Experimental Study of Stroke Shortcuts for a Touchscreen Keyboard with Gesture-Redundant Keys Removed

Ahmed Sabbir Arif [1,2], Michel Pahud[1], Ken Hinckley[1], and Bill Buxton[1] *

Microsoft Research, Redmond, WA, USA[1] and York University Dept. of Computer Science & Engineering [1,2]

ABSTRACT

We present experimental results for two-handed typing on a graphical QWERTY keyboard augmented with linear strokes for *Space*, *Backspace*, *Shift*, and *Enter*—that is, swipes to the right, left, up, and diagonally down-left, respectively. A first study reveals that users are more likely to adopt these strokes, and type faster, when the keys corresponding to the strokes are removed from the keyboard, as compared to an equivalent stroke-augmented keyboard with the keys intact. A second experiment shows that the keys-removed design yields 16% faster text entry than a standard graphical keyboard for phrases containing mixed-case alphanumeric and special symbols, without increasing error rate. Furthermore, the design is easy to learn: users exhibited performance gains almost immediately, and 90% of test users indicated they would want to use it as their primary input method.

Keywords: multi-touch keyboards; text entry; stroke input

Index Terms: H.5.2 Information Interfaces & Presentation: Input

1 INTRODUCTION

The proliferation of mobiles and tablets has led to widespread use of graphical touchscreen keyboards, and a corresponding user demand for efficient text entry techniques. Researchers have pursued many strategies to improve touchscreen typing, from non-QWERTY key layouts [22,30], to shape-writing entire words in a single stroke [32,33], to approaches that heavily multiplex keys and resolve ambiguous inputs through language models [6].

While such techniques can yield substantial performance advantages, they also often demand substantial investment of skill acquisition from users before performance gains can be realized. In practice, this limits how many users will stick with a new technique long enough to realize such gains.

This paper explores the performance impact of an alternative approach: augmenting graphical touchscreen keyboards with linear stroke shortcuts, i.e. short finger swipes. While we do not expect large performance gains (such as those observed for expert shape-writing users [32]), if stroke-augmented keyboards can offer significant performance benefits while maintaining a high degree of transfer from existing QWERTY touch-typing skills, this complementary approach could represent low-hanging fruit for improving tablet text entry.

Stroke-augmented QWERTY keyboards are well-known, yet nonetheless under-studied in the modern context of touchscreen text entry. While single-point-of-contact stylus input [4,5,13,15] has been widely considered, the performance of linear stroke gestures for two-handed typing on a multi-touch keyboard has not been subjected to experimental scrutiny.

Furthermore, we show that an unusual design decision—that of removing the *Space*, *Backspace*, *Shift*, and *Enter* keys made redundant by the stroke gestures—can actually lead to superior performance. Although our research unearthed one previous example of such a design in the context of stylus text entry on a handheld [4], to our knowledge the insight that removing four of the most heavily used keys could potentially improve touchscreen text entry has not been anticipated by the literature.

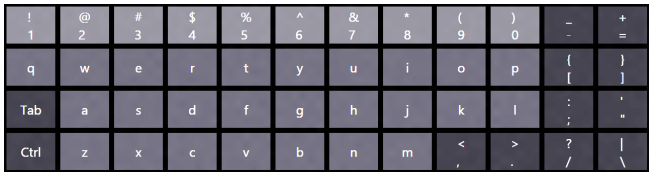

Fig. 1 The stroke-augmented graphical QWERTY keyboard tested in our studies. Note that the keyboard looks familiar despite the absence of the *Space*, *Backspace*, *Shift*, and *Enter* keys.

Likewise, despite previous related studies and examples, the design rationale for linear stroke keyboards (especially with the gesture-redundant keys removed) has not been fully articulated. Thus, in addition to possible time-motion efficiencies of the stroke shortcuts themselves, the design we pursued (*Fig. 1*) yields a number of interesting properties:

- Allowing the user to input stroke gestures for *Space*, *Backspace*, and *Enter* anywhere on the keyboard eliminates fine targeting motions as well as any round-trips necessary for a finger to acquire the corresponding keys.
- Instead of requiring two separate keystrokes—one to tap *Shift* and another to tap the key to be shifted—the *Shift* gesture combines these into a single action: the starting point selects a key, while the stroke direction selects the *Shift* function itself.
- Removing these four keys frees an entire row on the keyboard.
- Almost all of the numeric, punctuation, and special symbols typically relegated to the secondary and tertiary keyboards can then be fit in a logical manner into the freed-up space.
- Hence, the full set of characters can fit on one keyboard *while holding the key size, number of keys, and footprint constant.*
- By having only a primary keyboard, this approach affords an economy of design that simplifies the interface, while offering further potential performance gains via the elimination of keyboard switching costs—and the extra key layouts to learn.
- Although the strokes might reduce round-trip costs, we expect articulating the stroke gesture itself to take longer than a tap. Thus, we need to test these tradeoffs empirically.

Our studies demonstrate that overall the removal of four keys—rather than coming at a cost—offers a net benefit. Specifically, our experiments show that a stroke keyboard with the gesture-redundant keys removed yields a 16% performance advantage for input phrases containing mixed-case alphanumeric text and special symbols, without sacrificing error rate. We observed performance advantages from the first block of trials onward. Even in the case of *entirely lowercase* text—that is, in a context where we would not expect to observe a performance benefit because only the *Space* gesture offers any potential advantage—we found that the design illustrated in *Fig. 1* still performed as

* {mpahud, kenh, bibuxton}@microsoft.com; asarif@cse.yorku.ca

well as a standard graphical keyboard. Moreover, users learned the design with remarkable ease: 90% of users wanted to keep using the method, and 80% of test users believed they typed faster than on their current touchscreen tablet keyboard.

Our work thus contributes a careful study of stroke-augmented keyboards, filling an important gap in the literature as well as demonstrating the efficacy of a specific design; shows that removing the gesture-redundant keys is an unexpectedly critical design choice; and that earlier results assessing stroke shortcuts for single-point-of-contact stylus input (e.g. [15]) do not fully translate to the modern touch-typing context. Although our studies focus on the immediate end of the usability spectrum (as opposed to longitudinal studies over many input sessions), we believe the rapid returns demonstrated by our results illustrate the potential of this approach to complement other text-entry techniques.

2 Related Work

Touchscreen text entry is difficult and error prone due to small keys and less salient feedback [22,32,33]. This has led to wide exploration of alternative layouts [21,30], as well as techniques to support many characters on a small number of keys [6,22,25].

Yet, most commodity devices—including the iPad, Android tablets, and Windows 8 tablets—still employ a QWERTY key layout, while relegating numeric keys and most symbols to secondary keyboards or pop-up menus. For example, *Fig. 2* illustrates the primary, secondary, and tertiary keyboard layouts on a Windows 8 tablet, which we used as the "Default" keyboard for our study comparisons. The iPad and Android keyboards, while differing in various details, employ similar layout strategies.

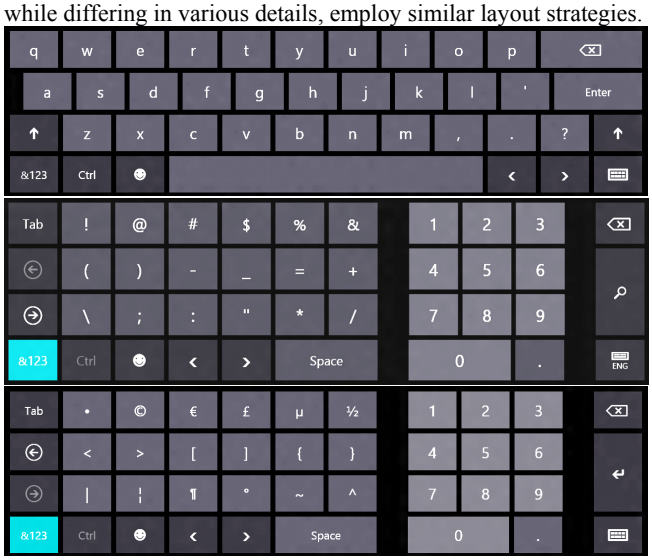

Fig. 2 Default Windows 8 graphical keyboard, with its primary (top), secondary (center) and tertiary (bottom) virtual keyboards.

Stroke-based techniques have often been proposed as alternatives to keyboards. Unistrokes [11], Graffiti [20,26], and EdgeWrite [29] all use a single-stroke shorthand to enter individual characters. Users can quickly achieve high performance with Graffiti [20] because of the similarity its symbols to normal letter-forms. While faster than hand-printing, maximum entry speeds are still typically far slower than multi-finger typing on graphical keyboards (at least for a skilled typist).

Unipad [19] augments single-stroke shorthand with language-based accelerators, including word completion, suffix completion, and frequently-used word prompting. Likewise, many modern touchscreen keyboards include some form of word prediction. Such complementary approaches could enhance stroke keyboards, but here we focus on the influence of the strokes themselves.

The alternate keyboard we study (*Fig. 1*) falls into a hybrid category of tap + stroke text entry. While our design differs from previous examples in its particulars, the basic concept of using linear strokes is well-known, and even removing the keys made redundant by the gestures has precedent [4]. Our contribution is to revisit this somewhat neglected approach for multi-touch keyboards, as well as to investigate how to fully realize its potential benefits.

Most prior examples emphasize single-point-of-contact graphical keyboards with stylus input. A 1995 patent [5] augments a standard graphical keyboard with linear stroke shortcuts for the *Space*, *Backspace*, *Shift*, and *Enter* keys, but several other systems demonstrate similar gestures [13,14,15,24]. In particular, the Microsoft Windows CE graphical keyboard (circa 2005, in Windows Mobile 5 and 6) supported these linear stroke shortcuts by default, and even provided an option to remove the gesture-redundant keys [4]. However, this was poorly documented, and to our knowledge no studies of the design have ever been published.

Isokoski [15] models a stroke-augmented keyboard, and assesses the model versus expert text entry performance for several keyboard layouts. Isokoski finds that strokes perform significantly slower at first, and only pull even with tap-based stylus entry after 20 experimental sessions. However, this is for stylus input, without the gesture-redundant keys removed, and using alternate keyboard layouts. These (and perhaps other) differences are crucial to study empirically in a more modern context; indeed, our results suggest that, in the right context, strokes can produce performance gains almost immediately.

Shape-writing enables input of words via gestures on top of a QWERTY graphical keyboard [16,31,32,33]. Although it takes practice, the technique is gaining traction commercially [27] because it is fast and easy to learn the most common words.

Techniques such as 1Line [6] and MessageEase [25] use strokes to reduce footprint, whereas our goal is to promote more keys to the primary keyboard. Other techniques extend multi-touch gestures to non-alphanumeric input [8,9] and text editing [10].

Grossman et al. [12] study users' ability to learn and recall keyboard hotkeys: users learn such keyboard shortcuts much faster, and are more likely to use them, after they have trained in a condition where the shortcuts offer the *only* way to select commands from a menu. We explore a similar choice in the context of text entry by assessing performance with or without removing the gesture-redundant keys.

3 Details and Considerations for the Stroke Keyboards

The aim of our studies was to investigate hybrid tap and stroke keyboards, where the user inputs ordinary alphanumeric characters via taps, and the gestures by short linear strokes.

The specific stroke gestures we employed, designed to maintain stimulus-response compatibility with the corresponding actions, were assigned as follows:

- *Space*– stroke to the right, anywhere on the keyboard;
- *Backspace*– stroke to the left, also anywhere;
- *Enter*– stroke diagonally, again anywhere, down to the left;
- *Shift*– stroke up, starting from the desired key.

Based on pilot study data, we interpreted any finger movement of less than 4.7 mm as a tap. Likewise, to input a stroke gesture, the finger had to slide further than this threshold; we found this cleanly separated taps from strokes.

3.1 Keyboard Variants Implemented for Study

To assess the impact of these strokes, with or without removing the corresponding keys, we implemented the following three keyboard variants in our own instrumented code-base:

- ***Default:*** First, we implemented a keyboard that conformed pixel-for-pixel to the default Windows 8 keyboard (*Fig. 2*), using identical colors, key sizes, and layouts.
- ***Default + Strokes:*** Second, we implemented a variant that supported the strokes enumerated above on top of the "Default" keyboard. Visually it looked identical to the Windows 8 keyboard (*Fig. 2*) in all respects, except that swiping a finger produced the desired stroke shortcut.
- ***Removed + Strokes:*** Third, we implemented the keys-removed design using the layout shown in *Fig. 1*. We maintained a likeness to QWERTY wherever possible. For instance, the **!** character appears as the shift symbol on the **1** key, the **:** character appears above the **;** key, and so on.

It is critical to re-emphasize here that these designs—including the *Removed + Strokes* design—all kept the overall keyboard footprint constant. We also kept the standard key sizes absolutely constant. Other details such as key colors and highlight cues (on finger contact) were also identical.

None of the studied keyboards provided auditory click feedback for key presses. Also note that none of them (including the default Windows 8 keyboard) included ESC or CAPS LOCK keys.

Of course, the *Removed + Strokes* design necessitates shifting some keys around to use the freed-up space. We carefully designed and considered the symbol placements and other minor key layout differences. Also note that our current design arranges the keys in rectilinear rows and columns, rather than a more traditional staggered key pattern where alternate rows are shifted slightly (*Fig. 2, top*). This reflects the preliminary nature of our design, rather than any desire on our part to avoid a staggered layout; indeed, as of this writing, we are working on a slightly improved layout that includes staggered keys.

Thus, while removal of the keys is the primary intervention of interest, these various aspects (including but not limited to key removal) may influence its performance. However, since our goal was to empirically test a practical and usable instance of the keys-removed idea, we felt this trade-off would yield the most insightful performance data for such a design.

4 EXPERIMENT 1: STROKES AS ALTERNATIVE VS. KEY REMOVAL

The purpose of our first study was to assess whether it was necessary or desirable to remove the gesture-redundant *Space*, *Backspace*, *Shift*, and *Enter* keys in order for users to achieve the best performance.

As such, Experiment 1 compares the *Default+Strokes* keyboard detailed above to the *Removed+Strokes* design. If the strokes offer significant benefit without having to remove the keys, this would support including them on status quo QWERTY graphical keyboards. If not, it would suggest that the (arguably) counter-intuitive step of removing four very frequently used keys might afford superior performance.

At present, our focus is to evaluate how the strokes impact performance, so we simply told users about the presence of the strokes when they first encountered the keyboards, but clearly more sophisticated self-revelation mechanisms for the gestures could be devised (e.g. [3,18]).

For the *Default+Strokes* keyboard, users were free to choose either approach (tapping keys, or making a stroke) when they needed to input *Space*, *Backspace*, *Shift*, and *Enter*.

For the *Removed+Strokes* design, since the corresponding keys were absent, the only option for users was to employ the strokes. However, users clearly recognized the keyboard when they first encountered it; indeed, due to its likeness with normal QWERTY, many users did not notice that the keyboard was "different" until they tried to start typing on it.

4.1 Apparatus – Experiment 1

We used a Samsung Series 7 tablet (11.66″ × 7.24″ × 0.51″, with 1366 × 768 screen resolution at 135 pixels/inch) for our study, running the Windows 8 Release Preview. We placed the device on a desk with a custom stand that tilted the device to a comfortable ~15° typing posture (Fig. 3).

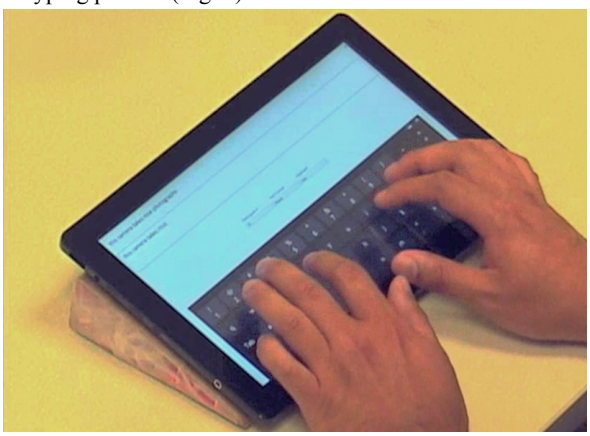

Fig. 3 User typing on a tablet, as presented in all our studies.

4.2 Participants – Experiment 1

Fourteen people participated, from 23-49 years old (average 32). Five were female and two were left-handed. All had at least ten years of experience on QWERTY. Eleven participants used a touch-based device, such as mobile phone, on a regular basis. Eight owned an iPad tablet; three used their device frequently to input text such as emails or notes (almost every day), three used it occasionally (about three days a week), while two used it rarely (once a week or less). All participants were native or fluent speakers of English, and each received a gratuity for participation.

4.3 Design – Experiment 1

The study compared the *Default+Strokes* technique to the *Removed+Strokes* design. We counterbalanced the order of conditions via a standard Latin square, in a design as follows:

14 participants ×
2 techniques (*Default+Strokes* vs. *Removed+Strokes*) ×
3 blocks × 10 phrases
= 840 phrases in total.

The study included two different types of phrases, *Regular* and *Mixed*, with 5 of each type (in random order) per block.

Regular phrases contained only lowercase letters and spaces, drawn from the MacKenzie-Soukoreff [23] corpus (with all British spellings changed to American usage). We chose this corpus due to its high correlation with English language character frequencies and its wide use in text entry studies.

Mixed phrases included **7%** uppercase characters, **10%** numeric characters, **7%** symbols, with the other **76%** lowercase and spaces, using a custom corpus as further detailed below.

Thus, we carefully bracketed both keyboards' performance by testing with all-lowercase conditions, as well as mixed-case conditions. This therefore gives an honest assessment of how the keyboards perform. In real life scenarios, users often must input such phrases, and the difficulty in entering such symbols is a major shortcoming of present graphical keyboards.

4.4 Procedure – Experiment 1

Participants entered short phrases, which appeared on-screen in a dialog, using the assigned keyboard. We instructed participants to take the time to read and understand the phrases in advance,

then to enter them as quickly and accurately as possible. When finished, participants pressed the *Enter* key (or made an *Enter* gesture, based on condition) in order to see the next phrase. All participants used both hands to type, and were allowed to rest between conditions, blocks, or trials.

Direct insertion point control (touch-drag to place the carat or select text) was disabled during the studies, so users had to employ *Backspace*, exclusively, to repair any errors. We instructed participants to correct any errors as soon as they noticed them, but they could ignore ones that were more than 10 characters back. This ensured that participants did not have to delete a lot of text to correct a single mistake.

Although experts notice and correct over 98% of mistakes within five characters [2], less experienced users (such as the ones recruited in this study) might input more characters before noticing an error. Thus, we believe this contributed to the somewhat elevated error rates (in all conditions) reported below. This also motivated us to recruit more experienced touchscreen-typing users for Experiment 2, as discussed later in this paper.

Our custom corpus generated phrases containing uppercase, numeric, and special characters in known patterns, such as URLs, email addresses, phone numbers, and mailing addresses. The corpus placed items from lists of places, addresses, and relationships into the phrases in a meaningful manner, such as: "My **son's** phone number is **+1 (638) 283-9375**". Here, the first bolded fragment was generated from the relationships list, and the latter was produced from the phone-number pattern. The *Mixed* phrases averaged seven words: the average phrase length was 36.81 characters and the average word length was 4.46 characters.

4.5 Results – Experiment 1

Timing started from the finger-down event for the first character and ended with the finger-up event of the last character. We used the standard Words per Minute (WPM) metric for entry speed. We computed Error Rate (ER), which counts only incorrect characters in the transcribed text, as well as Total Error Rate (TER)—which unlike ER also includes corrections of erroneous keystrokes [1]—as our error metrics. (Although not all researchers agree on the utility of TER, we report it here for additional perspective.) Note also that a lowercase character in place of an uppercase was counted as an error. Finally, Operations per Character (OPC) is the average number of actions required to produce one character.

An ANOVA revealed a significant effect of *Technique* on entry speed ($F_{1,13} = 9.44$, $p < 0.01$). For this task *Removed+Strokes* performed significantly faster than *Default+Strokes* with averages of 17.61 WPM ($SE = 0.28$) vs. 15.75 ($SE = 0.25$), respectively.

Removed+Strokes also required significantly fewer Operations per Character than *Default+Strokes* ($F_{1,13} = 36.0$, $p < .0001$), with average OPC of 1.22 (SE = 0.04) and 1.51 (SE = 0.02), respectively. Hence, removing the keys yielded 11.8% faster text entry with 19.2% fewer actions required of the user, as compared to the *Default+Strokes* keyboard.

There was no significant difference for ER ($F_{1,13} = 0.32$, ns) or of *Technique* × *Block* ($F_{2,26} = 0.81$, ns). On average ER for the *Removed+Strokes* and *Default+Strokes* keyboards were 10.60 (SE = 5.53) and 6.92 (SE = 2.04), respectively. Similarly, there was no significant difference for the TER metric ($F_{1,13} = 2.93$, $p = 0.1$), nor any significant effect of *Technique* × *Block* ($F_{2,26} = 1.09$, $p = 0.3$). On average TER for the *Removed+Strokes* and *Default+Strokes* keyboards were 10.15 (SE = 0.52) and 12.58 (SE = 0.63), respectively. However, the disparity between ER (6.92) and TER (12.58) for the *Default+Strokes* keyboard suggests that users committed and corrected more errors with *Default+Strokes* than *Removed+Strokes* (which exhibited similar ER (10.60) and TER (10.15) metrics). This reflects the fact that only the TER metric takes corrected errors into account [1].

4.5.1 Gesture Use

Further analysis revealed that in cases where gestures could have been used to input characters, on average participants used gestures during the *Default+Strokes* condition only 28.3% ($SE = 5.66$) of the time, ranging from as little as 4% of the time to a maximum of 58% usage. A Kruskal-Wallis test found this to be highly significant ($H_1 = 160.16$, $p < .0001$), as compared to the 100% occurrence of gestures in the *Removed+Strokes* condition.

While strokes were used by necessity in the latter condition, it is noteworthy that users did not complain about having to use the gestures. Rather, they often made positive remarks about them ("the gestures were really intuitive," "once I got a hold of the gestures my typing got really fast").

4.6 Summary of Findings – Experiment 1

This pattern of results strongly suggests that removing the gesture-redundant keys was essential to make stroke shortcuts a worthwhile enhancement to the keyboards we tested. The *Removed+Strokes* design resulted in faster text entry, and about 19% fewer Operations per Character, without any significant difference in error rates.

The *Default+Strokes* participants tended to use the strokes infrequently and achieved little, if any, benefit from them, despite knowing that they were available as an optional shortcut—and their performance suffered for it. We therefore pursued the keys-removed design as the foundation for the stroke keyboard in the main experiment that follows.

5 Experiment 2: Main Experimental Study

The purpose of our second experimental study was to evaluate the performance of the *Removed+Strokes* design in comparison to the *Default* keyboard with tap input only (no strokes). The apparatus was exactly as described for Experiment 1.

5.1 Participants – Experiment 2

Twenty participants, none of whom participated in Experiment 1, took part in the study. All had a minimum of 10 years of experience on QWERTY, and all had owned an iPad tablet for at least six months and typed on it frequently, such as to write emails, for at least three hours three times a week.

Note that this is a higher, and more carefully controlled, level of experience than the participants we recruited in the first study. We wanted users more familiar with touchscreen tablet keyboards, and at present, such users are most readily recruited in significant numbers for the iPad. Furthermore, current iPad users are not familiar with the Windows 8 keyboard, and thus come at both the *Default* and *Removed+Strokes* designs with fresh eyes. They also represent the population of users most likely to encounter our proposed technique if it were to be widely deployed.

The participants we recruited were then randomly assigned to the following groups:

- ***Default* Group:** The ten participants in this group used the *Default* keyboard during the study. Their age ranged from 19 to 49 years (average 39.3). Five were male, one was left-handed.
- ***New* Group:** Ten participants used the *Removed+Strokes* technique—which for brevity we henceforth call the ***New*** keyboard. Their age ranged from 26 to 44 years (average 33.6). Six were male and all were right-handed.

We ran Experiment 2 as a **between-subjects study** to allow participants to input as many phrases as possible with each keyboard, while avoiding any possibility of skill transfer or other interference effects between the two conditions.

5.2 Procedure – Experiment 2

Participants entered phrases following the same procedure as Experiment 1. However, in Experiment 2 the *Regular* and *Mixed* phrases were presented in two separate conditions rather than in random order within blocks.

In the first condition users entered *Regular* text (all-lowercase from a standard corpus [23]). Note that this tests the *New* keyboard in a setting where it would have the least possible benefit—that is, for phrases that contain no uppercase or special characters. In the second condition users entered *Mixed* text (from our custom corpus as described in Experiment 1). In addition to the WPM, ER, and TER metrics, participants in Experiment 2 also responded to a questionnaire with 7-point Likert scales for their assigned keyboard.

5.3 Design – Experiment 2

Participants were thus equally divided and assigned to either the *Default* or the *New* group, in an experimental design as follows:

Between-subject group: *New vs. Default* ×
10 participants per group ×
2 conditions (*Regular vs. Mixed text types*) ×
3 blocks × 15 phrases (+ 2 additional practice phrases)
= 1800 phrases per group (excluding practice phrases).

Note that all participants started with *Regular* text, so that we could assess their typing performance with lowercase text first.

5.4 Results – Experiment 2

5.4.1 Text Entry Speed

An ANOVA revealed no main effect of *Keyboard* on entry speed ($F_{1,18} = 0.42$, ns), but the overall means conceal the fact that there was a highly significant effect of *Keyboard* × *Text Type* ($F_{1,18} = 9.68$, $p < .01$), which echoes the results found in Experiment 1: the *New* keyboard was faster than the *Default* keyboard, but only for the *Mixed* phrase sets.

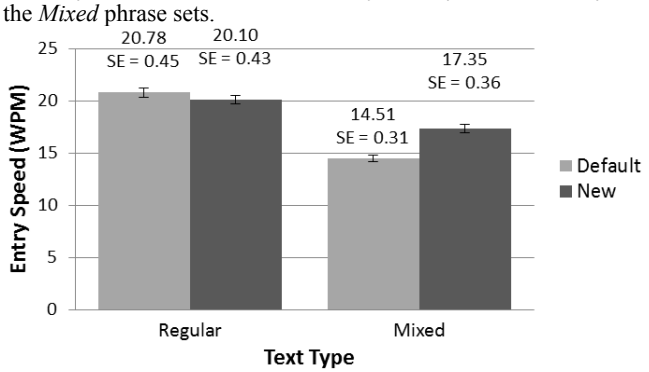

Fig. 4 Keyboard entry speeds for *Regular* vs. *Mixed* text.

Average entry speed for *Default* in the *Regular* and *Mixed* conditions was 20.78 (*SE* = 0.45) vs. 14.51 (*SE* = 0.31) WPM, respectively. By contrast, average entry speed for the *New* keyboard with regular and mixed text were 20.10 (*SE* = 0.43) and 17.35 *WPM* (*SE* = 0.36), respectively (*Fig. 4*). A Tukey-Kramer test (MSE = 396.67, *df* = 18, $z_\alpha = 3.08$) confirmed that inputting *Mixed* phrases was significantly faster with the *New* keyboard than the *Default* one ($p < .05$), but there was no significant difference for *Regular* phrases ($p > .05$).

5.4.2 Errors (ER and TER metrics)

There was no significant effect of *Keyboard* on either ER ($F_{1,18} = 0.89$, ns) or TER ($F_{1,18} = 0.28$, ns). The *Keyboard* × *Text Type* interaction was also not significant. This confirms the benefits observed for the *New* keyboard did not come at the cost of increased error rates (*Fig. 5)*.

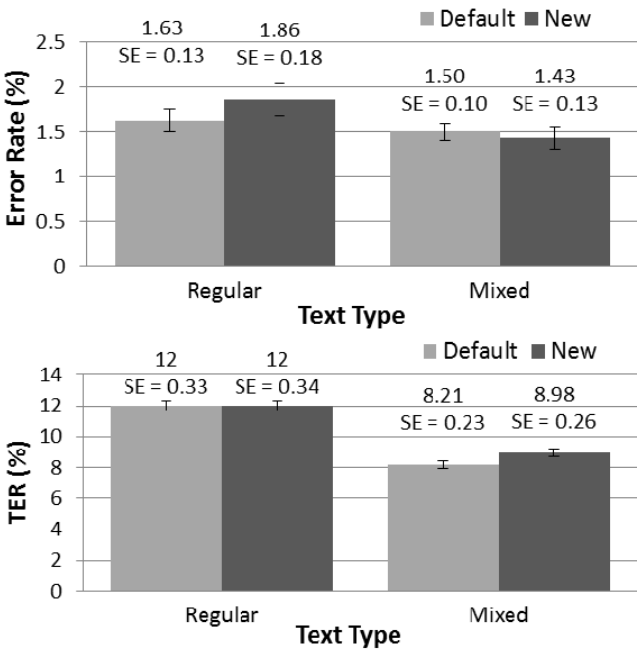

Fig. 5 ER (top) and TER (bottom) for *Regular* vs. *Mixed* text.

5.4.3 Operations per Character

An ANOVA revealed a significant effect of *Keyboard* on OPC ($F_{1,18} = 6.91$, $p < .05$). There was also a significant effect of *Keyboard* × *Text Type* ($F_{1,18} = 14.63$, $p < .005$). Average OPC with the *Default* keyboard was 1.37 (SE = 0.03) for *Regular* text and 1.42 (SE = 0.03) for Mixed text. By contrast, OPC for the *New* keyboard was 1.33 (SE=0.03) for *Regular* text and 1.21 (SE=0.02) for *Mixed* text. A Tukey-Kramer test (MSE = 0.189, *df* = 18, $z_\alpha = 2.64$) confirmed that the *New* keyboard required significantly fewer OPC than the *Default* keyboard ($p < .05$). There were no significant effects or interactions with *Block* for OPC.

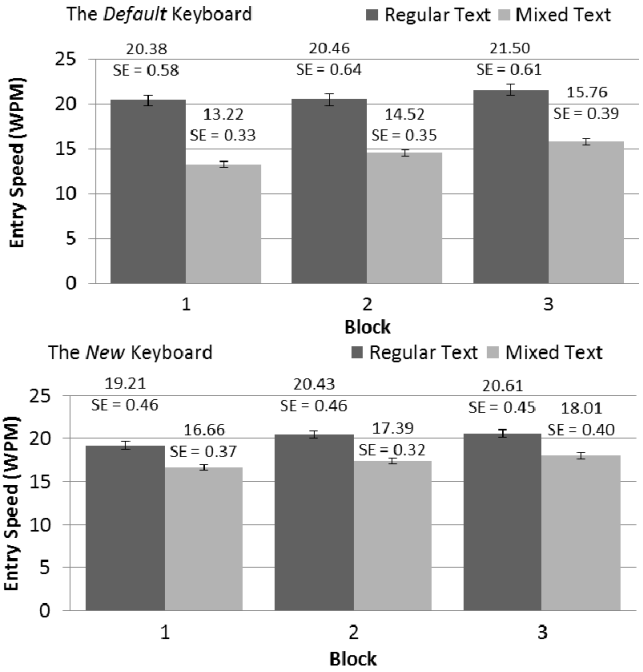

Fig. 6 Entry speeds (WPM) broken out per block.

5.4.4 Learning Effects on Text Entry Speed across Blocks

The ANOVA showed a significant effect of *Block* on entry speed for the *Default* keyboard ($F_{2,9} = 11.40$, $p < .001$) as well as for the *New* keyboard ($F_{2,9} = 4.61$, $p < .05$). This shows that learning occurred with both keyboards across blocks (*Fig. 6*), but the lack of any significant *Text Type* × *Block* or *Keyboard* × *Text Type* × *Block* interactions suggests there was not a strong difference in the rates of learning observed by either *Keyboard* or *Text Type*.

Since our users were all experienced with the iPad, the learning effect observed for the *Default* condition does not reflect any lack of expertise for our participants with touchscreen text entry; it simply reflects that the Windows 8 touch keyboard is not identical to the iPad keyboard.

5.5 Analysis of Round Trip Times – Experiment 2

To gain additional insights, we compared the round-trip costs for *Shift* and the other functions as follows.

5.5.1 Round-Trip Time Including Preparatory Motions

We computed the total round-trip time, $T_{round}{}^t$, for the *Space*, *Backspace*, *Enter* and *Shift* functions on both the *New* keyboard and the *Default* keyboard. This considers the "true cost"—in the sense advocated by Dillon [7]—of employing these gestures, including:

- Time for preparatory motion following the *previous* action, up to the onset of finger contact with the screen.
- The time spent with the finger in contact with the screen to input the current gesture (a tap or stroke, depending on the keyboard type of *Default* or *New*, respectively).
- All motion following lift-off, up to the *next* finger contact with the screen.

We accounted for all of these time-motion costs (which might also include hesitations or mental preparation) in the round trip time so that we could be sure to consider any and all impacts of a gesture on the surrounding operations.

5.5.2 Impact of Keyboard Switching on Performance

During the *Mixed* text condition with the *Default* keyboard, our results show that fully 7.79% of all keystrokes served solely to switch between the different keyboard layouts.

On the *Default* keyboard, the average round trip time for keyboard layout swapping was over two seconds (2124.2 ms, *SE* = 33.2), a cost which the *New* design completely eliminates. We believe this is a major source of the performance benefits for the *New* design on *Mixed* phrases.

Further investigation revealed that 5.08% of all keystrokes on the *Default* keyboard, and 9.25% of all keystrokes and gestures on the *New* keyboard, were made to input *Shift*. This was somewhat expected because users always had to use *Shift* to input special characters on the *Default* keyboard, but we also observed that users often also pressed *Shift* to see which special characters were available on the primary layout. Our interpretation of this behavior was that it highlights how difficult it is even for experienced users to memorize the secondary and tertiary keyboard layouts for numbers and special symbols.

5.5.3 Impact of Individual Gestures on Round-Trip Time

An ANOVA on $T_{round}{}^t$ revealed a significant effect of *Keyboard* on round-trip time for *Backspace* ($F_{1,9} = 97.23$, $p < .0001$) but not for *Space*, *Enter*, or *Shift*. A Tukey-Kramer test (MSE = 78239.63, $df = 18$, $z_\alpha = 2.97$) revealed that $T_{round}{}^t$ for *Backspace* was significantly longer for the *New* vs. the *Default* keyboard ($p < .05$). This suggests the *Backspace* gesture, which users often apply in rapid succession, could benefit from design improvements to make repeated invocation faster, such as by issuing additional backstrokes in proportion to the stroke length.

Finally, *Block* was also significant for all four keyboard actions, indicating that users' performance was becoming more efficient with *both of the keyboards* over the course of the experiment. These trends are illustrated in *Fig. 7*.

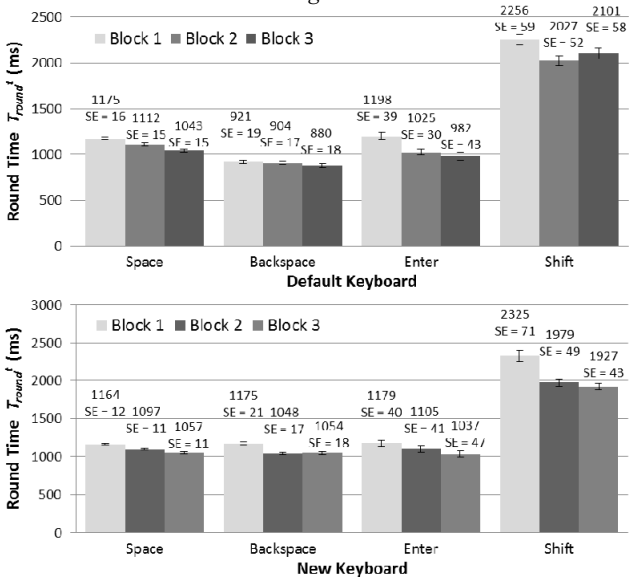

Fig. 7 Per-key round trip times for *Default* (top) vs. *New* (bottom).

5.6 Qualitative Findings

At the conclusion of the experiment, we asked users to compare the keyboard they had just tried—either our facsimile of the standard Windows 8 keyboard in the *Default* condition, or the stroke keyboard with redundant keys removed in the *New* condition—with the touchscreen keyboard on iPad (which all of our recruited participants used heavily). Hence, because the Windows 8 touchscreen keyboard was unfamiliar to all users, both the *Default* and *New* conditions were novel to users in each group.

5.6.1 Keyboard Preference and Willingness to Use

A Kruskal-Wallis test on the questionnaire data indicated significance with respect to user preference across techniques ($H_1 = 18.05$, $p < .05$). 80% of the *New* keyboard users liked the new keyboard, while only 40% of the *Default* keyboard users liked it compared to their existing method for tablet touchscreen text entry (i.e. the iPad graphical keyboard).

We also asked users how willing they were to use the keyboard compared to the iPad keyboard that they were familiar with. Users also responded significantly positively to this question ($H_1 = 7.24$, $p < .05$), with 90% of the *New* keyboard users expressed their interest in using it as their primary input method, while only 50% of the *Default* keyboard users expressed willingness to use it compared to the their existing method.

5.6.2 Overall Experience with the New Keyboard

Users were extremely positive regarding their overall typing experience with the *New* keyboard. All of our participants found the *New* keyboard comfortable to use. They all agreed that the gestures used to replace the *Space*, *Backspace*, *Shift*, and *Enter* keys were intuitive and easy to remember. Interestingly, even though our empirical results showed there was no significant difference in entry speed while inputting lowercase text with the *New* keyboard, 80% of users *thought* their typing speed increased substantially with the *New* keyboard even when typing such text.

Users in the *New* group quickly grew to like the stroke gestures (typical comments were that "it didn't take me much time to learn

the gestures," "once I knew them I started using it like a regular keyboard," and "The keyboard was more responsive than my normal tablet keyboard"), although at least one user did verbalize that it "takes time to get used to it."

Users appreciated having a single keyboard (commenting that "I liked the fact that you don't have to swap between different keyboards to enter special characters" and "This is perfect for entering URL's, digits, addresses.") They also found the stroke keyboard comfortable and easy to use ("I liked how it felt under my hands," and "I liked that I could apply the gestures without moving my fingers").

Of course, although the balance of the feedback was positive, some users did comment on issues they encountered. In particular, in the Likert-scale questions, 80% of users had a greater than neutral response when asked if they sometimes triggered unintended gestures by accidentally brushing the screen. Some users commented on this ("I sometimes accidentally performed gestures, which caused errors"). Accidental sliding of the entire tablet if the user applied too much pressure when gesturing was also occasionally an issue. The *Shift* gesture seemed most prone to this issue, e.g. one user commented that "I had difficulties performing the Shift stroke, which made my typing slower."

While we cannot distinguish intentional versus accidental touches in our experimental data, we did *not* observe a statistically significant difference in error rate between the techniques. However, since users did perceive a higher incidence of errors, this is an area of concern that should be examined further in future work. Nonetheless, as reported above, 90% of the *New* keyboard users still expressed interest in using the strokes as their primary input method, and many asked if they could download the *New* keyboard to use on their personal tablet device.

A few users did mention the missing keys ("I missed the space bar, I would have liked an option to enable it") or other functionality absent from our prototype implementation ("It would have been great if the new keyboard augmented word prediction"). Another user observed the difficulty in correcting errors ("Backspace is time consuming, especially when multiple characters needed to be deleted. In a regular keyboard, I can tap-hold the Backspace key to delete multiple characters.") Future studies and design refinements should explore these options and capabilities.

6 Discussion

The results clearly showed that the *New* keyboard provided faster text entry for *Mixed* text phrases, at near-identical error rates, suggesting that the benefits of the design do not come at the cost of a speed-accuracy tradeoff. But just as importantly, even in the case of all-lowercase text where we would not expect the *New* keyboard to offer much benefit, it was statistically equivalent to the status-quo keyboard—from the very first block of trials onward. This emphasizes that Isokoski's findings [15] were in a different context (single-touch stylus input on a handheld, among other differences) that does not fully translate to multi-touch typing on a tablet, at least for the designs we tested.

Likewise, since we observed our results after only three blocks of 15 phrases, it strongly suggests users do not require extensive training to start using the new design and realize performance gains from it.

Although in future work we would like to examine learning trends across a much larger number of trials, our current focus on the immediate end of the usability spectrum acknowledges the practical reality that users are unlikely to adopt a new technique if they do not find it useful in the short term. Even in the limited number of trials examined by our current studies, users adapted to the new technique almost immediately—unlike many alternate keyboard layouts or other text-entry enhancements reported in the literature, the benefits we observed did not require a long period of skill acquisition. This, we believe, further highlights the potential of the approaches we studied. In our study, each user inputted 90 total phrases, which took about an hour including practice and breaks; this was about as long as a single session could go before fatigue effects would start to set in.

Furthermore, our studies demonstrate that even issues as subtle as whether or not the gesture-redundant keys are removed from the keyboard can significantly influence user behavior and success with the technique. Hence our studies also contribute the novel result that the removal of the redundant keys is crucial to the successful realization of linear stroke shortcuts.

6.1 Tapping Rhythm

We acknowledge that the standard tap-tap rhythm of typing cannot be maintained with the *New* keyboard. However, this is not to say that a new and perhaps equally effective typing rhythm cannot be established. In our own use of the keyboard, and in observing a few of the fastest users in Experiment 2, we have noticed a tendency for a rhythm of fast taps and ballistic strokes to emerge. Furthermore, a strategy of entering the *Space* gesture by quickly scrubbing *right* with the thumb of the *left* hand—and *Backspace* using the thumb of the opposite hand—particularly lends itself to such a rhythm. However, more trials (or preferably a longitudinal study) would be necessary to establish whether users naturally pick up such skills. We also plan to study whether performance with the technique might be enhanced by gradually steering users towards this strategy somehow.

6.2 Performance Modeling and the Phrase Corpus

Predicting the performance of a hybrid keyboard, such as the one presented here, is difficult with existing models, as they either focus on one-handed, single point text entry (e.g. with a stylus), or do not account for hybrid tap and stroke inputs [2]. While we chose to start by collecting performance data, it should be possible to devise a model for hybrid designs using KLM or GOMS-style techniques; this would then enable rapid model-based exploration of many additional design permutations.

The choice of corpus is also critical: in our study we used a standard corpus with all-lowercase text, as well as a custom corpus for mixed-case phrases containing symbols and numbers. Another approach would be to employ a corpus of actual mobile email data (e.g. [28]), but one issue with such an approach is that current keyboard designs discourage the entry of special symbols because they are inefficient to enter. Hence one would likely see a corresponding absence of those symbols in such a corpus, due to users' natural tendency to avoid tedious tasks. Thus, a range of approaches seems to be called for in terms of the ideal corpus for studies—as well as deployment and longitudinal evaluation of our keyboard to users for their real work, which we are also pursuing as of this writing.

6.3 Self-Revelation (Learnability) of Stroke Gestures

One key issue with removing the keys, of course, is that if users are confronted by a keyboard with no *Space*, *Backspace*, *Shift*, or *Enter* key—and without any instruction whatsoever—they may quite justly feel uncertain how to proceed. However, in our studies we found that a brief verbal explanation was enough to get users started. Furthermore, a variety of suitable self-revelation techniques appear in the literature [3,17,18]. In our current deployment of a refined version of the keyboard studied here, we have found that a short one-minute video tutorial sufficient to get users started on the stroke shortcuts.

Another approach follows from Grossman et al.'s work on hotkey adoption: *only* offer the shortcut, which forces users to adopt it and learn it quickly [12]. Indeed, this is exactly the approach we follow with the removal of redundant keys. While draconian in one sense, users liked the new approach and clearly performed better when using it, as opposed to a design which left both methods available—and hence implicitly forced people to reflect on and decide which of the two approaches (key press or shortcut gesture) to use while typing—which, by foisting a design choice on the user, is draconian in a different sense. That users failed to adopt the gestures, then, when they were faced with such a choice is perhaps another reminder that sometimes 'less is more,' supported in this case by the clearly superior performance and user preference we observed with redundant keys removed.

However, with perhaps a different technique for self-revelation, it is possible that a linear stroke keyboard design that does *not* remove the redundant keys could still be realized—but we certainly did not discover any support for that in our studies. Furthermore, the rapidity with which users took to the new keyboard, and the preference expressed by many users for the new design after a relatively brief exposure, suggests that it is worth investing further effort in devising ways for the technique to reveal itself to users.

7 Conclusion and Future Work

Perhaps the primary virtue of the stroke keyboard designs studied in this paper is that they require little change to existing habits of use—even (and perhaps especially) when the *Space*, *Backspace*, *Shift*, and *Enter* keys are removed. As our studies revealed, users benefitted from the latter design almost immediately. While we do not claim linear stroke shortcuts can rival the peak performance expert users might attain with techniques such as shape-writing, linear stroke keyboards are worth studying and understanding as an interesting class in their own right because of the extremely low barrier to entry, which affords a smaller but still significant performance benefit with minimal investment in skill acquisition.

Furthermore, it may be possible to combine linear stroke keyboards and shape gestures in future work (a design path that we are actively exploring), which underscores the need to carefully isolate and analyze the potential contribution of linear stroke shortcuts to the touchscreen text-entry problem.

Finally, by taking the design step of removing the gesture-redundant keys—which while not without precedent [4], has never been quantitatively studied before—the *Removed+Strokes* ("*New*") keyboard design tested here offers a simple, single-level keyboard design that almost entirely eliminates the need to switch to secondary and tertiary virtual keyboards for entering special characters, thus saving keystrokes while also avoiding the visually and cognitively jarring changes of the keyboard context required by touchscreen keyboards in common use today.

Looking forward, our research program aims to further advance this approach. We are currently conducting a longitudinal deployment of a stroke keyboard with the gesture-redundant keys removed in the context of users' day-to-day text entry. Whether stroke-based designs can exhibit advantages in other contexts—such as mobile phones, large-format displays, or one-handed text entry—remains to be demonstrated. We also intend to explore the potential of adding other enhancements, such as word prediction, shape-writing gestures, and integrated text editing and selection techniques [10].

8 References

[1] Arif, A.S., Stuerzlinger, W. Analysis of text entry performance metrics. *Proc. IEEE TIC-STH 2009*.

[2] Arif, A.S., Stuerzlinger, W. Predicting the cost of error correction in character-based text entry technologies. *CHI '10*.

[3] Bragdon, A., et al. GestureBar: improving the approachability of gesture-based interfaces. *CHI '09*.

[4] Buxton, B. *Too Hidden Features of the Windows CE Graphical Keyboard*. Sept. 17, 2013. Available from: www.billbuxton.com/Windows%20CE%20Graphical%20Keyboard.pdf.

[5] Buxton, W., Kurtenbach, G., *Graphical keyboard, U.S. Patent 6,094,197, May 17, 1995*.

[6] Chun Yat Li, F., et al. The 1Line Keyboard: A QWERTY Layout in a Single Line. *UIST'11*.

[7] Dillon, R.F., et al. Measuring the True Cost of Command Selection: Techniques and Results. *CHI'90*.

[8] Findlater, L., et al. Beyond QWERTY: augmenting touch screen keyboards with multi-touch gestures for non-alphanumeric input. *CHI '12*.

[9] Findlater, L., et al. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. *CHI '11*.

[10] Fuccella, V., et al. Gestures and Widgets: Performance in Text Editing on Multi-Touch Capable Mobile Devices. *CHI'13*.

[11] Goldberg, D., Richardson, C. Touch-typing with a stylus. *CHI '93*.

[12] Grossman, T., et al. Strategies for accelerating on-line learning of hotkeys. *CHI '07*.

[13] Hashimoto, M., Togasi, M. A virtual oval keyboard and a vector input method for pen-based character input. *CHI '95*.

[14] Hot Virtual Keyboard. http://hot-virtual-keyboard.com. 2012.

[15] Isokoski, P. Performance of menu-augmented soft keyboards. *CHI '04*.

[16] Kristensson, P.-O., Zhai, S. SHARK 2: a large vocabulary shorthand writing system for pen-based computers. *UIST'04*.

[17] Kurtenbach, G., Buxton, W. Issues in combining marking and direct manipulation techniques. *UIST'91*.

[18] Kurtenbach, G., et al. Contextual Animation of Gestural Commands. *Proc. Graphics Interface'94*.

[19] MacKenzie, I.S., et al. Unipad: Single-stroke text entry with language-based acceleration. *ACM NordiCHI 2006*.

[20] MacKenzie, I.S., Zhang, S.X. The immediate usability of graffiti. *Proceedings of the conference on Graphics interface '97*..

[21] MacKenzie, I.S., Zhang, S.X. The design and evaluation of a high-performance soft keyboard. *CHI'99*.

[22] MacKenzie, I.S., et al. Text entry using soft keyboards. Behaviour & Information Technology, 1999. **18**: p. 235-244.

[23] MacKenzie, S., Soukoreff, W., Phrase sets for evaluating text entry techniques. *CHI'03 Extended Abstracts*.

[24] Masui, T. An Efficient Text Input Method for Pen-based Comptuers. *CHI'98*.

[25] MessageEase. *MessageEase keyboard for touch screen devices*. 2012. Available from: www.exideas.com.

[26] Palm Inc. *Getting Started*. 2000. Available from: http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/a/pdf/Palm%20m500%20User%20Manual.pdf.

[27] Swype. http://www.swypeinc.com. 2012.

[28] Vertanen, K., Kristensson, P.O. A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails. *MobileHCI'11*.

[29] Wobbrock, J., et al. EdgeWrite: A Stylus-based text entry method designed for high accuracy and stability of motion. *UIST'03*.

[30] Zhai, S., et al. The Metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. *UIST'00*.

[31] Zhai, S., Kristensson, P.O. Shorthand writing on a stylus keyboard. *CHI '03*.

[32] Zhai, S., Kristensson, P.O., The word-gesture keyboard: reimagining keyboard interaction. Commun. ACM, 2012. **55**(9): p. 91-101.

[33] Zhai, S., Kristensson, P.O., Appert, C., Andersen, T.H., Cao, X., Foundational Issues in Touch-Surface Stroke Gesture Design - An Integrative Review. Foundations and Trends in Human-Computer Interaction, 2012. **5**(2): p. 97-205.