TiltWriter: Design and Evaluation of a No-touch Tilt-based Text Entry Method for Handheld Devices

Steven J. Castellucci
Amazon,
York University
Toronto, Ontario, Canada
stevenc@yorku.ca

I. Scott MacKenzie
Computer Science and Engineering
York University
Toronto, Ontario, Canada
mack@cse.yorku.ca

Mudit Misra
Human-Computer Interaction Group
University of California, Merced
Merced, California
mmisra@ucmerced.edu

Laxmi Pandey
Human-Computer Interaction Group
University of California, Merced
Merced, California
lpandey@ucmerced.edu

Ahmed Sabbir Arif
Human-Computer Interaction Group
University of California, Merced
Merced, California
asarif@ucmerced.edu

ABSTRACT

Touch is the predominant method of text entry on mobile devices. However, these devices also facilitate tilt-based input using accelerometers and gyroscopes. This paper presents the design and evaluation of *TiltWriter*, a non-touch, tilt-based text entry technique. *TiltWriter* aims to supplement conventional techniques when precise touch is not convenient or possible (e.g., the user lacks sufficient motor skills). Two keyboard layouts were designed and evaluated: Qwerty, and "Custom", a layout inspired by the telephone keypad. Novice participants in a longitudinal study achieved speeds of 12.1 wpm for Qwerty, 10.7 wpm for Custom. Error rate averaged 0.76% for Qwerty, 0.62% for Custom. A post-study extended session yielded 15.2 wpm with Custom, versus 11.5 wpm with Qwerty. Results and participant feedback suggest that a selection dwell time between 700 - 800 ms benefits accuracy.

CCS CONCEPTS

• **Information Interfaces and Presentation (e.g. HCI): User interfaces → Input devices and strategies**.

KEYWORDS

Text input, tilt-input, mobile device, handheld devices.

ACM Reference Format:
Steven J. Castellucci, I. Scott MacKenzie, Mudit Misra, Laxmi Pandey, and Ahmed Sabbir Arif. 2019. *TiltWriter*: Design and Evaluation of a No-touch Tilt-based Text Entry Method for Handheld Devices. In *MUM 2019: 18th International Conference on Mobile and Ubiquitous Multimedia (MUM 2019), November 26–29, 2019, Pisa, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3365610.3365629

MUM 2019, November 26–29, 2019, Pisa, Italy

ACM ISBN 978-1-4503-7624-2/19/11…$15.00
https://doi.org/10.1145/3365610.3365629

1 INTRODUCTION AND MOTIVATION

Mobile text entry is pervasive and typically involves repeated precise selections on a touchscreen. But how can users enter text when precise selection is not possible or there is no touchscreen present? The latter is exemplified in virtual reality (VR) applications, which are increasing in popularity. This paper presents *TiltWriter*, a text entry technique that uses input from the orientation sensors (accelerometers and gyroscopes) found in most mobile devices (e.g., smartphone, tablets) and other hand-held devices (e.g., game controllers, VR controllers). Using device tilt [23] to enter text solves many challenges with mobile device interaction: 1) Users might be too encumbered by carrying other items (e.g., a bag, drink, or umbrella) to precisely use the touchscreen while walking [2, 18]; 2) The touchscreen might not recognize selections if the user is wearing gloves [22]; 3) Typing and holding the device with one hand might cause accidental selection with the palm [25]; 4) The touchscreen might even recognize raindrops as selections [32]; and 5) Selecting small touchscreen keys might be too difficult for users with large fingers (i.e., the "fat-finger problem" [33]). Using tilt also provides an accessible text entry solution for older users [17] and users with fine motor challenges, such as hand tremors [21]. Additionally, no-touch with *TiltWriter* also makes it suitable in VR applications. Although there are numerous methods for entering text in VR [5], there is not yet a conventional one. In subsequent sections, we detail the design of *TiltWriter* and its evaluation in a longitudinal user study.

2 DESIGN

TiltWriter uses a device's tilt to move a tracking virtual ball over an onscreen keyboard. Moving the ball over a key highlights it. Keeping the ball over a key (i.e., dwelling) selects the key, as if the user had pressed or tapped it. Although dwelling incurs additional time to enter text, it is required to disambiguate the intended target and also gives users opportunity for corrective action, if necessary.

Figure 1 shows *TiltWriter* with a Qwerty keyboard. To aid text entry, *TiltWriter* provides word suggestions and completions in a list above the keyboard. Selecting a word from the list enters the word, followed by a space. Although the list only displays three words at once, selecting the "NEXT" key shows the next three words

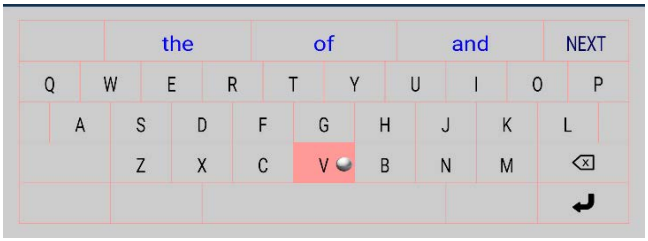

Figure 1: *TiltWriter* with a QWERTY layout. The tracking ball is visible, and the V-key is highlighted, but not yet selected.

in the list. The words in the completion list are derived from an English corpus using the previously entered letters, and are sorted in decreasing order of frequency within the corpus. If no input has yet occurred, the three most-frequent words in the corpus ("the", "of", and "and") appear.

The dwell duration for selection is user-configurable, and ranges from 300 to 1000 ms. To prevent accidental selections, there is a gutter, or rest area, around the perimeter of the keyboard, where no selections occur. Users can place the tracking ball in the gutter while they scan the keyboard for their next selection, search the completion list for the desired word, or pause while they compose text. The tracking ball is position-controlled, meaning that the device's tilt angle maps to a specific ball location on the screen [31, eq. 6-9]. Thus, holding the device at a constant angle keeps the tracking ball stationary at a specific location, and changing the device's tilt repositions the ball. This is counter to velocity-controlled tracking, where device angle controls the velocity of movement.

Typical text entry on mobile devices involves rapid, precise touch selections. But finger occlusion and the large size of a finger (relative to a key) often result in accidental selection of adjacent keys. To improve accuracy, many input methods use statistical decoding with a language model [7] to automatically select the most probable intended key. This technique was not implemented with *TiltWriter*, as the tracking ball eliminates the issue of both finger size and occlusion. Additionally, the key is highlighted before selection. The user can easily adjust the tilt before a key is selected.

As of now, *TiltWriter* does not facilitate input of upper-case letters, special characters, digits, or punctuation characters. However, such functionality could be easily implemented using available keys or submenus on the keyboard.

2.1 Return of the Ambiguous Keyboard

TiltWriter can include any selection-based keyboard. In the early days of mobile communications, users entered text messages using the 12-key phone keypad. With this layout, 26 letters are grouped with three or four letters on each numeric key (Figure 2, left). Such a mapping is "ambiguous", as a single key press could correspond to any of the letters on that key.

A popular technique called *T9* [19] allowed users to press one key for each letter in the desired word. It would then look-up the key sequence in a corpus to determine the corresponding word. If a "collision" occurred (i.e., multiple words matched the key sequence), the user cycles through candidate words by pressing a "next" key

Figure 2: Inspired by the T9 technique (left), *TiltWriter* also implements a Custom ambiguous keyboard (right).

(typically the *-key). The user then accepts the displayed word by entering a space (typically the 0-key).

TiltWriter implements a phone-like keypad as well as a "Custom" T9-like technique that maps letters to five keys, and displays up to five words in the completion list along the right of the keyboard (Figure 2, right). As with the QWERTY technique, users enter text by selecting the keys whose letters spell the desired word, and select the word from the completion list. Ambiguous keyboards rely on a corpus to disambiguate user input. Thus, the Custom keyboard can only enter words in its dictionary. A secondary, multi-tap mode could be implemented to enter non-dictionary words (and save them to the dictionary). In multi-tap, a user would select a key multiple times to indicate the desired character (e.g., once for "a", twice for "b", and thrice for "c"). Entering two characters from the same key would simply require the user to pause for a time-out duration between characters.

The Custom keyboard was designed with several issues in mind. With only five letter keys, the keys are larger and therefore require less precision for selection. This benefits encumbered users, or those with hand tremors or limited motor skills. Since there are fewer keys, the Custom keyboard occupies less space on the screen than the QWERTY layout. Thus, text entry should involve shorter distances between selections. The letter arrangement is alphabetic, reducing the visual scan time to find letters. Furthermore, the letter-keys are in a single column, simplifying the motor movements to navigate the letters. With these constraints, the division of the alphabet over five keys was driven by the desire to minimize the overall keystrokes for English text entry, as now described.

2.2 KSPC Analysis

Keystrokes per character (*KSPC*) is an established metric that captures the keystroke efficiency of a text entry method [13]. Of course, "keystrokes" is "tilt gestures" in the present context. For ambiguous keyboards, KSPC reflects the overhead in resolving the ambiguity when a "collision" occurs - a key sequence corresponding to more than one word. *TiltWriter* includes a built-in dictionary with about 10,000 words. With this, a phone keypad has *KSPC* = 1.0072 [11]. In other words, the overhead, on average, is just 7.2 keystrokes per 1000 keystrokes. The *KSPC* calculation is for T9-style input,

where the user navigates an ordered list of words when collisions occur. A similar calculation for the Custom keyboard yields *KSPC* = 1.0298, for an overhead of 29.8 keystrokes per 1000 keystrokes. This is less efficient than a phone keypad, but the keys are bigger - a clear tradeoff. However, the Custom keyboard includes a word completion list with five words; thus, the user has the opportunity to reduce keystrokes by selecting words early. The *KSPC* calculation for the Custom keyboard including word completion is *KSPC* = 0.6490. For a Qwerty keyboard, the calculations without and with word completion are *KSPC* = 1.0000 and *KSPC* = 0.4989, respectively. Since the Custom keyboard is ambiguous, it requires increased use of the "NEXT" key to search colliding and completed words in the completion list. However, little Next-key usage is required because colliding and completed words appear in groups of five. The longitudinal study detailed below aims to compare specific performance metrics between the Qwerty and Custom *TiltWriter* layouts.

3 RELATED WORK

Other text entry methods use device tilt as input, but most also require some touch-based selection. Users of *TiltType* [20] enter characters on a watch-sized device by combining tilt and button presses. *Unigesture* [24] facilitated ambiguous text entry using eight tilt directions, and button presses for corrections. It uses two custom layouts that arrange the letters and the space character in eight zones and thus requires precise target selection and tilting the device in all directions, which may not be feasible for individuals with hand tremors or limited motor skills. *TiltText* [35] uses phone tilt to disambiguate the intended character from keypad presses. *UniGest* [3] and *GesText* [10] created a gesture alphabet using vertical, horizontal, and rolling motions performed mid-air with a game controller. *SWiM* [36] uses a tilt-controlled tracker to draw on a shape writing [11] keyboard; a screen tap delineates input. *Swiftkey* [16] released a similar method as an April Fools' Easter Egg that enabled users to shape write, then dwell the tracking ball over a suggested word in the suggestion bar to enter the respective word. Little information is available about this method; however, a video demonstration [12] suggests that tracking used velocity-control rather than position-control.

Rotext [32] has the goal of sight-free text entry. It arranges characters in an arc layout, optimized for disambiguation. Users tilt a smartphone to indicate the approximate location of the desired character. Flicking and swinging motions confirm or correct input, and audio feedback speaks the disambiguated word. Finally, a study by Fitton et al. [6] investigates using tilt input while mobile. Teenage participants performed a tilt-controlled character selection task while walking the perimeter of a square 2.5 meters (8.2 feet) in length. They expressed interest in tilt-based input and willingness to use tilt for text entry. This method uses a position-control mapping like *TiltWriter*, but with a more physically demanding layout that arranges the letters in a 5×3 grid.

Work by Speicher et al. [28] evaluated multiple techniques for text entry in VR. Although four methods involved button presses for selection, two of them were touch-free: *Controller Tapping* (CT) and *Freehand* (FH). With CT, users hold hand-held controllers like pens and hunt-and-peck at a virtual keyboard. The FH technique is similar, but users use their fingers instead of controllers. Chen et al. [4] evaluated a different method that enables users to shape write [10] in VR. With this method, users use a controller like a laser pointer to draw shapes on a virtual keyboard. Pressing the "trigger" button of the controller starts drawing a shape, releasing it stops drawing and enters the predicted word.

4 EVALUATION

An experiment was conducted to evaluate the *TiltWriter* technique. In addition, the experiment compares the performance of a Qwerty layout to one with ambiguous input.

4.1 Participants

Ten participants were recruited from a local university campus. Three identified as female, and seven identified as male. Ages ranged from 19 to 37, with a mean of 24.8 (*SD* = 4.7). One participant was ambidextrous, but used his left hand as the dominant one during the study. Participants who completed the study received US $60 for their assistance.

Using the Interagency Language Roundtable (ILR) scale [9], four participants rated their English writing proficiency as Level 5 ("Functionally Native"), one as Level 4 ("Advanced Professional"), and five as Level 3 ("General Professional"). All participants had experience with smartphones, averaging 6.6 years (*SD* = 1.3). Experience with other touchscreen-based devices (e.g., tablets) averaged 4.9 years (*SD* = 1.4). Eight participants had 4.3 years (*SD* = 2.0) experience with tilt-based games. Regarding text entry, all participants had experience using predictive methods, averaging 5.9 years (*SD* = 2.3). Surprisingly, they also all had an average of 5.9 years (*SD* = 3.4) experience with ambiguous keyboards. Some participants used *T9* before transitioning to smartphones, while others used Pinyin keyboards to enter Chinese characters.

4.2 Apparatus

The interface for the study is shown in Figure 3. The software was written in Java for Android 5.1 or later.

The app implements the *TiltWriter* input method with the option for the Qwerty or Custom layout. It presents phrases, logs performance metrics, and allows configuration of study parameters, such as dwell time, keyboard layout, and the phrases file. Phrases were chosen randomly (without replacement) from a 500-phrase set [15]. They did not contain any numbers, punctuations, or uppercase letters. The corpus used for the word completion list was based on work by Silfverberg et al. [26]. It is a word-frequency list that represents the 9022 most frequent words in English [1], plus the words in the phrases file.

The experiment used a *OnePlus One* smartphone running Android 5.1.1 (Lollipop). The touchscreen measured 5.5 inches (139.7 mm) with a resolution of 1080×1920 pixels and a pixel density of 401 ppi. The device used an LIS3DH accelerometer [30] and an L3GD20 gyroscope [29]. The tracking ball used position-control with a sensor sampling rate of 50 Hz. Position control was used in combination with a nominal tilt gain setting of 50. With the device tilted about 25 to 40 degrees from flat, the ball was positioned at the side of the display for portrait or landscape orientation, respectively. With the device flat, the ball was in the center of the display. This is consistent with the control scheme described by Teather

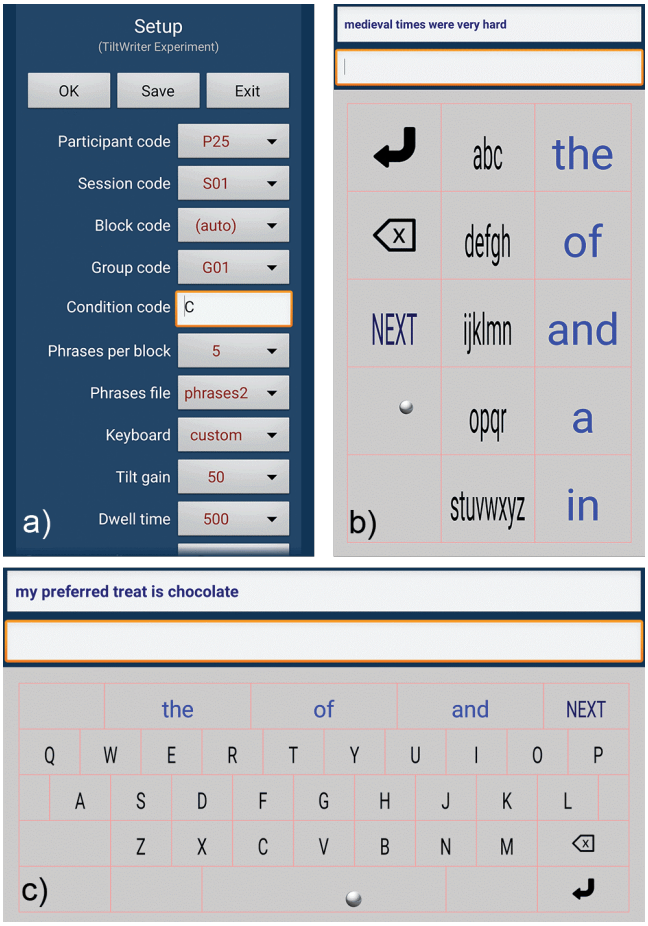

Figure 3: The *TiltWriter* interface facilitated parameter configuration (a), and input via the Custom (b) and Qwerty (c) layouts.

and MacKenzie in their tilt-based study [31, eq. 6-9]. Dwell-time selection was used, whereby the virtual ball was positioned and maintained inside a key for a predetermined duration (see Design below). In practice, *TiltWriter* could accommodate users with motor challenges by filtering the tilt sensor input, adjusting the tilt gain setting, or increasing the key size.

4.3 Procedure

In Session 1, the investigator demonstrated how to use each technique. He demonstrated how to select letters from the keyboard, how to select words from the completion list, and how to use the "NEXT" button on the Custom layout to see the next most probable words. Selecting ENTER (↵) signaled the completion of the current phrase and displayed the subsequent phrase. To make the text entry strategy more consistent between the two keyboards, the spacebar was disabled on the Qwerty keyboard. Thus, with either keyboard, words could only be entered by selecting them from the completion list. This approach encouraged participants to learn and leverage the idea and benefits of word completion. Following the demonstration, the participants practiced by entering five random phrases.

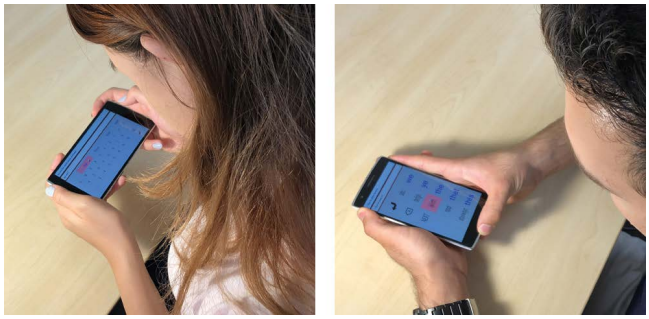

Figure 4: Participants entered text using Qwerty (left) and Custom (right) keyboards.

The Qwerty condition required participants to hold the device in landscape orientation, while the Custom condition required portrait orientation. The investigator demonstrated moving the tracking ball to the gutter area to avoid accidental selections while scanning for a letter. He then instructed participants to enter phrases "as fast and accurately as possible", and to "correct mistakes as they notice them". To accommodate participants' increase in expertise, the dwell time was decreased between each of the first five sessions. Further details are in the Design section below. During the study, display auto rotation, Wi-Fi, and phone functionality were disabled, and all non-essential apps were closed.

Participants scheduled sessions at their convenience, with two restrictions: sessions must be on different days, but with at most two days in between. As participants learned the techniques, the duration of sessions decreased. The testing time decreased from about 28 minutes in Session 1 to about 16 minutes in Session 10. Sessions took place in a quiet office environment, with participants seated comfortably at a desk. An LED lamp provided diffuse lighting and aimed to eliminate any glare on the device screen. Participants completed a demographics questionnaire at the start of Session 1 and consent forms before each session. A post-study questionnaire completed at the end of Session 10 combined NASA-TLX [8] questions with questions specific to the two input techniques.

4.4 Design

The study employed a 2 × 10 within-subjects design. The independent variables and levels were as follows:

- Keyboard: Qwerty (Q), Custom (C)
- Session: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

To offset learning effects of the keyboards, participants were assigned to one of two groups. Group 1 used Qwerty, followed by Custom, while Group 2 used the reverse order. Each session contained three blocks, with five phrases per block, for a total of 10 × 2 × 10 × 3 × 5 = 3000 phrases.

The dwell time for selection varied between sessions 1 to 10 as follows: 1000, 900, 800, 700, 600, 500, 500, 500, 500, and 500 ms. Although dwell time is a confounding variable, the initially higher values are meant to aid in learning the input methods, while the lower values are meant to benefit performance.

The dependent variables were entry speed, error rate, and efficiency. Entry speed was measured in words per minute (wpm), where a "word" is five characters, including spaces. Error rate was measured using the minimum string distance (MSD) metric [27, eq. 2]. Efficiency was measured using the keystrokes per character (KSPC) metric described earlier. Specifically, let $KSPC_{min}$ represent the minimum number of selections required to enter the presented phrase. This typically involves selecting words from the completion list as soon as they appear. Furthermore, let $KSPC_{phrase}$ represent the actual number of selections made to enter that presented phrase. Efficiency is thus the quotient $KSPC_{min}$ / $KSPC_{phrase}$, with perfect efficiency yielding a value of 1.0, and smaller values representing lower efficiency.

5 RESULTS AND DISCUSSION

Seventeen out of the 3000 phrases entered (i.e., 0.57%) had an error rate above 50%, and were omitted from further analysis. Presumably, participants selected ENTER prematurely, causing the exceptionally high error rate. Conversely, 94.0% of all transcribed phrases were error-free.

ANOVAs were conducted only on data from Session 6 to Session 10. Dwell time was constant at 500 ms during those sessions, thus eliminating any confounds. The group effect was not statistically significant for any of the dependent variables, indicating that counterbalancing worked.

5.1 Entry Speed

As shown in Figure 5, entry speed of both keyboards started at 6.4 wpm, but Qwerty surpassed Custom in Session 2. By Session 10, entry speed was 12.1 wpm for Qwerty and 10.7 wpm for Custom. The difference in entry speed between the two keyboards was always within 2 wpm, and the effect of keyboard was statistically significant ($F_{1,8}$ = 34.6, p < .0005). These results are accurately represented (R^2 > .97) by power law of learning models [14]. These models project entry speeds of 15 wpm and 13 wpm by Session 20 for Qwerty and Custom, respectively.

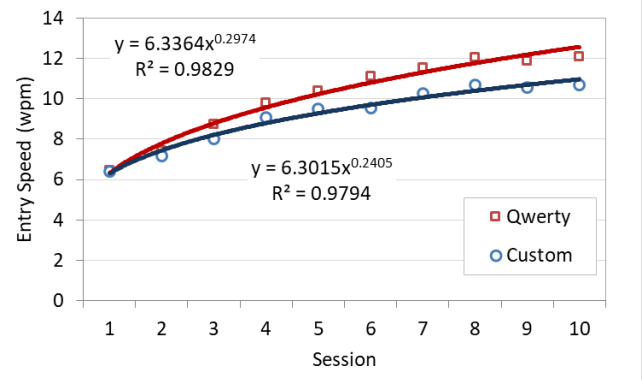

Figure 5: Qwerty entry speed surpassing Custom.

The ambiguous nature of the Custom keyboard required more selections to disambiguate and select completed words. $KSPC_{min}$ for the phrases in the Custom condition was 0.74 (SD = 0.02), versus 0.62 (SD = 0.02) for Qwerty. This increase in required selections could have contributed to the slower entry speed with the Custom keyboard.

In comparison to the techniques described in the Related Work section, *TiltWriter* fairs well in terms of entry speed. *GesText* entry speed started at 3.7 wpm and rose to 5.3 wpm over four sessions of 15 phrases [9], while *TiltText* grew from 7.4 wpm to 13.6 wpm over 16 blocks of four phrases [35]. *Rotext* had about 10 wpm at the end of five 20-minute sighted sessions [34]. *SWiM* participants achieved 15.5 wpm after three blocks of 10 phrases [34]. The VR techniques *CT* and *FH* averaged 12.7 wpm and 9.8 wpm, respectively [26].

5.2 Error Rate

The effect of keyboard on error rate was not significant ($F_{1,8}$ = 1.43, p > .05). Qwerty averaged 0.76% (SD = 1.27) and Custom averaged 0.62% (SD = 0.95). Qwerty error rates were higher than those of Custom during the first and last sessions, but they often crisscrossed each other throughout the study. The error rates are rather erratic (Figure 6), which is not unusual for word-based text entry methods. When transcription errors did occur, participants misspelled entire words due to an incorrect selection from the completion list, or omitted them entirely. Consequently, those phrases would have rather high error rates. In comparison, the *SWiM* error rate was 0.88% [34], while *Rotext* had an impressive 0.01% error rate in the sighted sessions [32]. Error rates for VR techniques *CT* and *FH* were high, at 1.2% and 7.6%, respectively [26]. The other techniques presented in the Related Work section did not involve any quantitative empirical evaluation of error rate, or used a non-equivalent metric.

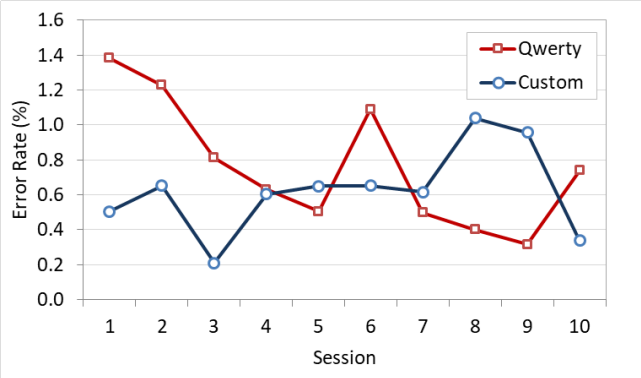

Figure 6: Error rates were erratic, which is not unusual.

The MSD error rate metric only reveals uncorrected errors in the transcribed phrase. Investigating corrective actions, in the form of backspace use, yields additional insight. Keyboard had a significant effect on backspace use ($F_{1,8}$ = 30.9, p < .0005), with the Custom condition exceeding Qwerty. Figure 7 illustrates consistently fewer corrections with the Qwerty keyboard, suggesting that participants opted to leave errors uncorrected. With the Custom keyboard, participants committed and corrected more errors. They tended to correct more errors in Session 5 to Session 7, but left more errors and incurred higher error rates in Session 8 to Session 10

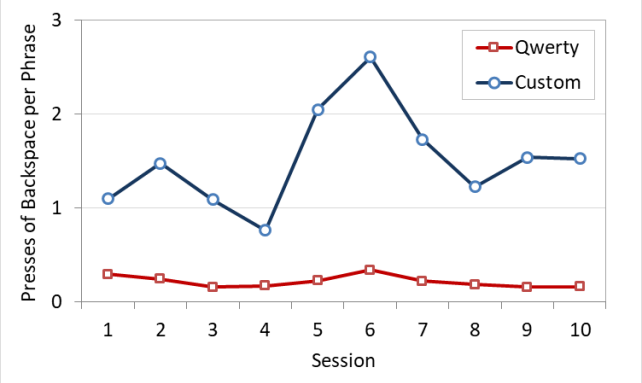

Figure 7: Participants corrected more with the Custom keyboard versus the QWERTY keyboard.

5.3 Efficiency

The effect of keyboard on efficiency was statistically significant ($F_{1,8}$ = 109.4, p < .0001). QWERTY efficiency remained steady around .92, while Custom efficiency fluctuated from .72 to .85 (Figure 8). Efficiency for both keyboards peaked early in the study: Session 3 for QWERTY, and Session 4 for Custom. These sessions correspond to dwell times of 800 ms and 700 ms, respectively. Both keyboards also experienced a dip in efficiency and a spike in backspace use during Session 6 (Figure 7), when dwell time decreased to 500 ms. For QWERTY, this also corresponds to a spike in error rate (Figure 6). Perhaps a dwell time in the 700 to 800 ms range allowed participants to scan for the next selection trajectory before moving the tracking ball.

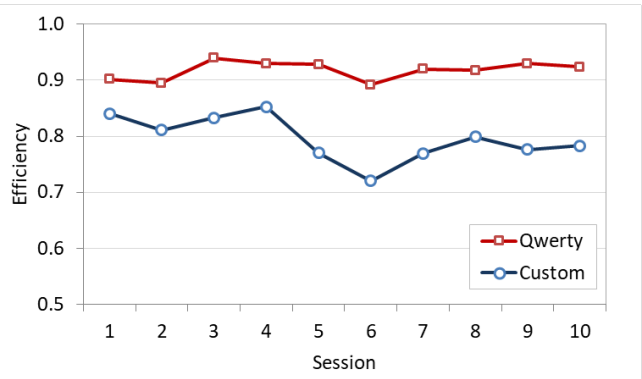

Figure 8: Efficiency by keyboard and session. Both keyboards experienced a dip in efficiency when dwell time dropped to 500 ms in Session 6.

A future study could investigate this further. The efficiency decrease with a shorter dwell time suggests that participants hesitated on a key and accidentally selected it twice. Participants corrected the mistake when using the Custom keyboard (hence, the increased backspace use), but left the error when using QWERTY (thus, the high error rate). Although participants' efficiency with the lower dwell time did increase by Session 10, they failed to reach the previous highs.

5.4 Extended Session

Entry speed results in Session 10 suggested that participants were still learning and improving their performance. An extended session was added to investigate *TiltWriter* performance with experienced users. The same apparatus was used, configured with a 500 ms dwell time. Six of the original 10 participants were recruited and paid an additional US $10. Half used the QWERTY keyboard for the extended session, while the other half used the Custom keyboard. The session involved five blocks of entering "the quick brown fox jumps over the lazy dog" 10 times, for a total of 6 × 5 × 10 = 300 additional phrases. However, four phrases were omitted from further analysis: two had an error rate above 50%; two exhibited long, unexplained pauses during input.

Entry speed for the first phrase was 9.6 wpm for QWERTY and 9.3 wpm for Custom. After the third phrase, Custom surpassed QWERTY, finished at 15.2 wpm, and averaged 13.6 wpm (SD = 2.1) overall. QWERTY finished at 11.5 wpm and remained relatively steady, averaging only 11.1 wpm (SD = 0.9). However, this difference was not statistically significant ($F_{1,4}$ = 4.91, p > .05). Delving deeper revealed that keyboard did have a significant effect on keystrokes (i.e., selections) per second ($F_{1,4}$ = 23.21, p < .01). As shown in Figure 9, keystrokes per second improved moderately with QWERTY, but saw an early surge with Custom.

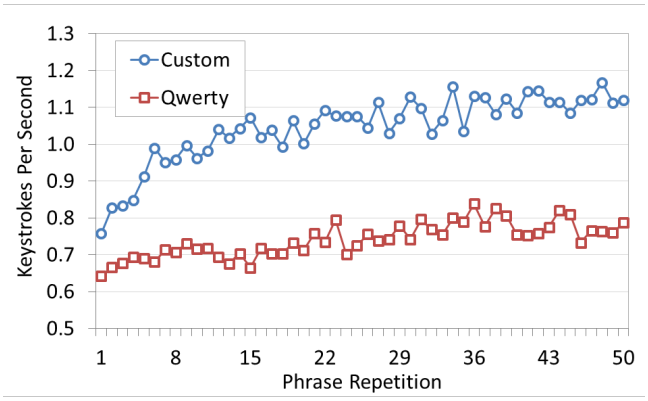

Figure 9: The rate of key selections per second.

Participants using the Custom keyboard consistently averaged more than one keystroke per second beyond the 19^{th} repetition. This is likely due the Custom keyboard having fewer keys to visually scan, a smaller area to traverse for selections, and requiring traversal in mainly one direction (i.e., vertical). But since selection speed relates to entry speed, why is there discrepancy in significance between keystrokes per second and entry speed? Entry speed is based on the transcribed phrase, while keystrokes per second is based on raw input. Those inputs could be corrective (e.g., selecting the backspace key) or inefficient (e.g., not selecting a word when it first appears in the completion list). Repeatedly entering the same phrase allowed participants to anticipate words appearing in the completion list and to better optimize their actions. The Custom

keyboard saw more use of the completion list compared to previous sessions, reaching an efficiency of .93. There was no change in efficiency with the QWERTY keyboard, which remained at .92. The difference in efficiency between keyboards was not statistically significant ($F_{1,4}$ = 0.04, ns). As with the previous sessions, error rates were erratic and keyboard layout did not have a significant effect ($F_{1,4}$ = 1.03, p > .05). Error rate in the Custom keyboard group decreased to 0.34%, but the QWERTY group saw a threefold increase to 2.83%; they likely ignored errors to improve entry speed. Further investigation showed the opposite with Custom participants; they seemed to correct more errors. Custom participants averaged 0.59 (SD = 0.49) backspaces per phrase, while QWERTY participants averaged only 0.18 (SD = 0.50). Like error rate, this difference was not statistically significant ($F_{1,4}$ = 3.63, p > .05). However, both metrics could explain why the significance (i.e., difference in the means) of keystrokes per second was attenuated in entry speed.

5.5 Participant Feedback

Participants found both *TiltWriter* keyboards to be quickly learnable and easy to use. However, they described the tilting technique as too cumbersome to use frequently, given the speed and simplicity of the default keyboards on mobile devices. This is understandable. *TiltWriter* is not meant to replace the ubiquitous, touch-based QWERTY keyboard; it was designed to provide an alternative input method when operating a touchscreen is not convenient or possible. One participant commented on the dwell time setting of the Custom keyboard: "[S]low down the quickness of the selection. Sometimes I couldn't move fast enough and would take time to delete an incorrect selection." As mentioned previously, a dwell time in the 700 to 800 ms range benefitted efficiency more than the 500 ms used in the latter half of the study. It allowed users to better plan their next selection, which resulted in fewer errors. Based on the NASA-TLX responses (Figure 10), participants achieved a high level of performance (characterized by a low score) with moderate effort and little frustration. Mental demand was slightly elevated for both keyboards. Participants reported that Custom was more taxing than QWERTY in the mental and effort workloads. However, the effect of keyboard was not statistically significant for any of the workload metrics (p > .05). The analysis was conducted using the Wilcoxon signed-rank test, due to the non-parametric data and study design [14].

6 CONCLUSION

TiltWriter provides an alternative (not a replacement) to touch-based techniques for text entry. Participants found both the QWERTY and Custom keyboards easy to use. After 10 sessions, entry speeds reached 12.1 wpm for QWERTY and 10.7 wpm for Custom. Error rates averaged 0.76% for QWERTY and 0.62% for Custom. Participants corrected more with Custom, but left more transcription errors with QWERTY. Although efficiency was better with QWERTY than with Custom, both keyboards peaked during early sessions with higher dwell times.

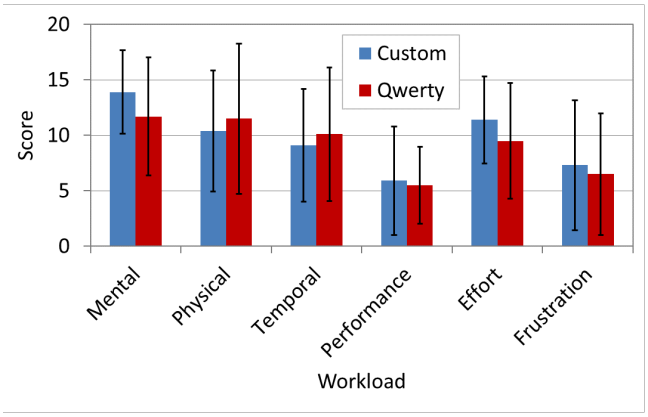

Figure 10: Participant feedback using NASA-TLX workload scores. Error bars represent ±1SD.

7 FUTURE WORK

Future work is numerous and multi-disciplinary: identifying the ideal dwell time for experienced users; evaluating *TiltWriter* without a predictive system; comparing *TiltWriter* to tapping and shape writing techniques with older adults and those with fine motor challenges; and evaluating *TiltWriter* in a VR environment with a hand-held device (possibly for both text entry and remote pointing).

REFERENCES

[1] Oxford Text Archive. [n. d.]. British National Corpus (World edition). Retrieved May 20, 2001 from http://www.natcorp.ox.ac.uk
[2] Ahmed Sabbir Arif. 2012. A Survey on Mobile Text Entry Handedness: How Do Users Input Text on Handheld Devices While Nomadic?. In *2012 4th International Conference on Intelligent Human Computer Interaction (IHCI)*. IEEE, 1–6.
[3] Steven J. Castellucci and I. Scott MacKenzie. 2008. Unigest: Text Entry Using Three Degrees of Motion. In *CHI'08 Extended Abstracts on Human Factors in Computing Systems*. ACM, 3549–3554.
[4] Sibo Chen, Junce Wang, Santiago Guerra, Neha Mittal, and Soravis Prakkamakul. 2019. Exploring Word-gesture Text Entry Techniques in Virtual Reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, LBW0233.
[5] Tafadzwa Joseph Dube and Ahmed Sabbir Arif. 2019. Text Entry in Virtual Reality: A Comprehensive Review of the Literature. In *Human-Computer Interaction. Recognition and Interaction Technologies*, Masaaki Kurosu (Ed.). Springer International Publishing, Cham, 419–437.
[6] Daniel Fitton, I. Scott MacKenzie, Janet C. Read, and Matthew Horton. 2013. Exploring Tilt-Based Text Input for Mobile Devices with Teenagers. In *Proceedings of the 27th International BCS Human Computer Interaction Conference*. British Computer Society, 25.
[7] 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*. ACM, 194–195.
[8] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Advances in psychology*. Vol. 52. Elsevier, 139–183.
[9] Interagency Language Roundtable (ILR). [n. d.]. Interagency Language Roundtable Language Skill Level Descriptions - Writing. Retrieved October 4, 2018 from http://www.govtilr.org/Skills/ILRscale5.htm
[10] Eleanor Jones, Jason Alexander, Andreas Andreou, Pourang Irani, and Sriram Subramanian. 2010. GesText: Accelerometer-Based Gestural Text-Entry Systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2173–2182.
[11] Per Ola Kristensson. 2007. *Discrete and Continuous Shape Writing for Text Entry and Control*. Ph.D. Dissertation. Institutionen för datavetenskap.
[12] Droid Life. 2013. Swiftkey Tilt is a Real Keyboard, Check It Out. Video. Retrieved October 9, 2019 from https://youtu.be/3rscERWxc-M?t=53
[13] I. Scott MacKenzie. 2002. KSPC (keystrokes per character) as a Characteristic of Text Entry Techniques. In *International Conference on Mobile Human-Computer Interaction*. Springer, 195–210.

[14] I. Scott MacKenzie. 2012. *Human-Computer Interaction: An Empirical Research Perspective*. Newnes.

[15] 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*. ACM, 754–755.

[16] Microsoft. 2016. Download SwiftKey the Smart keyboard and Get More Done. Retrieved October 9, 2019 from https://www.microsoft.com/en-us/swiftkey

[17] Lilian Genaro Motti, Nadine Vigouroux, and Philippe Gorce. 2013. Interaction Techniques for Older Adults Using Touchscreen Devices: a Literature Review. In *Proceedings of the 25th Conference on l'Interaction Homme-Machine*. ACM, 125.

[18] Alexander Ng, Stephen A Brewster, and John H Williamson. 2014. Investigating the Effects of Encumbrance on One-and Two-Handed Interactions with Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1981–1990.

[19] Nuance. [n. d.]. T9 Text Input: The Global Standard for Mobile Text Input. Retrieved August 1, 2018 from https://www.nuance.com/mobile/mobile-solutions/text-input-solutions/t9.html#tab3

[20] Kurt Partridge, Saurav Chatterjee, Vibha Sazawal, Gaetano Borriello, and Roy Want. 2002. TiltType: Accelerometer-Supported Text Entry for Very Small Devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*. ACM, 201–204.

[21] Katrin Plaumann, Milos Babic, Tobias Drey, Witali Hepting, Daniel Stooß, and Enrico Rukzio. 2016. Towards Improving Touchscreen Input Speed and Accuracy on Smartphones for Tremor Affected Persons. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*. ACM, 357–360.

[22] Ondrej Polacek, Thomas Grill, and Manfred Tscheligi. 2013. NoseTapping: What Else Can You Do With Your Nose?. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 32.

[23] Jun Rekimoto. 1996. Tilting Operations for Small Screen Interfaces (Tech Note). In *UIST*, Vol. 96. 167–168.

[24] Vibha Sazawal, Roy Want, and Gaetano Borriello. 2002. The Unigesture Approach One-Handed Text Entry for Small Devices. In *International Conference on Mobile Human-Computer Interaction*. Springer, 256–270.

[25] Julia Schwarz, Robert Xiao, Jennifer Mankoff, Scott E. Hudson, and Chris Harrison. 2014. Probabilistic Palm Rejection using Spatiotemporal Touch Features and Iterative Classification. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009–2012.

[26] Miika Silfverberg, I. Scott MacKenzie, and Panu Korhonen. 2000. Predicting Text Entry Speed on Mobile Phones. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 9–16.

[27] R William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and A New Unified Error Metric. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 113–120.

[28] Marco Speicher, Anna Maria Feit, Pascal Ziegler, and Antonio Krüger. 2018. Selection-Based Text Entry in Virtual Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 647.

[29] STMicroelectronics. [n. d.]. L3GD20 - Low Power 3-axis Gyroscope. Retrieved August 1, 2018 from https://www.st.com/en/mems-and-sensors/l3gd20.html

[30] STMicroelectronics. [n. d.]. LIS3DH - 3-axis MEMS Accelerometer. Retrieved August 1, 2018 from https://www.st.com/en/mems-and-sensors/lis3dh.html

[31] Robert J Teather and I Scott MacKenzie. 2014. Position vs. Velocity Control for Tilt-based Interaction. In *Proceedings of Graphics Interface 2014*. Canadian Information Processing Society, 51–58.

[32] Ying-Chao Tung, Mayank Goel, Isaac Zinda, and Jacob O Wobbrock. 2018. RainCheck: Overcoming Capacitive Interference Caused by Rainwater on Smartphones. In *Proceedings of the 2018 on International Conference on Multimodal Interaction*. ACM, 464–471.

[33] Daniel Vogel and Patrick Baudisch. 2007. Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 657–666.

[34] William S Walmsley, W Xavier Snelgrove, and Khai N Truong. 2014. Disambiguation of Imprecise Input with One-dimensional Rotational Text Entry. *ACM Transactions on Computer-Human Interaction (TOCHI)* 21, 1 (2014), 4.

[35] Daniel Wigdor and Ravin Balakrishnan. 2003. TiltText: Using Tilt for Text Input to Mobile Phones. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*. ACM, 81–90.

[36] Hui-Shyong Yeo, Xiao-Shen Phang, Steven J Castellucci, Per Ola Kristensson, and Aaron Quigley. 2017. Investigating Tilt-based Gesture Keyboard Entry for Single-Handed Text Entry on Large Devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 4194–4202.