Typing on an Invisible Keyboard

Suwen Zhu[1] **Tianyao Luo**[1] **Xiaojun Bi**[1] **Shumin Zhai**[2]

[1]Department of Computer Science
Stony Brook University
Stony Brook, New York, USA
{suwzhu, tiluo, xiaojun}@cs.stonybrook.edu

[2]Google Inc.
Mountain View, California, USA
zhai@acm.org

ABSTRACT

A virtual keyboard takes a large portion of precious screen real estate. We have investigated whether an invisible keyboard is a feasible design option, how to support it, and how well it performs. Our study showed users could correctly recall relative key positions even when keys were invisible, although with greater absolute errors and overlaps between neighboring keys. Our research also showed adapting the spatial model in decoding improved the invisible keyboard performance. This method increased the input speed by 11.5% over simply hiding the keyboard and using the default spatial model. Our 3-day multi-session user study showed typing on an invisible keyboard could reach a practical level of performance after only a few sessions of practice: the input speed increased from 31.3 WPM to 37.9 WPM after 20 - 25 minutes practice on each day in 3 days, approaching that of a regular visible keyboard (41.6 WPM). Overall, our investigation shows an invisible keyboard with adapted spatial model is a practical and promising interface option for the mobile text entry systems.

ACM Classification Keywords

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

Author Keywords

Text entry; virtual keyboards; touchscreen.

INTRODUCTION

The HCI field has long aspired to realize *Invisible user interfaces* as an interaction paradigm. For example, Fishkin, Moran, and Harrison [9] envisioned that interaction paradigms would progress towards an ideal of the invisible user interface where there is no perceived mediation between users and computers, which mirror more real-world interactions. A sizable amount of research [18, 19, 20, 26] in line with this vision has been carried out, especially in the context of mobile computing.

A virtual keyboard is one of the most important user interfaces on mobile devices. In line with the invisible UI paradigm, reducing the keyboard's visibility is compelling for many reasons, including the following:

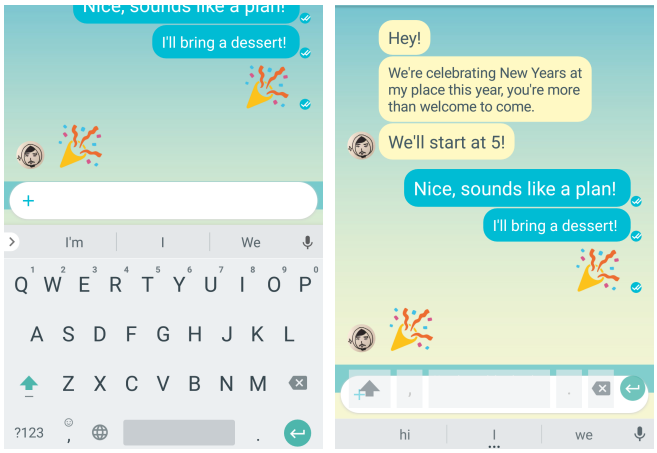
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-5620-6/18/04…$15.00

DOI: https://doi.org/10.1145/3173574.3174013



(a) (b)

Figure 1. (a) is a visible keyboard; (b) is an invisible keyboard, which saves screen real estate for displaying more conversation history in a chat application.

First, an invisible keyboard frees a large portion of screen real estate, which offers more space for displaying multimedia content and other UI space. A keyboard usually occupies more than 40% of screen real estate (e.g., Figure 1a), resulting in an unsatisfying user experience. Li et al.'s survey [27] over 50 iPad users showed that the users desired to minimize the space occupied by the virtual keyboard. In fact, their ratings of the iPad display size dropped from 5 (1: very unsatisfied; 5: very satisfied) to 4 when the keyboard was visible. One of the reasons, as commented by the users, was that the remaining screen real estate was too small to work on properly, and they had to scroll up and down all the time.

Second, typing on an invisible keyboard mirror how an expert (i.e., touch typist) types on a physical keyboard or a typewriter. On traditional typewriters, typing without looking at the keyboard is considered superior to "hunt-and-peck", because it allows the typist to focus attention on the composition of the text. Touch typing has been the standard typing method taught since at least the 1920's [22]. Typing on an invisible keyboard connects to such typing experience.

If an invisible keyboard is desirable, is it practical? We expected it would be extremely challenging, if not impossible. The input signals would be very noisy. Previous research has shown the imprecision of finger touch and small key sizes

already result in noisy input signals on a fully visible keyboard [2, 21]. The amount of noise would increase drastically if keys become invisible. Although many users have learned the key positions via the 10-finger motor skill on physical keyboards, such a skill may not fully transfer to virtual keyboards with 1 or 2 finger-typing.

On the other hand, research has shown cases that users are able to interact with spatial interfaces without visual feedback. Virtual Shelves [26] showed that users were able to point at a set of spatial positions without visual feedback; imaginary user interface research [18] showed users could interact with spatial interfaces that existed only in their imagination. Thanks to the dominance and wide adoption of the Qwerty layout, users' memory on key locations might be strong enough to guide them to land the touch point in the vicinity of the target key position without visual cues.

Advances in statistical decoding also offer increasingly greater error tolerance to typing on virtual keyboards. With currently available statistical decoding technology, users were able to type between 22 - 24 WPM on a watch-sized keyboard [14], suggesting that the modern statistical decoder can handle the amount of imprecision due to the larger finger relative to the small watch-sized full Qwerty. Likewise, the correction ability of the decoder might be strong enough to correct input errors on invisible keyboards.

Contributions. We have investigated whether it was possible to make a keyboard invisible but functional, how users type on such a keyboard, and how to support it. Our investigation led to the following findings.

Our initial Wizard of Oz study showed all participants could correctly specify relative key positions on a virtual Qwerty keyboard even when keys were invisible, even though their typing patterns varied from typing on a visible keyboard.

Second, based on the initial study results, we designed the spatial model of the invisible keyboard decoder to account for the typing patterns. We derived the adapted spatial model for the invisible keyboard from our study data, integrated it into a state-of-the-art decoder with a language model, and systematically evaluated it. User studies showed that this method was very effective: it increased the input speed by 11.5% over using the default spatial model in decoding. We disclose the parameters of the derived spatial model (Appendix), which are available for use by other researchers.

Third, a 3-day multi-session evaluation showed that users could learn typing on an invisible keyboard with the adapted spatial model to a practical level. The input speed was 37.9 WPM, which was close to the speed on a regular visible keyboard (41.6 WPM), after 20 - 25 minute practice on each day in 3 days. This surprisingly promising performance was contrary to the common expectation that removing the key visual would hamper typing performance. The input speed on the invisible keyboard also increased from 31.3 WPM on day 1 to 37.9 WPM on day 3, showing a rapid learning process.

RELATED WORK

Our work builds on prior research on invisible and imaginary user interfaces, keyboards with reduced screen real estate, and statistical keyboard decoding techniques.

Invisible and Imaginary User Interfaces

Fishkin, Moran and Harrison [9] envisioned that user interfaces evolved from keyboard UI to Graphical UI (GUI), Gestural UI, Tangible UI, Embodied UI, and finally to invisible user interfaces, reflecting a progression towards tighter embodiments, more directness in manipulating the intended object, and more coincidence between input and output. Ever since then, researchers have made progress toward realizing this vision, and setting a general context for our work.

Research on invisible, imaginary and translucent user interfaces has shown the feasibility of users interacting without visual feedback, and the benefits of these interfaces. Li et al. [26] showed that users could pick an item out of 11 locations spread out along the phi and theta planes of a hemisphere. Gustafson et al.'s studies [18] showed participants could draw basic characters and sketches on an imaginary sketchpad, and acquire points in an imaginary space using left hand as a reference, despite the lack of visual feedback. Their research [19] also showed by using a physical device (e.g. an iPhone), a user inadvertently learned the interface and could then transfer that knowledge to an imaginary interface. Gupta et al. [17] proposed porous interfaces, a paradigm that combined translucent interfaces with finger identification, to support efficient multitasking on mobile devices.

Keyboards with Reduced Screen Real Estate

A considerable amount of effort has been invested into reducing the space occupied by a keyboard. Back to the era of feature phones, Letterwise [30] and other commercial dictionary-based disambiguation software supported reduced (or cluster) keyboard with 12 keys; Green et al. [15] created a reduced Qwerty keyboard which mapped four rows of a standard keyboard into the home row, with different characters encoded via modifier keys and multi-tap input. Clawson et al. [6] investigated the impacts of limited visual feedback on Twilddler and a mini-Qwerty physical keyboard, which showed that blind typing on Twiddler was faster and more accurate than on a mini-Qwerty keyboard.

As smartphones are becoming popular, research has been conducted on reducing the size of a virtual keyboard, to release more screen space for displaying content and interaction. Li et al. [27] proposed the 1Line keyboard: condensing three rows of keys into one row by arranging 26 letters in eight keys at the bottom of a tablet device. Although the 1Line keyboard was designed for 10-finger typing on a tablet, which had greater screen real estate than many portable touchscreen devices, users still benefited substantially from the extra space released from the keyboard. The Minuum Keyboard [23] is another commercialized one-row keyboard available on both Android and iOS.

Previous research also investigated back-of-the-device interaction for text entry. Wigdor et al. [39]'s LucidTouch enables

touch input on the back of the devices, by overlaying an image of the user's hands onto the screen and creating an illusion of semi-transparency. It allows for accurate targeting, but still requires the presence of an on-screen keyboard for text entry tasks. Others also attempted applying a physical keyboard to the backside of the mobile devices to maximize the use of the display for visual output. Scott et al. [36]'s RearType supported 10-finger typing on iPad-size mobile devices at an average typing speed of 15 WPM. Kim et al. [24] presented the Back Keyboard, a Qwerty keyboard on the back of a mobile phone. Users were able to reach an average text entry rate of 15.3 WPM.

Beyond reducing keyboard sizes, the ASETNIOP Keyboard on tablet [28] provides a chorded keyboard system which support 10-finger typing on an transparent keyboard; BlindType made a video demo showing the prototype of an invisible keyboard; the Fleksy Keyboard [10] supports an invisible keyboard mode. However, there is not any published document explaining how users type, how the keyboards work, or the performance for any of these keyboards.

Disappearing keyboard has been appealing to users. Previous researchers have explored supporting it on TV [29] and smartwatches [32]. Lu et al.'s work [29] showed that on a smart TV users could recall relative key positions on a touch pad with the keyboard displayed on the TV screen. Users could type at 17-23 WPM in such a setting. Mottelson et al. [32] developed a swipe-based transparent keyboard on a smartwatch. Users could enter text at 10.6 WPM after 30 minutes.

Previous research has also explored making the keyboard translucent. KeyGlasses [35] displays additional semi-transparent keys on the keyboard after each character input, in order to minimize the movement of the pointing device. Arif et al. [1] proposed the translucent keyboard with the camera view behind the keys, which enables users to perform text entry on the move while still able to see the surroundings for navigational purposes. Users of different expertise levels were able to reach an average of 24.07 WPM typing speed while walking through a course path. Going further, we have extended the invisible interaction paradigm to direct tapping on smartphones.

Keyboard Decoding

A significant amount of effort has been invested in improving virtual keyboard decoding, in order to handle noisy and error-prone input signals from finger touch.

Similar to the algorithm used in a feature phone keyboard, one approach is to firstly convert a touch point sequence into a key sequence, and search for the most likely match between the key sequence and a word. Both the 1Line keyboard [27] and relative keyboard input system [34] adopted such an approach. Rashid and Smith's work [34] allows a user with the ability to touch-type to type anywhere on the sensing surface without a visual keyboard. Evaluated with real user data, however, the average accuracy of such a keyboard was very low, ranging from 48.5% to 45.6%. The modern statistical decoding techniques employed in our research substantially outperform such method, with the accuracy more than 95%.

More and more recent research including this work has adopted a statistical approach: rather than deterministically mapping touch points into key sequences, it treats a touch point as a noisy signal, which has a probability distribution over multiple keys. The probability inferred from touch points are then combined with the probability from language models via the Bayes' Rule to determine the probability of a word candidate. Goodman et al. [13] were the first to propose combining a language model with pen/touch model to correct input errors on soft keyboards. Bi and Zhai [4, 5] derived FFitts law [4] to model finger touch location with a dual Gaussian distribution model, offering a better approximation of touch model for text entry. Weir et al. [38] proposed a Gaussian Process regression approach to model the touch locations, and allowed users to control the uncertainty of input via touch pressure. Vertanen et al. [37] further proposed a sentence-level decoding approach by combining touch model and language models.

In addition to taking advantage of language models, researchers also explored other methods for decoding. Kristensson and Zhai [25] presented a pattern-matching approach which viewed the hit points as a high resolution geometric pattern, and matched it against patterns formed by the letter key center positions of legitimate words in a lexicon. Gunawardana et al. [16] developed key-target resizing algorithms, where the underlying target areas for keys were dynamically resized based on their probabilities.

EXPERIMENT 1: UNDERSTANDING TYPING ON AN INVISIBLE KEYBOARD

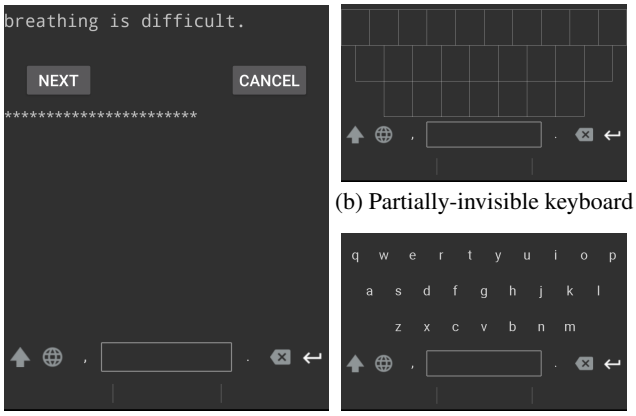
We first carried out a study to investigate the typing behavior on an invisible touchscreen keyboard. We aimed to answer the following two questions: 1) can users type when keys are invisible? 2) if they can, how do they type differently?

Experiment Setup

We designed an invisible Wizard of Oz keyboard to first capture users' natural typing behaviors which were not biased towards any decoding algorithms. In the study, participants were instructed to type as naturally as possible, and assume that the keyboard would correct input errors. The output of the keyboard was shown as asterisks, as shown in Figure 2. In addition to avoiding bias towards any decoding algorithms, this task also reflected expert typing behavior where users typed ahead, ignored the intermediate output and relied on the keyboard to correct errors.

Tasks. The study included three keyboard conditions: *visible*, *partially-invisible* , and *invisible* (Figure 2). We included visible and partially-invisible as baseline conditions. In the partially-invisible condition, key labels were hidden but the boundaries were visible. Including these baseline conditions allowed us to observe how typing behaviors change as the keyboard visuals were reduced. In both conditions, the keyboard layouts were the same size as a regular Google keyboard on Nexus 5X. The key size was 6.5 by 8.1 mm (108 by 135 pixels). The invisible keyboard is shown in Figure 2a. The last row of the keyboard (with functional keys and space

key) was always visible to users; keyboard position and size were unknown to the users.



(a) Invisible keyboard (b) Partially-invisible keyboard (c) Visible keyboard

Figure 2. The application used in Experiment 1. (a) is an invisible keyboard; (b) is a partially-invisible keyboard, where only the boundaries of each key are visible; (c) is a visible keyboard.

Participants and Apparatus. We recruited 18 participants (8 female), aged from 18 to 50. All participants had at least one year experience of typing on touchscreen keyboards on mobile devices. The median of self-reported familiarity with Qwerty keyboard (1: extremely unfamiliar; 10: extremely familiar) was 9. They used their preferred posture: 9 participants (50%) with two thumbs, 6 with one thumb (33.33%), and 3 with one index finger (16.67%). Approximately 70% of the participants used Google keyboards on a daily basis, and the rest of the participants used iOS keyboards on iPhones.

The experiment was conducted on a LG Nexus 5X device running Android 6.0. The phrases were randomly selected from MacKenzie and Soukoreff's test set [31]. The same set of phrases were used for all the participants.

Design. The study was a within-subject design. The independent variable was the three levels of keyboard visibility conditions: visible, partially-invisible, and invisible. The orders of the three conditions were counterbalanced across participants. Participant were instructed to complete three tasks. Each of the tasks was for one keyboard condition. Each task consisted of 4 blocks, and each block had 10 phrases. Participants could take a short break after each block.

In total, the study included:
3 layouts $\times$ 4 blocks $\times$ 10 phrases $\times$ 18 participants
= 2160 phrase.

Results

Figure 3 shows the distributions of touch points for each key in each condition. As shown in Figure 3c, the centers of touch point distributions formed a Qwerty layout, showing that participants were able to correctly memorize the relative key positions even when keys were invisible.

Vertical Offsets

We define the vertical offset as $y_t - y_c$, where y_t is the y coordinate of a touch point and y_c is the y coordinate of the geometric center of the target key. A positive/negative vertical offset indicates the touch point is below/above the key center.

The mean vertical offset across keys (in pixels) were -96.4 ($SD = 38.8$) for the invisible, 22.6 ($SD = 5.0$) for partially-invisible conditions, and 20.8 ($SD = 7.6$) for the visible condition. The invisible condition resulted in the greatest offset, which was more than 4 times as great as that in the visible condition. ANOVA showed the keyboard visibility had a main effect on vertical offset ($F_{2,34} = 74.66, p < 0.0001$). Pairwise comparisons with bonferroni adjustment showed the differences between invisible vs. partially-invisible condition and invisible vs. visible condition were significant (both $p < 0.0001$), while the difference between partially-invisible vs. visible condition was not ($p > 0.05$).

As shown in Table 2 in Appendix, the mean offset for 25 out of 26 keys in the visible condition and for all keys in the partially-invisible condition were positive, indicating that participants tended to land touch points below key centers. This finding concurs with the observation in Azenkot and Zhai's study [2]. In contrast, in the invisible condition touch points had a strong upward shift: the mean vertical offset of every key was negative. In the invisible condition, touch points for the top-row keys (e.g., "q", "w", "e") had greater vertical offsets in magnitude than those for the bottom-row keys (e.g., "x", "c", "v"). The mean offset was -44.52 ($SD = 18.2$) for the bottom row, -93.1 ($SD = 8.77$) for the middle row, and -135.7 ($SD = 8.2$) for the top row (in pixels). The vertical offset in the top row was 3.05 times as great as that in the bottom row.

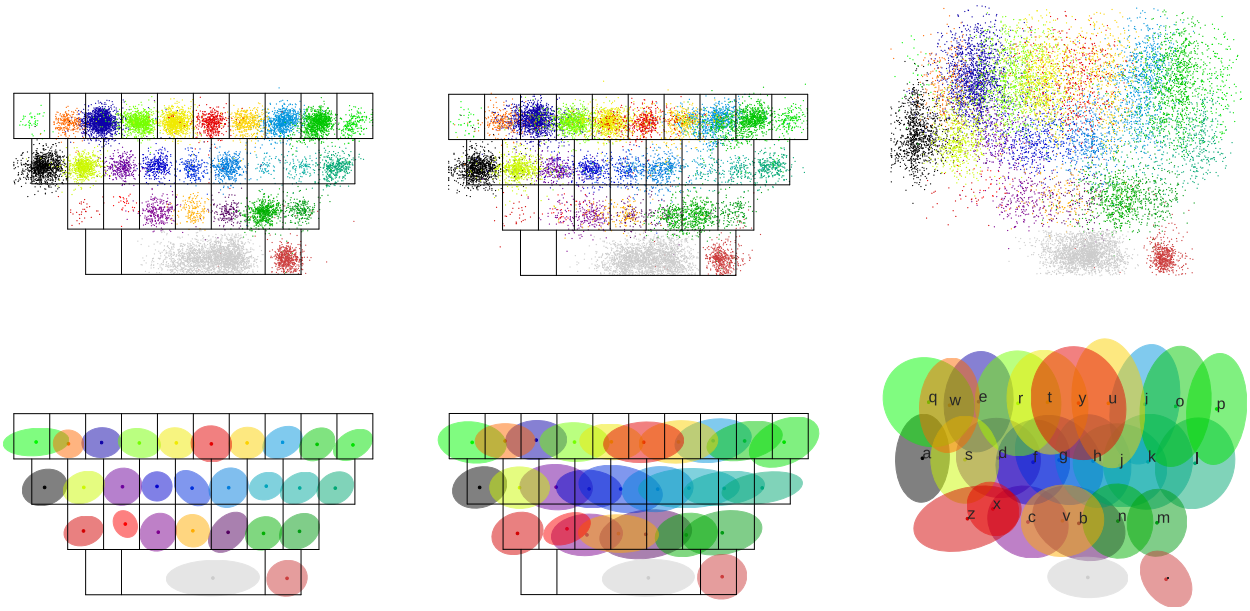
Horizontal Offsets

Similarly, we define the horizontal offset as $x_t - x_c$, where x_t is the x coordinate of a touch point and x_c is the x coordinate of the geometric center of the target key on a regular keyboard. A positive/negative offset indicates hitting to the right/left of the key center.

The mean horizontal offset was the largest in magnitude in the invisible condition, at -47.3 pixels ($SD = 35.1$), which was also 4 times as great as that in the visible condition ($M = -10.5, SD = 5.7$). The mean for keys in the partially-invisible condition was -25.3 pixels ($SD = 19.2$), smaller than in the invisible condition, but greater than that in the visible condition. There was a main effect of keyboard type on horizontal offset ($F_{2,34} = 36.06, p < 0.0001$). Pairwise comparisons with bonferroni adjustment showed the differences between any two conditions were significant ($p < 0.0001$).

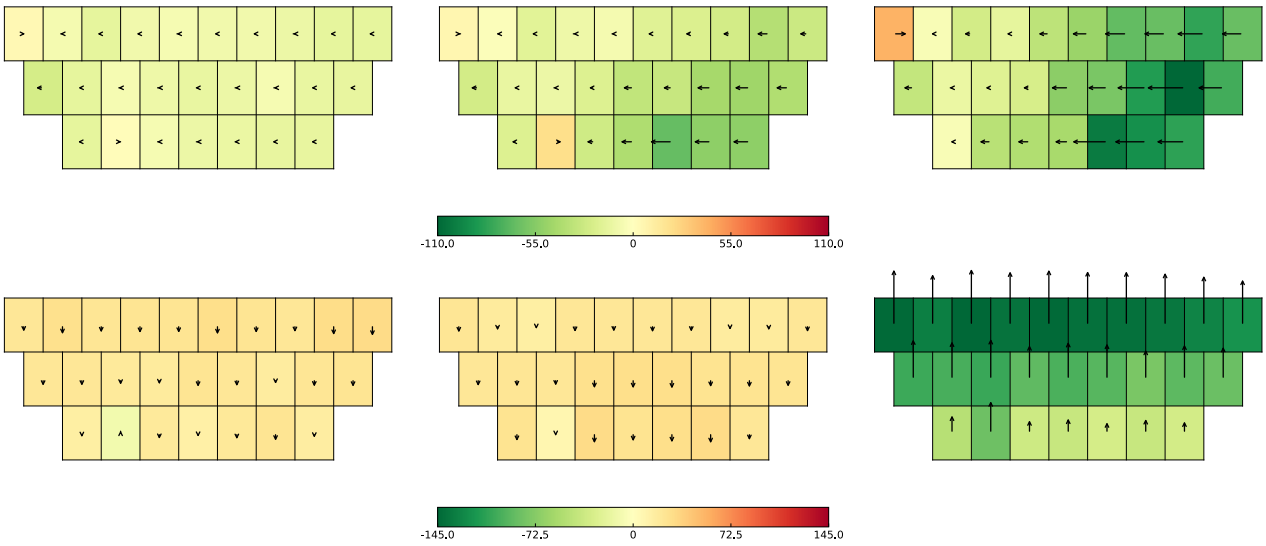
Variances

As expected, we observed greater variances in the invisible and partially-invisible conditions. The mean standard deviations (in pixels) across keys were 48.0 (x direction) and 38.7 (y direction) for the invisible, 33.6 and 13.8 for the partially-invisible, and 15.5 and 10.2 for the visible condition, as shown in Figure 5. The standard deviations in the x, and y directions in the invisible condition were 3.10 times, and 3.81 times as great as those in the visible condition, respectively. ANOVA showed keyboard type had a main effect on mean standard deviations ($F_{2,34} = 24.87, p < 0.0001$ for x



(a) Visible keyboard (b) Partially-Invisible keyboard (c) Invisible keyboard

Figure 3. Scatter plots (top) and 95% confidence ellipses (bottom) of touch points in three keyboard layouts. The figures are drawn in the same scale.



(a) Visible keyboard (b) Partially-Invisible keyboard (c) Invisible keyboard

Figure 4. Horizontal (top) and vertical (bottom) offsets for touch points in three keyboard layouts. The color gradient and arrow of each key indicate the value of the offset in pixels. The arrows are drawn in the scale of key size. The key boundaries in invisible keyboard are for illustration purposes only.

direction, $F_{2,34} = 41.54, p < 0.0001$ for y direction). Pairwise comparisons with bonferroni adjustment also showed that the differences between any two conditions were significant ($p < 0.0001$). The results showed that in the invisible condition the standard deviation in the y direction increased drastically from the bottom row to the top row. The mean standard deviation in y were 50.6 ($SD = 5.8$) for the bottom row, 65.2 ($SD = 3.7$) for the middle row, and 80.7 ($SD = 9.3$) for the top row in pixels. The standard deviation for the top row was 60% greater than that for the bottom row.

Discussion

Can Users Type When Keys Are Invisible?

The answer to this question is affirmative. The results showed users were able to correctly recall key locations relative to the

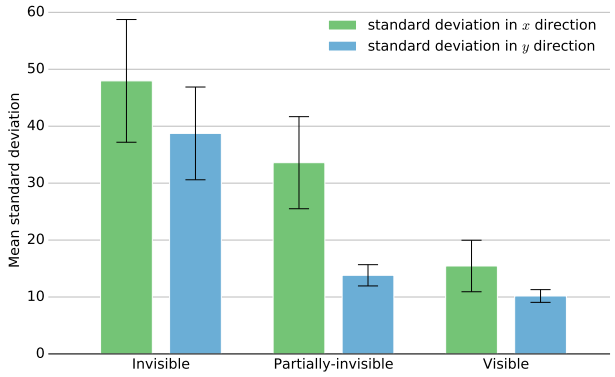


Figure 5. The means (95% confidence interval) of standard deviations across keys for x and y directions in three conditions.

Qwerty layout in both invisible and partially-invisible conditions: the centers of touch points still formed a Qwerty layout (Figure 3). It shows that users' memory on the key locations is strong even with 1- or 2-finger typing. Although previous research [8] showed users could specify relative key positions with 10-finger typing, it was unknown whether users were able to do so on a phone-sized keyboard, because the motor memory on key positions learned via 10-finger touch typing might not translate to 1 or 2-finger typing. Our study showed that users' memory recall could play a stronger than expected role even on mobile typing with 1 or 2 fingers. All participants also completed the typing task even in the invisible condition, not withstanding that fact that this was a Wizard of Oz where the users was not given the knowledge of results.

Typing Patterns on an Invisible Keyboard

The study results also revealed distinctly different typing patterns when keys were invisible. First, we observed greater offsets and variances of touch points in the invisible condition. The magnitudes of both vertical and horizontal offsets in the invisible condition were 4 times as great as those in the visible condition. The standard deviations in x and y directions were 3.10 and 3.81 times as those in the visible condition, respectively.

Second, users tended to land touch points higher than the original key position in the invisible condition. The mean vertical offset for every key was negative, showing a strong upward shift trend. This pattern also contrasted with the typing behavior in the visible condition where users tended to hit below the key centers, which was observed both in our study and Azenkot and Zhai's study [2]. The results also showed much greater variances of touch points in y direction than in the x direction in the invisible condition. The standard deviation of touch points in y direction was 48.0, 24% greater than the standard deviation in x direction, which was at 38.7. These results showed that without knowing the upper bound of the keyboard, participants were uncertain about the y coordinate of a key, resulting in large offset and variance.

Third, touch points for the top-row keys (e.g., "q", "w", "e") had greater vertical variances and offsets in magnitude than those for the bottom-row keys ("x", "c", "v"). The mean standard deviation in y for top-row keys was 60% greater than that for the bottom-row keys, and the mean vertical offset was 3.05 times as great as that of the bottom-row keys. The bottom row was adjacent to the space bar row, which might serve as a reference for locating the bottom-row keys, resulting in less variances and offsets.

Typing Patterns on a Partially-Invisible Keyboard

Our study also showed how users typed in the partially-invisible condition. The offsets and variances of touch points in the partially-invisible condition were greater than those in the visible condition, but smaller than those in the invisible condition. These results matched the intuition that the level of visual guidance in the partially-invisible condition was an intermediate between the visible and the invisible conditions. Interestingly, as shown in Figures 3 and 4, the touch point distributions in the partially-invisible condition were very different from those in the invisible condition, but similar to those in the visible condition.

ADAPTING SPATIAL MODEL FOR INVISIBLE KEYBOARDS

Experiment 1 showed it was possible to type on an invisible keyboard. On the other hand, the distinct typing patterns indicated that the typical keyboard decoder, which is designed to handle input signals on a regular visible keyboard, might not work well for keyboards with reduced visibility. To account for the distinct typing patterns, we have derived a spatial model based on the study results for the invisible keyboard, integrated them into a state-of-the-art decoder with language models, and systematically evaluated it. In this section, we first explain the basic principle and spatial model for statistical decoding. Readers familiar with these two concepts can skip it. We then describe how we derived the spatial models for the invisible keyboard and integrated it into a state-of-the-art decoder.

Statistical Decoding Principle

Fundamentally, the current keyboard decoding [5, 11, 13, 37, 38] usually relies on two types of knowledge represented by two models: a *spatial model* (SM) that relates a touch point to keys on a keyboard, and a *language model* (LM) that gives the prior probability distributions of words given history. As soon as the user reaches the word boundary such as hitting a space or punctuation, the decoder combines the probabilities from these two models to generate the final probability of a word according to the Bayes' theorem, and suggests the words with highest final probabilities to the user.

Formally, given a set of touch points on the keyboard $S = \{s_1, s_2, s_3, ..., s_n\}$, the decoder is to find word W^* in lexicon L that satisfies:

$$W^* = \arg\max_{W \in L} P(W|S). \quad (1)$$

From the Bayes' rule,

$$P(W|S) = \frac{P(S|W)P(W)}{P(S)}. \quad (2)$$

As $P(S)$ is an invariant across words, Equation (1) becomes:

$$W^* = \arg\max_{W \in L} P(S|W)P(W). \quad (3)$$

$P(W)$ is obtained from a language model (LM) and $P(S|W)$ is from a spatial model (SM).

Spatial Model

This section describes how a decoder calculates $P(S|W)$ from a spatial model.

Assuming that W is comprised of n letters: $c_1, c_2, c_3, ..., c_n$, S has n touch points, and each tap is independent, we have:

$$P(S|W) = \prod_{i=1}^{n} P(s_i|c_i). \quad (4)$$

As shown in previous research, touch points for tapping a key follow a bivariate Gaussian distribution [2]. Therefore, assuming the coordinates of s_i is (x_i, y_i), $P(s_i|c_i)$ can be calculated as:

$$P(s_i|c_i) = \frac{1}{2\pi\sigma_{i_x}\sigma_{i_y}\sqrt{1-\rho_i^2}} \exp\left[-\frac{z}{2(1-\rho_i^2)}\right], \quad (5)$$

where

$$z \equiv \frac{(x_i-\mu_{i_x})^2}{\sigma_{i_x}^2} - \frac{2\rho_i(x_i-\mu_{i_x})(y_i-\mu_{i_y})}{\sigma_{i_x}\sigma_{i_y}} + \frac{(y_i-\mu_{i_y})^2}{\sigma_{i_y}^2}, \quad (6)$$

(μ_{i_x}, μ_{i_y}) is the center of the touch point distribution aimed on key c_i; σ_{i_x} and σ_{i_y} are standard deviations; ρ_i is the correlation.

The decoder usually uses the center of key c_i, or the center with a small, constant offset as (μ_{i_x}, μ_{i_y}), and constant σ_{i_x} and σ_{i_y} across keys.

Deriving Spatial Model for Invisible Keyboards

To adapt the decoder to the invisible keyboard, we calculated the spatial model parameters according to the study results. More specifically, for each key c_i, we changed μ_{i_x}, μ_{i_y}, σ_{i_x}, σ_{i_y} and ρ_i in Equations (5) and (6) to the values estimated from the study data (Figure 3 and Table 2 in Appendix). We then swap the existing spatial model of the Google Keyboard decoder with the adapted spatial model we calculated.

Note that a keyboard decoder also adds penalties to different types of spelling or cognitive errors such as insertion, omission, and transposition errors. Since our study did not show whether the error type changed when typing on an invisible keyboard, we kept this procedure unchanged. We used a language model which had a lexicon of 70K words.

We kept auxiliary keys such as space bar, period, comma, shift, and backspace keys visible, because input from these keys could not be corrected by the decoder, and some served special functions (e.g., backspace for deletion and shift for switching keyboard layouts).

To fully release the screen real estate previously occupied by the keyboard to the background content, we moved the suggestion bar to the bottom of the keyboard. It also allowed users to easily select suggestions at the end of entering a word, since it was close to the word delimiter keys (e.g., space, punctuation). We decided including the suggestion bar for the invisible keyboard design because of the following two reasons. First, it reflected the common practice in existing touchscreen keyboards. The suggestion bar is by default turned on in major keyboard products (e.g., SwiftKey, Swype, Google Keyboard). Even the iPhone keyboard (iOS 11) recently switched to the default-on mode for suggestion bar. Although its general utility is low [33], the suggestion bar serves as a "safety net" in case the top auto-correction fails, and is necessary for entering Out-Of-Vocabulary words. Second, the suggestion bar could be very useful, if not necessary, when the input signal is ambiguous or noisy especially on an invisible keyboard. We have implemented the invisible keyboard (shown in Figure 1b) as a general touchscreen input method service, which is compatible with all applications running on the Android devices.

EXPERIMENT 2: EVALUATING SPATIAL ADAPTION METHOD

We conducted a study to investigate the effectiveness of the spatial adaption method. We compared the invisible keyboard with the adapted spatial model (referred as *adapted*), with a baseline invisible keyboard (referred as *unadapted*) whose decoder remained the same as a regular visible keyboard. In other words, the baseline was a condition in which we simply hid keys and made no changes to the decoder.

Experiment Setup

Design. The study was a typical phrase transcription task. It was a within-subject design. The independent variable was the invisible keyboard with two different decoders: spatially-adapted and unadapted.

Before the formal study, participants performed a practice session with five phrases. Each participant was instructed to complete a phrase input session for each keyboard. A session included 4 blocks, each with 10 phrases. To reflect the regular text entry behavior, participants were allowed to freely use backspace and suggestion bars. At the beginning of each trial, a phrase was displayed to help participants quickly memorize the phrase. The participant was instructed to enter it as quickly and accurately as possible. After entering a phrase, the user pressed the "Done" button to proceed to next phrase. The order of the two keyboards were counter-balanced across participants.

Participants and Apparatus. 12 subjects (2 female), aged from 18 to 40 participated in this experiment. All participants had at least one year experience of typing on touchscreen keyboard on mobile devices. The median of self-reported familiarity with the Qwerty keyboard (1: extremely unfamiliar; 10: extremely familiar) was 9. They used their preferred posture (two thumbs, one thumb, or index finger) throughout the study. The experiment was conducted on an LG Nexus 5X device running Android 6.0. Figure 6 shows the application used in the study. Each participant typed the same set of 40 phrases randomly chosen from MacKenzie and Soukoreff's test set [31].

In total, the study included:
12 participants $\times$ 2 keyboards $\times$ 40 phrases = 960 phrases.

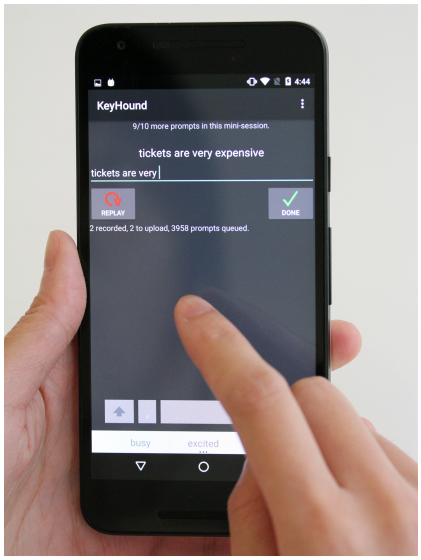


Figure 6. Experiment 2 setup. A user held the device in one hand, and typed on the invisible keyboard with her preferred posture. The spatially-adapted and the unadapted keyboards shared the same appearance.

Results

As widely adopted in text entry research, we measured the performance in *speed* and *error rate*.

Speed. The input speed was calculated as:

$$WPM = \frac{|S-1|}{T} \times \frac{1}{5}, \tag{7}$$

where S is the length of the transcribed string in character, T is the time elapsed in minutes from the first touch event to the last touch event for the string. Every five characters were counted as one word.

The means of speed in WPM were 29.30 ($SD = 6.20$) for the spatially-adapted keyboard, and 26.29 ($SD = 7.08$) for the unadapted, as shown in Figure 7. There was a main effect of keyboard type ($F_{1,11} = 7.959, p = 0.0166$) on input speed.

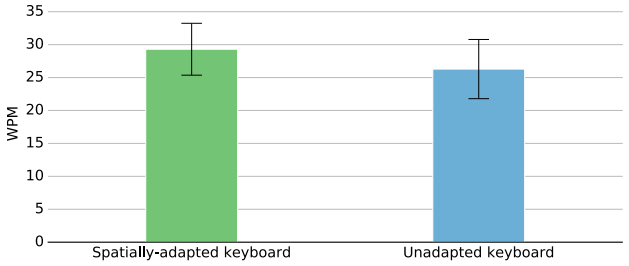


Figure 7. Means (95% confidence interval) of input speed for spatially-adapted and unadapted invisible keyboards.

Error Rate. Since all the keyboards in this research performed word-level corrections, we used the word error rate. The word error rate [3] is based on Minimum Word Distance (MWD), which is the smallest number of word deletions, insertions, or replacements needed to transform the transcribed string into the expected string. The word error rate is defined as:

$$r = \frac{MWD(S,P)}{LengthInWords(P)} \times 100\%, \tag{8}$$

where $MWD(S,P)$ is the minimum word distance between the transcribed phrase S and the target phrase P, and $LengthInWords(P)$ is the number of words in P. Since participants were encouraged to correct errors while entering phrases, this value reflects the word error rate after correction.

The means of word error rate were 3.83% ($SD = 5.5\%$) for the spatially-adapted, and 2.48% ($SD = 2.24\%$) for the unadapted. ANOVA did not show a significant main effect of keyboard type ($F_{1,11} = 1.057, p = 0.326$).

Subjective Rating. At the end of the study, participants were asked to rate the keyboard using a 5-level scale (1: very dislike, 5: very like). The median ratings were 3 (OK) for the spatially-adapted, slightly higher than that of the unadapted (2: Dislike); the modes were 4 (like) for both keyboards. Many users commented that they preferred typing on the spatially-adapted keyboard because the keys were positioned further apart and made it easier for typing.

Discussion

The derived spatial model outperformed the existing spatial model. The input speed of the keyboard with the derived spatial model was 11.5% faster than the unadapted in the invisible condition. The difference was statistically significant ($p < 0.05$).

This promising result was observed after the adapted spatial model was integrated into a state-of-the-art decoder with language models. It suggests that the spatial typing patterns are strong enough to generate significant benefits even when language models are present. This is not obvious because the spatial adaption could have just negligible effects due to the presence of language models: the correction ability from language models might be very strong so that the improvement from the spatial adaption has only minor effects. Our study shows this is not the case. Our finding also has strong external validity: the participants and phrases in this study were totally different from those in the previous study based on which the spatial model was derived.

Our finding expands the understanding on the effectiveness of spatial adaption. Literature has shown that spatial adaption reduced the error rates for 10 finger typing on a tablet [7], for posture, key location and individual adaption [12, 40], and for TV with a keyboard displayed [29]. Our research shows spatial adaption is effective for typing on an invisible keyboard for 1- and 2-finger typing.

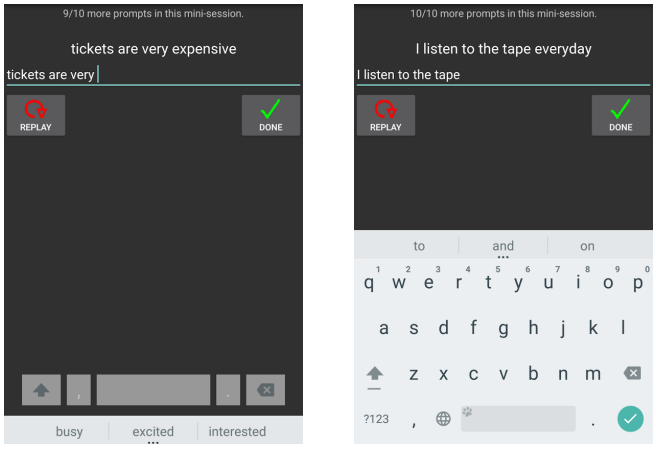
EXPERIMENT 3: EVALUATING INVISIBLE KEYBOARD

To understand the overall performance of the invisible keyboard and its learnability, we conducted a 3-day multi-session user study. Because the previous study showed the spatial adaption was effective, the decoders of the invisible keyboard was adapted according to the data collected in the data collection experiment (Table 2 in Appendix).

Experiment Setup

Design. We adopted a between-subject design to eliminate the potential carry-over effect between keyboard conditions. The independent variable was the keyboard visibility with 2 levels: invisible and visible. We included the visible condition as the baseline. In both conditions, participants performed phrase transcription tasks in three consecutive days. This multi-day study design allowed us to evaluate the learnability of the keyboard. The test phrase set included 40 phrases randomly chosen from MacKenzie and Soukoreff's test set [31]. The same test set was used over 3 days and keyboards, to ensure the validity of the measurement.

The phrase transcription task was the same with that in the previous study evaluating the spatial adaption method. Figure 8 shows the screenshots of the keyboards in the study. Each task on each day took approximately 20 - 25 minutes.



(a) Invisible keyboard (b) Visible keyboard

Figure 8. Screenshots of the two keyboards used in Experiment 3. (a): invisible keyboard. (b): visible keyboard.

Participants and Apparatus. 20 subjects (6 female, 30%) aged from 18 to 34 participated in the study: 10 for the invisible and 10 for the visible condition. All participants had at least one year experience of typing on touchscreen keyboard on mobile devices. The median of self-reported familiarity with the Qwerty keyboard (1: extremely unfamiliar; 10: extremely familiar) was 9. The apparatus was the same with that in the previous study.

Results

Speed. Figure 9 shows the input speed by keyboard condition and day. As shown, the input speeds of the invisible keyboards increased from day 1 to day 3. ANOVA showed the day had a significant main effect on the input speed ($F_{2,18} = 8.862, p = 0.00209$). Pairwise comparison with bonferroni adjustment showed the difference was significant for day 1 vs. day 2, day 1 vs. day 3 ($p = 0.039, p = 0.030$ respectively), but not for day 2 vs. day 3 ($p = 0.317$). The input speeds of the visible keyboard exhibited a similar trend over the three days, however, ANOVA did not show a significant main effect of the day on the input speed ($F_{2,18} = 3.092, p = 0.0701$). Pairwise comparison with bonferroni adjustment did not show significant differences between any two days ($p > 0.05$). The results confirmed that training had little effect on the performance of the visible keyboard.

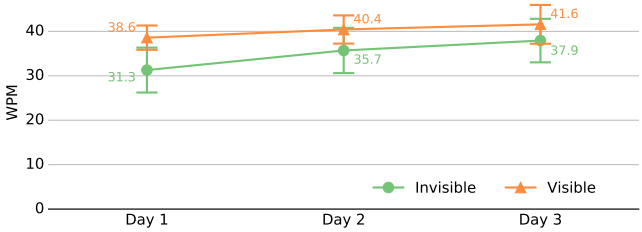


Figure 9. Means (95% confidence interval) of input speed by keyboard condition and day.

We also compared the input speeds of the invisible keyboard on day 3, which approximated the expert-level performance, with the baseline condition (i.e., visible keyboard). The means of input speeds in WPM were 37.9 ($SD = 6.85$) for the invisible keyboard on day 3, and 41.6 ($SD = 6.10$) for the visible keyboard. ANOVA did not show a main effect of keyboard type on input speed ($F_{1,18} = 1.587, p = 0.224$). This showed that after a 3-session training, the input speed of invisible keyboard (37.9) was very close to that of the visible keyboard (41.6).

Word Error Rate. The mean error rates for each keyboard condition on each day are shown in Table 1.

	Day 1	Day 2	Day 3
Invisible	0.85 (1.08)	1.45 (1.31)	2.43 (2.26)
Visible	2.74 (1.31)	1.86 (1.71)	2.40 (2.35)

Table 1. The means (SD) of error rates (%) for the two keyboard conditions on each day.

The mean error rates were all under 3% in both conditions on each day. For the invisible keyboard, ANOVA showed the day had a significant main effect on the error rate ($F_{2,18} = 3.999, p = 0.0366$). For the visible keyboard, ANOVA did not exhibit a significant main effect of the day on the error rate ($F_{2,18} = 0.924, p = 0.415$). As for the expert-level performance on the last day of the study, there was no significant effect of keyboard visibility on the word error rate ($F_{1,18} = 0, p = 0.984$).

Subjective Ratings. To understand users' preferences of different design options, each participant rated the keyboard using a 5-level scale: 1 (Very dislike) - 5 (Very like) at the end of the study session on each day. Figure 10 shows the median of the ratings over 3 days on each keyboard. As shown, users' subjective ratings increased from day 1 to day 3 for the invisible keyboard.

Discussion

Our study shows that the invisible keyboard is a practical and easy-to-learn design option for a virtual keyboard. The novice input speed was above 30 WPM. After 20 - 25 minutes practice for three consecutive days, users were able to enter text at 37.9 WPM, only 4 WPM slower than the speed of the regular visible keyboard. ANOVA did not show a significant difference for input speed on the last day ($p = 0.0701$). These

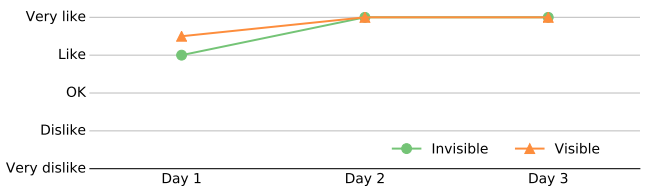


Figure 10. Median ratings by keyboard condition and day.

promising results were remarkable. Although previous research showed that users could type to a certain degree relying on finger motor memory [8, 29], our results showed users could type with no visuals on keyboard reasonably well after a short period of training. Note that our finding was obtained based a sample of 20 participants who all had experience with virtual keyboards. The sample size was relatively small and future research is needed to investigate the performance of users with little experience of typing on virtual keyboards.

The promising performance of the invisible keyboard can be largely attributed to the correction ability of the decoder. 70% (7 of 10) of the participants commented that the keyboard was able to correct most of their errors and they relied on it to enter text. To quantify the error correction ability, we examined *error reduction rate*, which is the number of correct auto-correction or suggestions divided by the total number of incorrect literal strings. This metric was 94.2% for the invisible keyboard.

Some users raised a question about interacting with the widgets underneath the keyboard during text entry. As shown in the previous research [17], multitasking is challenging on mobile devices. One solution is to use finger identification technique to assign input from different fingers to different applications [17].

Overall, our investigation shows that the invisible keyboard is viable, practical, and easy-to-learn. It saves precious screen real estate, and introduces little performance degradation after 20 - 25 minute practice on each day in 3 days.

CONCLUSIONS

Invisible keyboard is a desirable design option. Our research on invisible keyboard answered a set of questions through 3 experiments: Are users able type on an invisible keyboard? If so, how will users type differently? What technology can be applied to support it? What is its performance?

First, Experiment 1 showed it was possible to type on an invisible keyboard. Users had strong memory recall on key positions: they could correctly recall key positions relative to the Qwerty layout even when keys were completely invisible and with 1 or 2 finger typing. On the other hand, their touch point distributions exhibited different patterns compared with those on a visible keyboard.

Second, we examined whether adapting the spatial model of the decoder would improve the performance of the invisible keyboard. We derived the adapted spatial model for the invisible keyboard from our study data, integrated it into a state-of-the-art decoder with a language model, and systematically evaluated it. Experiment 2 showed this method was very effective. It increased the typing speed of the invisible keyboard by 11.5% over the baseline invisible keyboard which used the default spatial model (i.e., unadapted) and simply hid the keys. We disclose the parameters of the derived spatial model (in Appendix), which are available for use by other researchers.

Third, Experiment 3 showed typing on the invisible keyboard was easy-to-learn, practical and promising. It was remarkable that the input speed of the invisible keyboard was approaching the visible keyboard after 60 - 75 minutes practice in 3 days (20 - 25 minutes on each day). No significant difference was observed between the speed of the invisible keyboard on the last day (37.9 WPM) and the speed of a regular visible keyboard (41.6 WPM). Typing on an invisible keyboard was also easy to learn. The input speed of the invisible keyboard increased from 31.3 WPM on day 1 to 37.9 WPM on day 3, showing a rapid learning process.

Overall, our investigation shows that an invisible keyboard with adapted spatial model is a practical and promising interface option for the mobile text entry systems.

REFERENCES

1. 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 (OzCHI '11)*. ACM, New York, NY, USA, 21–30. DOI:http://dx.doi.org/10.1145/2071536.2071539
2. Shiri Azenkot and Shumin Zhai. 2012. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 251–260. DOI: http://dx.doi.org/10.1145/2371574.2371612
3. Xiaojun Bi, Shiri Azenkot, Kurt Partridge, and Shumin Zhai. 2013a. Octopus: Evaluating Touchscreen Keyboard Correction and Recognition Algorithms via Remulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 543–552. DOI: http://dx.doi.org/10.1145/2470654.2470732
4. Xiaojun Bi, Yang Li, and Shumin Zhai. 2013b. FFitts Law: Modeling Finger Touch with Fitts' Law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1363–1372. DOI: http://dx.doi.org/10.1145/2470654.2466180
5. Xiaojun Bi and Shumin Zhai. 2013. Bayesian Touch: A Statistical Criterion of Target Selection with Finger Touch. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 51–60. DOI: http://dx.doi.org/10.1145/2501988.2502058
6. James Clawson, Kent Lyons, Thad Starner, and Edward Clarkson. 2005. The Impacts of Limited Visual

Feedback on Mobile Text Entry for the Twiddler and Mini-QWERTY Keyboards. In *Proceedings of the Ninth IEEE International Symposium on Wearable Computers (ISWC '05)*. IEEE Computer Society, Washington, DC, USA, 170–177. DOI: http://dx.doi.org/10.1109/ISWC.2005.49

7. 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 (CHI '12)*. ACM, New York, NY, USA, 815–824. DOI: http://dx.doi.org/10.1145/2207676.2208520
8. Leah Findlater, Jacob O. Wobbrock, and Daniel Wigdor. 2011. Typing on Flat Glass: Examining Ten-finger Expert Typing Patterns on Touch Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2453–2462. DOI: http://dx.doi.org/10.1145/1978942.1979301
9. Kenneth P. Fishkin, Thomas P. Moran, and Beverly L. Harrison. 1999. Embodied User Interfaces: Towards Invisible User Interfaces. In *Proceedings of the IFIP TC2/TC13 WG2.7/WG13.4 Seventh Working Conference on Engineering for Human-Computer Interaction*. Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 1–18. http://dl.acm.org/citation.cfm?id=645349.650706
10. Fleksy. 2016. Fleksy Keyboard - GIFs, Custom Extensions, and Themes. http://fleksy.com/. (2016). [Online; accessed 10-February-2017].
11. Andrew Fowler, Kurt Partridge, Ciprian Chelba, Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2015. Effects of Language Modeling and Its Personalization on Touchscreen Typing Performance. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 649–658. DOI: http://dx.doi.org/10.1145/2702123.2702503
12. Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. ContextType: Using Hand Posture Information to Improve Mobile Touch Screen Text Entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2795–2798. DOI:http://dx.doi.org/10.1145/2470654.2481386
13. Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 194–195. DOI: http://dx.doi.org/10.1145/502716.502753
14. Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3817–3821. DOI: http://dx.doi.org/10.1145/2858036.2858242
15. 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 (CHI EA '04)*. ACM, New York, NY, USA, 1429–1432. DOI: http://dx.doi.org/10.1145/985921.986082
16. Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability Guided Key-target Resizing for Soft Keyboards. In *Proceedings of the 15th International Conference on Intelligent User Interfaces (IUI '10)*. ACM, New York, NY, USA, 111–118. DOI: http://dx.doi.org/10.1145/1719970.1719986
17. Aakar Gupta, Muhammed Anwar, and Ravin Balakrishnan. 2016. Porous Interfaces for Small Screen Multitasking Using Finger Identification. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 145–156. DOI: http://dx.doi.org/10.1145/2984511.2984557
18. Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. 2010. Imaginary Interfaces: Spatial Interaction with Empty Hands and Without Visual Feedback. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 3–12. DOI: http://dx.doi.org/10.1145/1866029.1866033
19. Sean Gustafson, Christian Holz, and Patrick Baudisch. 2011. Imaginary Phone: Learning Imaginary Interfaces by Transferring Spatial Memory from a Familiar Device. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 283–292. DOI: http://dx.doi.org/10.1145/2047196.2047233
20. Sean G. Gustafson, Bernhard Rabe, and Patrick M. Baudisch. 2013. Understanding Palm-based Imaginary Interfaces: The Role of Visual and Tactile Cues when Browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 889–898. DOI: http://dx.doi.org/10.1145/2470654.2466114
21. Niels Henze, Enrico Rukzio, and Susanne Boll. 2012. Observational and Experimental Investigation of Typing Behaviour Using Virtual Keyboards for Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 2659–2668. DOI: http://dx.doi.org/10.1145/2207676.2208658
22. Yamada Hisao. 1980. A Historical Study of Typewriters and Typing Methods: from the Position of Planning Japanese Parallels. *Journal of Information Processing* 2, 4 (feb 1980), 175–202.

23. Whirlscape Inc. 2015. Minuum Keyboard By Whirlscape. http://minuum.com/. (2015). [Online; accessed 10-February-2017].

24. 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 (CHI EA '12)*. ACM, New York, NY, USA, 1583–1588. DOI:http://dx.doi.org/10.1145/2212776.2223676

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

26. Frank Chun Yat Li, David Dearman, and Khai N. Truong. 2009. Virtual Shelves: Interactions with Orientation Aware Devices. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 125–128. DOI: http://dx.doi.org/10.1145/1622176.1622200

27. 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 (UIST '11)*. ACM, New York, NY, USA, 461–470. DOI: http://dx.doi.org/10.1145/2047196.2047257

28. Pointesa LLC. 2012. ASETNIOP. http://asetniop.com/. (2012). [Online; accessed 10-February-2017].

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

30. I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, and Eugene Skepner. 2001. LetterWise: Prefix-based Disambiguation for Mobile Text Input. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology (UIST '01)*. ACM, New York, NY, USA, 111–120. DOI: http://dx.doi.org/10.1145/502348.502365

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

32. Aske Mottelson, Christoffer Larsen, Mikkel Lyderik, Paul Strohmeier, and Jarrod Knibbe. 2016. Invisiboard: Maximizing Display and Input Space with a Full Screen Text Entry Method for Smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '16)*. ACM, New York, NY, USA, 53–59. DOI:http://dx.doi.org/10.1145/2935334.2935360

33. Philip Quinn and Shumin Zhai. 2016. A Cost-Benefit Study of Text Entry Suggestion Interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 83–88. DOI: http://dx.doi.org/10.1145/2858036.2858305

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

35. Mathieu Raynal. 2014. KeyGlasses: Semi-transparent Keys on Soft Keyboard. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS '14)*. ACM, New York, NY, USA, 347–349. DOI: http://dx.doi.org/10.1145/2661334.2661427

36. 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 (MobileHCI '10)*. ACM, New York, NY, USA, 171–180. DOI:http://dx.doi.org/10.1145/1851600.1851630

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

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

39. Daniel Wigdor, Clifton Forlines, Patrick Baudisch, John Barnwell, and Chia Shen. 2007. Lucid Touch: A See-through Mobile Device. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 269–278. DOI: http://dx.doi.org/10.1145/1294211.1294259

40. 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 (CHI '13)*. ACM, New York, NY, USA, 2775–2784. DOI: http://dx.doi.org/10.1145/2470654.2481384

APPENDIX

SPATIAL MODEL

Key	Visible condition					Partially-Invisible condition					Invisible condition				
	μ_x	σ_x	μ_y	σ_y	ρ	μ_x	σ_x	μ_y	σ_y	ρ	μ_x	σ_x	μ_y	σ_y	ρ
a	-24.37	33.49	21.33	27.21	0.16	-25.45	40.73	22.04	30.91	0.17	-31.93	40.41	-103.32	65.54	0.05
b	-12.40	29.43	19.89	29.86	0.40	-65.84	68.78	27.79	36.46	0.17	-101.43	69.12	-31.05	55.39	-0.24
c	-6.43	27.40	19.46	28.17	0.11	-26.69	53.30	29.91	30.94	0.08	-38.52	60.31	-36.09	53.46	-0.11
d	-6.15	29.06	18.78	27.71	0.05	-11.35	54.35	20.87	33.77	-0.04	-19.49	63.11	-104.38	58.96	-0.05
e	-15.23	29.44	22.87	22.20	0.04	-17.22	45.81	14.58	29.28	0.03	-25.00	51.90	-144.36	74.77	0.07
f	-10.28	22.60	17.65	21.77	0.01	-20.10	49.21	27.42	28.04	-0.10	-23.13	55.73	-88.86	62.77	0.07
g	-12.77	27.07	23.93	26.24	-0.36	-33.33	62.53	27.68	35.32	-0.24	-52.89	65.56	-97.07	66.30	-0.24
h	-10.62	29.56	22.34	29.41	0.11	-29.83	53.10	28.19	33.56	0.09	-58.59	56.80	-92.90	69.04	-0.13
i	-10.71	29.56	22.08	23.84	0.30	-25.08	57.27	16.70	32.63	0.18	-64.91	52.58	-134.94	88.97	0.19
j	-5.58	25.17	16.72	20.44	0.15	-43.82	75.05	24.57	29.02	0.06	-85.67	69.80	-77.53	61.28	-0.07
k	-12.95	29.13	23.40	23.55	0.30	-45.87	60.24	25.41	25.38	0.22	-109.52	66.60	-90.45	70.39	0.03
l	-14.01	27.49	23.95	24.33	0.22	-39.35	60.34	23.73	24.03	0.30	-75.96	59.82	-84.89	67.52	0.08
m	-13.62	29.96	16.36	26.62	0.24	-51.59	59.45	22.32	33.03	0.22	-82.43	44.72	-33.14	49.81	0.03
n	-13.86	27.32	23.84	24.93	0.13	-51.71	46.64	30.08	32.36	0.13	-92.07	52.66	-39.21	55.30	-0.03
o	-14.69	26.65	29.43	24.52	0.26	-38.33	56.56	17.60	29.54	0.30	-80.50	53.52	-127.38	89.54	0.13
p	-14.88	29.16	31.42	23.23	0.38	-27.57	52.11	21.99	37.36	0.37	-64.80	45.15	-117.89	82.50	0.11
q	4.07	48.88	20.80	20.41	0.14	6.02	50.48	22.94	30.46	-0.12	40.53	67.49	-142.73	65.53	-0.10
r	-9.43	31.60	24.31	21.60	-0.12	-10.08	51.96	21.73	28.87	-0.04	-14.54	64.43	-138.39	78.09	-0.01
s	-14.50	30.90	21.25	24.03	0.22	-13.33	44.91	22.87	30.91	-0.01	-12.12	51.24	-98.14	64.83	0.00
t	-6.14	26.93	24.06	22.60	-0.09	-7.09	47.48	20.80	26.67	-0.06	-33.73	60.84	-142.53	77.36	-0.08
u	-9.24	26.96	24.18	23.36	0.03	-21.26	59.01	21.06	31.61	0.08	-66.68	54.84	-138.59	95.92	-0.10
v	-10.50	24.90	14.28	24.16	-0.03	-39.82	59.93	25.15	27.88	-0.05	-42.60	61.81	-40.38	53.03	0.03
w	-6.69	22.70	27.33	20.57	-0.03	-2.78	44.87	17.70	25.75	0.00	-4.36	45.11	-131.41	70.28	0.07
x	2.38	17.96	-10.26	19.59	-0.19	21.29	34.51	7.75	23.78	0.37	-35.97	38.26	-83.76	39.11	-0.12
y	-8.77	30.38	27.67	26.89	-0.03	-18.04	60.11	22.35	30.20	0.09	-48.40	71.18	-139.17	84.06	-0.11
z	-15.44	28.87	14.97	21.90	0.17	-19.61	37.86	24.50	31.09	0.17	-4.42	79.42	-47.98	48.01	0.30

Table 2. Touch point distribution parameters (in pixels) per key in visible, partially-invisible and invisible conditions. μ_x and μ_y are the optimized key center coordinates, in relation to the geometric center of each key; σ_x and σ_y are standard deviations; ρ is the correlation.