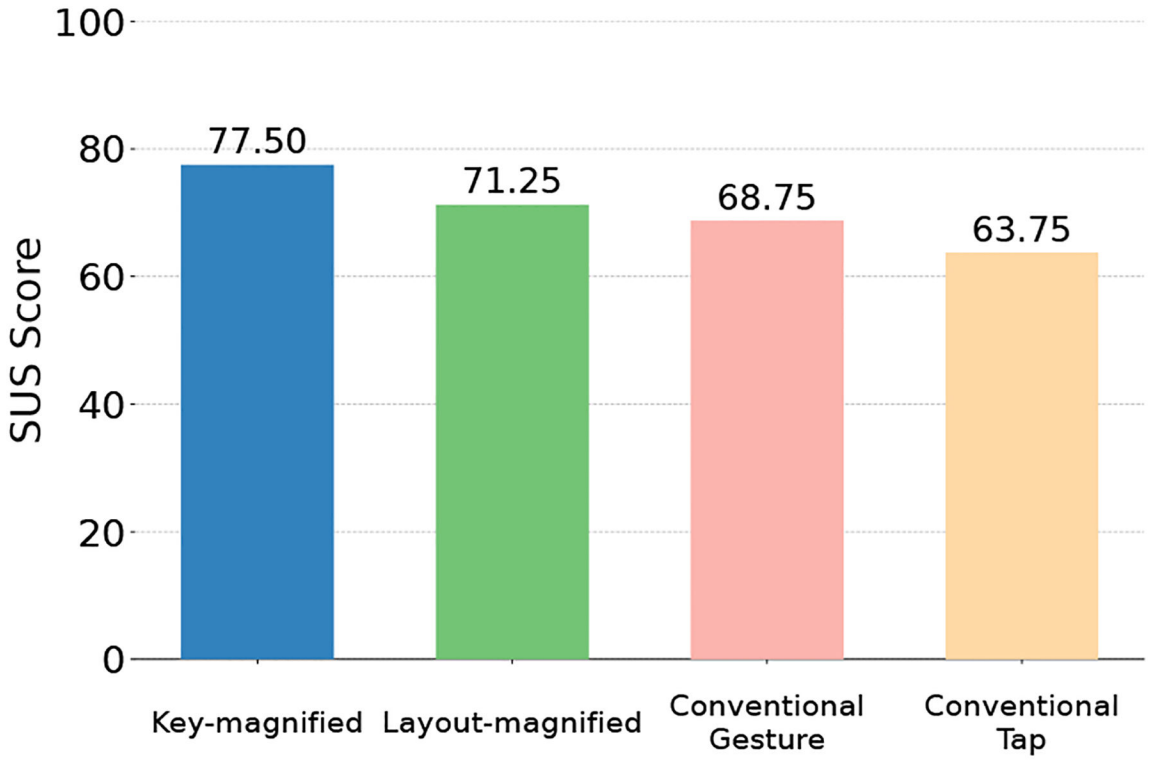


Figure 9:
The median SUS score of the four keyboards.

Table 1:

Demographic information of the 10 low vision participants and their accessibility settings (color inversion, voice feedback) used in user study I.

ID	Age	Gender	Diagnosis	Color Inversion	Voice Feedback
P1	49	F	Leber Congenital Amaurosis	No	Yes
P2	36	M	Optic Atrophy	No	Yes
P3	43	M	Retinitis Pigmentosa	No	Yes
P4	75	F	Diabetic Retinopathy	Yes	Yes
P5	39	M	Optic Atrophy	No	Yes
P6	66	F	Retinitis Pigmentosa	No	Yes
P7	71	F	Retinitis Pigmentosa	No	Yes
P8	69	F	Glaucoma	No	No
P9	75	F	Retinal Detachment	No	Yes
P10	56	M	Macular Degeneration	No	Yes

Proc ACM Symp User Interface Softw Tech. Author manuscript; available in PMC 2025 January 08.

Table 2:

Completed phrases of each participant in the user study I. Five participants were able to input all phrases, while four were unable to input a single phrase.

ID	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Completion Count	10	10	0	0	6	10	0	10	10	0

Table 3:

Demographic information of the 12 low vision participants.

ID	Age	Gender	Diagnosis	Typing Posture	Typing Expertise	Layout Familiarity
P1	51	F	Leber's Congenital Amaurosis	Index	Intermediate	Expert
P2	73	F	Glaucoma	Index	Intermediate	Intermediate
P3	66	F	Glaucoma	Index	Intermediate	Expert
P4	68	F	Congenital Cataracts	Index	Intermediate	Expert
P5	68	F	Retinitis Pigmentos	Index	Intermediate	Expert
P6	52	M	Congenital Cataracts	Index	Intermediate	Intermediate
P7	77	F	Glaucoma	Index	Intermediate	Expert
P8	42	M	Congenital Cataracts	Index	Intermediate	Expert
P9	36	F	Optic Neuritis	Thumb	Expert	Expert
P10	52	M	Optic Neuritis	Index	Intermediate	Expert
P11	41	M	Optic Atrophy	Index	Intermediate	Expert
P12	35	F	Glaucoma	Thumb	Expert	Expert

Proc ACM Symp User Interface Softw Tech. Author manuscript; available in PMC 2025 January 08.

Table 4:

Performance comparison of typing keyboards across different user groups: Expert vs. Non-expert, and New Users vs. Old Users.

Expert vs. Non-expert					
		Conventional Tap	**Conventional Gesture**	**Layout-magnified**	**Key-magnified**
Mean WPM	Expert	4.06	6.31	5.45	7.72
	Non-Expert	2.75	3.70	4.41	4.78
Mean WER (%)	Expert	13.5	7.0	5.0	2.5
	Non-Expert	26.3	11.8	7.5	5.6
New Users vs. Old Users					
		Conventional Tap	**Conventional Gesture**	**Layout-magnified**	**Key-magnified**
Mean WPM	Old	3.35	4.59	5.12	5.93
	New	2.21	3.21	3.50	3.96
Mean WER (%)	Old	20.0	10.2	7.1	4.3
	New	32.5	12.8	7.0	6.8