Typing on Glasses: Adapting Text Entry to Smart Eyewear

Tovi Grossman[1], Xiang 'Anthony' Chen[2], George Fitzmaurice[1]

[1]Autodesk Research
{firstname.lastname}@autodesk.com

[2]Carnegie Mellon University
xiangchen@acm.org

ABSTRACT

Text entry for smart eyewear is generally limited to speech-based input due to constraints of the input channels. However, many smart eyewear devices are now including a side touchpad making gesture-based text entry feasible. The Swipeboard technique, recently proposed for ultra-small touch screens such as smart watches, may be particularly suitable for smart eyewear: unlike other recent text-entry techniques for small devices, it supports eyes-free input. We investigate the limitations and feasibility of implementing Swipeboard on smart eyewear, using the side touch pad for input. Our first study reveals usability and recognition problems of using the side touch pad to perform the required gestures. To address these problems, we propose SwipeZone, which replaces diagonal gestures with zone-specific swipes. In a text entry study, we show that our redesign achieved a WPM rate of 8.73, 15.2% higher than Swipeboard, with a statistically significant improvement in the last half of the study blocks.

INTRODUCTION

Smart eyewear and heads up displays enable always-available access to information, yet only provides limited interaction possibilities due to its small and wearable form, making tasks like text entry extremely hard. One solution is to use a supplementary device, such as the Twiddler [16]. However, having to carry or be tethered to an additional device defeats the purpose of smart eyewear, which is meant to provide unobtrusive and immediate access to information. Vision-based techniques have also been explored [15]; yet they remain a suboptimal as such techniques are often computationally expensive and could potentially be error-prone due to the uncertainty of the environment. Voice input may be useful in certain situations, but when surrounded by peers it might incur privacy concerns and be socially unacceptable [25, 26].

MobileHCI '15, August 24 - 27, 2015, Copenhagen, Denmark
 ISBN 978-1-4503-3652-9/15/08…$15.00
DOI: http://dx.doi.org/10.1145/2785830.2785867

One promising method to provide text input on smart eyewear devices, that has yet to be explored, is to use the device's built in touch input. Specifically, a number of smart eyewear devices are now including a touchpad for gesture-based input on the side of the glasses (e.g. Google Glass, SiME Smart Glasses, Recon Jet, Optivent Ora). However, given their limited input space of these touchpads, text entry may still be difficult.

Recent research on Smart Watch interaction has helped identify a number of promising techniques for text-entry which require very small input real estate. These include ZoomBoard [22], SplitBoard [9], Callout [13], and ZShift [13]. However, these techniques all require direct target acquisition on the display surface. In contrast, Swipeboard is a watch-based technique that specifies each character using only two directional gestures: the first selects a subgroup of keys and the second specifies a key within the subgroup [7]. Such gesture-based methods offer a subtle, effective and low-cost text entry solution. More importantly, the technique is target-agnostic supporting eyes-free input, making it a promising candidate for use on the side touch pad of smart eyewear. However, this technique has yet to be implemented, deployed or tested on smart eyewear.

In this paper we investigate the limitations and feasibility of adapting a gesture-based text entry technique (Swipeboard) to smart eyewear. In our initial study we find that the unique configuration of a smart eyewear touch pad makes it difficult to distinguish between the 8 atomic directional gestures, and that diagonal swipes take longer to perform. To address this issue, we proposed a new design called SwipeZone. SwipeZone takes advantage of the relatively wider dimension on the side touch pad and divides it into three zones. The diagonal gestures of Swipeboard are replaced with zone-specific vertical gestures (Figure 1).

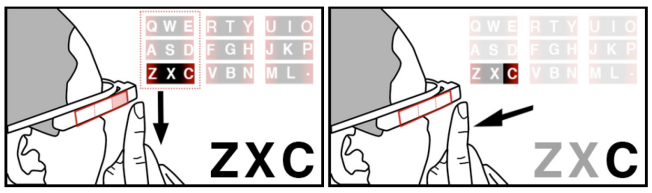

Figure 1. SwipeZone divides the width of a smart eyewear side touch pad into three zones, swiping down on the front zone selects the lower-left group 'ZXC', then swiping to the back selects 'C' – the rightmost key in the group.

Our second study tests both Swipeboard and SwipeZone on a smart eyewear unit using a text entry task. The results show that gesture-based text entry is feasible on a smart eyewear touch pad. The SwipeZone redesign outperforms the original Swipeboard technique (15.2% faster), however, only reaches statistical significance in the final half of the study. Overall, our main contributions are the first smart eyewear text entry techniques that utilize the built-in touch-sensitive input area, and two studies that investigate the feasibility of the techniques. While these contributions are based on studies using a Google Glass device, the results generalize to a variety of smart eyewear devices that our now being built with embedded touch pads.

RELATED WORK

To understand the challenges of smart eyewear's small and wearable form factors, we review text entry research for both *devices with very small input areas* and *head-mounted displays*. We also review *gesture-based text entry techniques* to inform our design on smart eyewear.

Text Entry Techniques for Small Form Factors

Past work has explored various text entry techniques for devices with very small form factors. Earlier work proposed the use of a miniaturized wearable keyboard [20]. More recent work propose the use of motion sensors, e.g., using tilting to specify characters for text entry [23, 28]. MultiWidget uses a dialing gesture along the watch's edges to specify a numeric value [4].

Recent work has also looked at the use of tiny QWERTY soft keyboards for entering text on small devices [13]. Such techniques have leveraged zooming [22], callouts [13], and swiping gestures [7, 9]. While not necessarily meant for small displays, Arif et al. combine a QWERTY keyboard with directional strokes to enter special keys [2].

Another approach is using alternate key mapping, such as chording, key selections or gestural shortcuts. For example, Wigdor and Balakrishnan add three chording keys to speed up typing on a numeric phone keypad [29]. Mackenzie demonstrates the use of three keys to enable selection-based text entry [17]. MessageEase places characters on a 3x3 matrix of keys, and characters are entered using a two-keystroke scheme [21]. The 1line keyboard incorporates touch into key selection, reducing the keyboard into one line of keys [14].

Text Entry Techniques for Head-Mounted Displays

Head-mounted displays have been used to create immersive or mobile experiences with digital information. However, their immersive and mobile nature also prevents the user from accessing regular input devices to perform text entry tasks. To solve this problem, researchers have experimented with various customized input devices. Twiddler is a one-handed chording keyboard that allows for eyes-free text entry for mobile or wearable devices [16]. This technique was compared to Miniature QWERTY keyboards in follow up-work [8].

The chording glove [27] and Huffman Base-4 Text Entry Glove [3] embed buttons into the glove and users can specify characters directly using hand gestures. Liu et al. propose a vision-based mechanism that uses a head-mounted camera to recognize mid-air handwriting [15].

Other researchers focused on using a wrist-worn input device for text entry, which is also a potential solution for typing while wearing a head-mounted display. For example, Airwriting employs inertial sensors and machine learning techniques to recognize handwriting based on the hand's motion [1]. One-key keyboard augments one single key to sense a user's fingertip position, thus allowing typing on a full QWERTY keyboard [11].

While all this work has demonstrated various possible text entry solutions for head-mounted display, they often require additional input devices, or rely on vision-based recognition that is potentially high-cost and error-prone. Minuum has proposed the use of the 1line keyboard for Google Glass in a proof-of-concept video[1], but the technique has yet to be implemented or evaluated. Our goal is to enable a text entry mechanism that is lightweight and self-contained within modern head-mounted displays, such as the Google Glass.

Gesture-based Text Entry Techniques

Gestural shortcuts can also be effective with small form factors. Chen et al. [7] summarized two ways of encoding characters with gestures: a continuous approach maps a word or a character to a continuous stroke (e.g., MDITIM [10], Graffiti [6], EdgeWrite [31], Shark[2] [12]), and a discrete approach maps each character to a number of symbolic tokens (e.g., H4-Writer uses a base-4 encoding [19] and Quikwriting uses base-9 [24]).

For example, EdgeWrite uses stylus-based gestures guided by the physical edges of a device, thus making it easier and faster to perform [31]. In contrast, Swipeboard specifies a character by two swipes: novices learn the gesture by swiping to locate a specific key while experts gradually learn and memorize the swipe combination for each character [7].

Given their limited input real-estate, discrete techniques, such as Swipeboard [7], may be preferable for smart eyewear devices. Furthermore, gesture-based input like Swipeboard is 'target-agnostic' [30], making it promising for smart eyewear, as absolute touch coordinates are never required. We are unaware of any implementations or evaluations of such techniques for smart eyewear in the research literature.

In summary, our review of the literature indicates that there are no existing gesture based text entry techniques for smart eyewear, and that of the existing techniques for small form factors, SwipeBoard holds particular promise. As such, we focus the remainder of the paper on adapting SwipeBoard to a smart eyewear form factor.

[1] http://minuum.com/google-glass-keyboard/

ADAPTING SWIPEBOARD TO SMART EYEWEAR

To adapt the gesture-based Swipeboard technique to smart eyewear, we first review its design and performance model. We then describe our implementation and initial experience of using it on a smart eyewear device.

Reviewing Swipeboard: How It Works

Swipeboard is a recently developed text entry technique that encodes each alphabetic character into a series of two touch actions. The technique utilizes the traditional QWERTY keyboard layout, to allow users to leverage their existing spatial memory of character locations.

The keyboard is divided into nine regions (Figure 2a), each containing 3 (e.g., 'ASD') or 4 (e.g., 'RTYU') characters. The first touch action is used to select the desired region. The swipe is in one of 8 directions (e.g. swiping left for "ASD" or up and to the right for "IOP".) A tap is used to select the middle region ("FGH"). Once a region is selected, a zoomed in view is displayed ((Figure 2b).

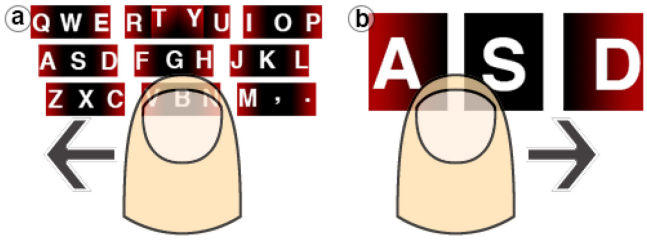

Figure 2. The original Swipeboard technique. a) The first swipe specifies one of the nine regions subdivided from a QWERTY keyboard. b) The second swipe specifies the character, in this example, 'D'. Figure from Chen et al. [7].

The second action is used to select the desired character within a region. Swiping left selects the left character, tapping selects the center character, and swiping right selects the right character. In the one case of four characters ('RTYU'), swiping left selects 'R', up-left selects 'T', up-right selects 'Y' and right selects 'U'. The user can swipe down to cancel the selected region and return to the first level keyboard view.

Additional gestures are used for other functions. A double-swipe down-left deletes a character, and double-swipe down-right enters a space, and a double-swipe up switches to a symbol and number keyboard.

An important property of Swipeboard is that it is target agnostic: the actions can occur anywhere on the display, and no spatial target selection is required. This makes it particularly appropriate for smart eyewear, since users cannot see where their finger lands on its side touchpad.

Performance Model

To understand the limits of its performance, a four-stage performance model is used to describe the execution time (T) for individual characters using the Swipeboard technique. Each character consists of two input events. Each input event consists of a planning phase, when the finger is up (T_{u1}, T_{u2}), and an action phase, when the finger is down (T_{a1}, T_{a2}). Thus, the completion time for a character is as follows:

$$T = T_{u1} + T_{a1} + T_{u2} + T_{a2}$$

Chen et al. [7] used previous performance model data to estimate that in optimal conditions an expert could perform at 464ms per character, or 25.87 WPM. Later, we relate our study results to this performance model.

Adaptation to Smart Eyewear

In our initial implementation, we used a Google Glass unit and developed a standalone Android application using the ADT plugin for the Eclipse IDE. The implementation reproduced the Swipeboard technique exactly. The X and Y coordinates were obtained through the Android *MotionEvent* object that the side touch pad produces.

There were some issues with this initial implementation which seemed to limit the efficiency of the technique. First, taps, which were defined by a travel distance of less than 10 pixels, were sometimes being recognized as swipes. This is due to the higher pixel density of the input device. Second, and more importantly, the diagonal swipes, which were detected at 45° angles, were difficult to perform. The short and wide nature of the touchpad afforded wider diagonal gestures at an angle much less than 45°.

To better understand and address these issues, we performed an experiment to investigate how accurately users could perform the atomic gestures required for the Swipeboard technique, on the Google Glass touch pad.

STUDY 1 – ATOMIC GESTURES

The Swipeboard technique uses 8 directional gestures and a tap as the building blocks for text entry. The technique was previously validated on a watch-sized input area (12mm by 12mm). However, the dimensions of the touch pad on smart eyewear may be quite different. For example, with Google Glass, the touch pad is 76.2mm by 10.4mm, with a resolution of 1366x187. It is unclear how well Swipeboard's atomic gestures can be performed and distinguished on touch pads with this unique form factor.

Apparatus

The study was performed using a Google Glass unit. The device was tethered to a laptop via a USB cable so that the screen output could be monitored by the experimenter.

Participants

We recruited 10 participants (1 female, 9 male), with an average age of 33.2. Nine of the participants were right handed and all participants used their right hand for input, as this is the side that has the touchpad on the Google Glass device. The participants were recruited from our institution and were not compensated. None of the participants had extensive experience with smart eyewear.

Design

A repeated measures within-participant design was used. The independent variables were *Direction (N, NE, E, SE, S, SW, W, NW, TAP*) and *Block* (1-8). Participants performed

the study in one session lasting approximately 10 minutes. The session was broken up into 8 blocks. Each block consisted of 36 trials, with each gesture appearing four times, in randomized order. This resulted in a total of 288 trials per participant, and a total of 2880 data points overall.

Procedure

Participants sat in a chair facing a black background, which allowed them to clearly see the content on the Google Glass display. The background was approximately 2 feet away from the user's head position. We allowed users to rest their elbows on the chair armrest to prevent fatigue.

The trial started by displaying the gesture direction. An arrow inside a square was used for the eight directional strokes, and a dot in the middle of the square was used for *TAP* (Figure 3a). The user then performed the associated gesture by swiping on the side touch pad. We mapped the right side of the display to the back of the Google Glass touch pad, so a swipe from front to back would perform the EAST stroke (Figure 3b). This mapping seemed more intuitive to users during pilot testing, and is the default Google Glass mapping.

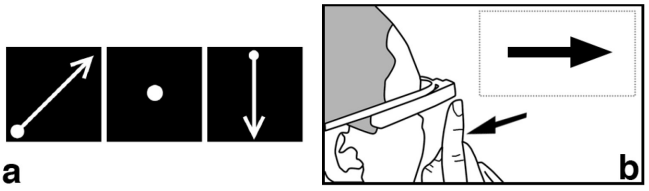

Figure 3. a) Exemplar visual stimulus (NE, TAP, S); b) illustration of the mapping of an EAST stroke.

The start and end coordinates of each touch event were logged, as was the time between the two events. There were no "errors" for this study, since the purpose was to measure users' accuracy of performing the gestures. However, if either the X or Y component of the stroke was in the opposite direction of the correct input, the input was ignored and the user was prompted to try again. Such cases indicated a mistaken interpretation of the desired gesture, not an inaccuracy in performing it. When a gesture was entered, the next trial was immediately displayed.

Results and Analysis

Analyzing and Visualizing Swiping Gestures

We illustrate the end points for each atomic swipe/tap gesture using a scatterplot in Figure 4. Each gesture is color-coded. The scatterplot shows the X coordinates of the horizontal and diagonal gestures are 'stretched', due to the wide form factor of the touchpad. Figure 5a shows the normalized vectors computed from the original touch event data. There is some degree of overlap between adjacent gestures, which is further illustrated in Figure 5b: it shows the possible ranges of each swipe, computed from their mean ±3 standard deviations. It shows that while the directionalities of the horizontal swipes (E and W) are fairly uniform, the vertical and diagonal swipes, however, are widely distributed and overlap with each other.

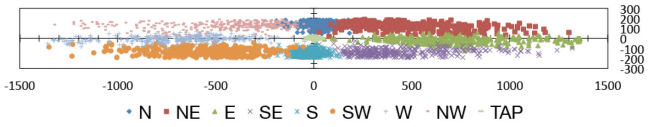

Figure 4. Scatterplot of each gesture's end points in Study 1.

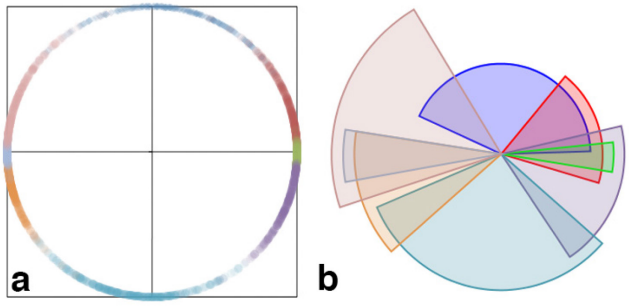

Figure 5. Directionalities of the atomic gestures from Study 1: scatterplot of the normalized vectors computed from each gesture (a); possible ranges of each swipe, computed from their mean ± 3× standard deviation (b).

Building Models to Recognize Swiping Directions

The above data indicates that using the vector angles to determine a stroke's direction may not be effective. Instead, we use the Cartesian coordinates (corresponding to the end points of the swipes). As shown in Figure 6a, we can define 8 quadrants by a δ_x and δ_y parameter. For example, a stroke with $X > \delta_x$ and $Y > \delta_y$ would be classified as *NE*.

We use an iterative search to find the optimal parameters to determine swipe direction. The first step in our recognition is to classify a tap, which is defined by touch points whose distance to the origins is smaller than a certain threshold. We use a bounding box with dimensions tap_x and tap_y. We then distinguish between the different swipes by finding appropriate values of δ_x and δ_y.

To find the optimal parameter values we perform a naïve stepwise iterative search across all possible combinations of the parameter pairs, first (tap_x, tap_y), and then ($\boldsymbol{\delta_x}$ and $\boldsymbol{\delta_y}$). We iterate with steps of 1px. We calculated the optimal parameters and resulting accuracy across the entire data set. The optimal values for recognizing a *TAP* (tap_x = 53, tap_y = 42) resulting in an accuracy of 99.97%. The optimal values of δ_x and δ_y were 136 and 59, with an accuracy of 94.0%. This method would result in a 6% error rate.

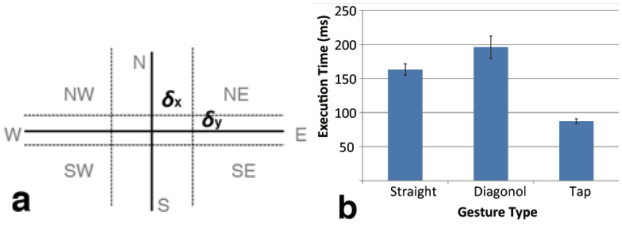

Figure 6. a) The gestures can be determined based on coordinates, δ_x and δ_y, which groups the end points of each swipe into eight swiping directions. b) Execution time by gesture shows diagonal swipes took significantly longer. Error bars are standard error.

Execution Times

A repeated measures ANOVA showed that the direction had a significant effect on the execution time of the gestures ($F_{7, 63} = 6.753$, $p < .0001$). The diagonals seemed to be consistently slower than their adjacent non-diagonal (straight) strokes. To confirm this, we performed an additional analysis comparing the gesture types (*Straight*, *Diagonal*, or *Tap*). The analysis showed that the gesture type had a significant effect on the execution time ($F_{2, 18} = 43.4$, $p < .0001$). The average times were 163.04ms for *Straight*, 195.99ms for *Diagonal*, and 87.26ms for *Tap* (Figure 6b). Post-hoc pairwise comparison using Bonferroni correction showed that the difference between all pairs was significant ($p < .05$).

SWIPEZONE

The results of our first study suggest that distinguishing between Swipeboard's eight-direction swiping may be feasible, however with some level of error. The results also show that diagonal strokes are significantly slower than non-diagonal swipes and taps. These results motivate us to propose an alternative design that fits the input characteristics of smart eyewear. Our proposed technique, SwipeZone, does so by eliminating diagonal swipes.

Our first modification in designing SwipeZone is to slightly change to the QWERTY layout. The original layout requires swiping diagonally at the second level when typing from the group 'RTYU'. To eliminate diagonal swipes, we move 'P' to the second row, and 'L' to the third, replacing the comma (Figure 7). This layout eliminates the need for diagonal swipes in the second level.

Figure 7. Swipeboard's layout is consistent with QWERTY but has a 4-character group. Our modified layout shifts the locations of 'P' and 'L' so each group has three characters.

Our second modification is to eliminate diagonals from the first level, by replacing them with *zone specific* vertical swipes, similar in spirit to Zone Menus [32]. In particular, we leverage the relatively wide dimensions of smart eyewear touch pads, and divide the input area horizontally into three equally sized zones. For tactile reference, we include a strip of tape on the middle zone (Figure 8). The tape's rough surface is easily distinguishable from the other two zones' smooth surfaces.

With SwipeZone, the diagonal gestures are replaced by vertical swipes in the corresponding zones. For *NE* and *SE* the user swipes up and down in the front zone; for *NW* and *SW* the user swipes up and down in the back zone: for N and S, the user swipes up and down in the middle zone. The taps and horizontal swipes are still target agnostic. For example, to type 'C', the user first swipes down in the front zone, which selects 'ZXC'. The user then swipes horizontally to select 'C' (Figure 9). Similar to the Swipeboard technique, user can cycle through alternative keyboard characters by swiping up twice.

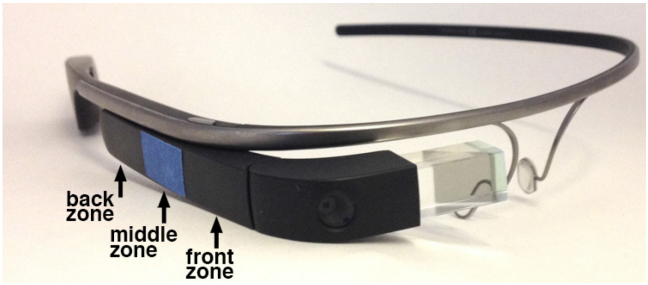

Figure 8. A strip of blue tape helps distinguish the three zones through tactile feedback.

We draw borders around the four regions that require zone-specific swipes, to help remind users that a zone-based gesture is required (Figure 9). We also show the zoomed-in region in-place while keeping the entire keyboard visible in the background. This allows users to plan their next character in parallel to performing the second level gesture for the current character (Figure 9).

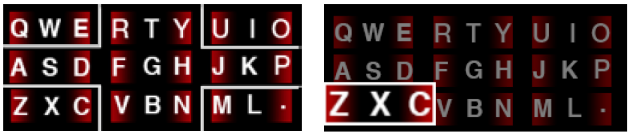

Figure 9. Visual lines help distinguish the characters which require a swipe in one of the side zones. Level 2 shows the selected region in-place, so the entire keyboard is still visible.

STUDY 2: TEXT ENTRY EVALUATION

To evaluate our SwipeZone technique for text entry on smart eyewear, we use a traditional text entry task to measure its performance in comparison with Swipeboard. We are not aware of any existing technique that has been implemented to use the side touch pad of smart eyewear for text entry. Thus, we do not include a baseline technique for comparison; however, we can contrast our results to prior studies on text entry. Our primary goals are to understand how feasible text entry is, if at all, on smart eyewear, and to identify if there are any performance differences in the SwipeZone and Swipeboard techniques. A broader evaluation of the design space of possible smart eyewear text entry techniques is left to future work.

Apparatus

The apparatus was the same as the first study.

Participants

We recruited 16 participants (7 female, 9 male), with an average age of 28.3. All of the participants were right handed and all used their right hand for the text entry task. The participants were recruited externally from a recruiting list that was generated from online postings, and were provided with $50 gift card. None of the participants had prior experience using Google Glass.

Design

A repeated measures mixed design was used. The between-participant independent variable was the technique (*Swipeboard*, *SwipeZone*). The within-participant independent variable was *Block* (1-20). Participants performed the study in one session lasting approximately 80 minutes. The session was broken up into 20 blocks. Each block consisted of 10 trials. In each trial, the user typed in a single 5-letter word, randomly chosen from Mackenzie's phrase set [18]. This resulted in a total of 200 trials per participant, and a total of 3200 trials overall.

Procedure

Before the study began, the assigned technique was demonstrated to participants using a Samsung Galaxy S4 phone. After explaining the technique, users performed a warm-up block on the phone, which consisted of 10 words.

The physical set up and seating position was the same as described in Study 1. Before the start of each trial, the system displayed a 5-letter word. After reading the word the participant tapped to begin the trial. The word then disappeared and the keyboard was displayed. This prevented participants from reading the word while the trial time was being timed. The participant used the assigned technique to transcribe the word (Figure 10). If the user typed the wrong character a beep was sounded and the correct word was displayed on the screen. However, the incorrect letter was not typed, so that users would not need to delete characters. The user would need to retry until they typed the correct character. This was recorded as a "hard error". We also recorded "soft errors" when the user's initial stroke activated the wrong region of the keyboard. The trial was completed when all five characters were correctly typed.

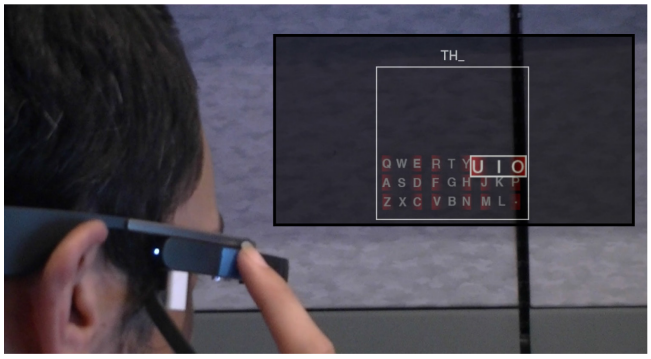

Figure 10. Users performed the text entry task on a Google Glass unit. Correctly typed letters are displayed in white.

For Swipeboard, we used the updated layout with each group containing only 3 characters (for consistency with SwipeZone). The Cartesian tessellation (Figure 6a) was used to classify the gestures.

Results and Analysis

Character Entry Time

The main measurement was the character entry time. Our analysis is based on error-free characters (we also provide an analysis of errors later in this section). Similar to prior work [7] we divide each character entry time into four phases: the time until the first touch event (*First Up*), the time taken for the first swipe or tap (*First Action*) the time until the second touch event (*Second Up*) and the time taken for the second swipe or tap (*Second Action*). The total character entry time was the sum of these four phases.

We first analyze the per-character completion time for error-free trials. A repeated measures ANOVA showed a main effect for block ($F_{19, 266} = 34.9$, $p < .0001$), but did not reach significance at the $p < .05$ level for the keyboard type ($F_{1, 14} = 3.350$, $p = 0.089$). The overall per-character completion times were 1.97s for Swipeboard and 1.67s for SwipeZone. The lack of a significant effect, despite a 15.2% performance difference, is likely due to smaller sample-size.

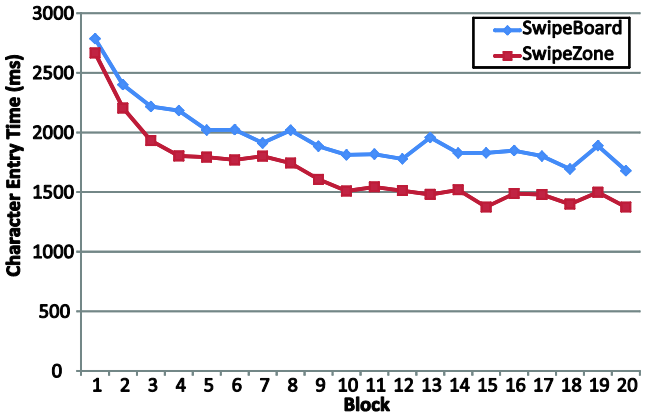

Figure 11. Error-free character completion time by block.

As illustrated in Figure 11, the performance differences do seem to increase as training continues. When we repeat the analysis on just the last 10 blocks, the difference does reach a significant level ($F_{1, 14} = 5.136$, $p < 0.05$). The entry times for the last 10 blocks are 1.812s and 1.467s for Swipeboard and SwipeZone, respectively.

The effect of the actual character being entered was also significant ($F_{25, 375}$=16.9, $p < .0001$). Figure 12 illustrates the character entry times for each character. The entry times for most common characters are fairly uniform. Unsurprisingly, 'G' is one of the fastest, since its gesture consists of two taps. The three slowest characters are those that appear rarely in the vocabulary set (J, X, Q). This shows evidence that learning with the technique occurs not only at the technique level, but may also occur at the individual character level.

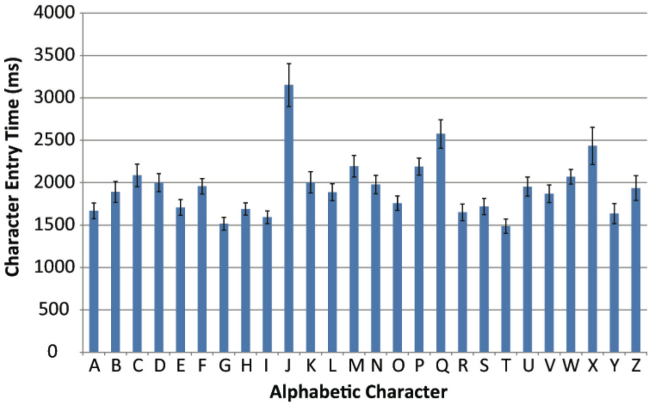

Figure 12. Completion times for each character. Uncommon characters (e.g., J, X, Q) took significantly longer, suggesting learning also happened at the individual character level.

Words per Minute (WPM)

Figure 13 shows the error-free WPM rates for each of the techniques. In the last block, the WPM rates were 8.73 for SwipeZone, and 7.14 for Swipeboard. Both rates increase according to the Power Law of Learning, fitting to the curves at $R^2 = 0.95$ for SwipeZone and $R^2 = 0.90$ for Swipeboard. If these trends continued, SwipeZone would reach 10 WPM after a total of 40 blocks.

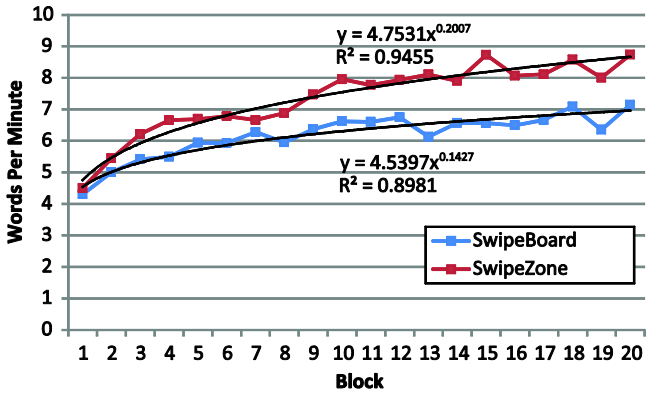

Figure 13. WPM rates and the associated power curves.

Performance Model

To further understand the difference in the two techniques, we look at a breakdown of the character entry time, by each stage of the performance model (Figure 14).

An interesting effect is that the character entry times are dominated by the *First Up* phase, where the user locates their character and plans the gesture input. Anecdotally, we observed that users, after some blocks of training, started to plan for both levels of the gesture before performing the associated actions. This could explain why the *Second Up* phase is much shorter. Figure 14 also shows that the actual stroke times (*First Action, Second Action*) were much shorter than the two phases that involved decision-making.

The other interesting observation is that the main difference between the two techniques is at the *Second Up* phase. The average *Second Up* times were 564ms for Swipeboard and 408ms for SwipeZone. This difference was the one phase where *Technique* had a significant effect ($F_{1, 14} = 10.695$, $p < .01$). This increased time seemed to be a result of the Swipeboard users waiting to see if their first stroke was interpreted correctly or not, whereas the tactile feedback of SwipeZone gave users immediate feedback.

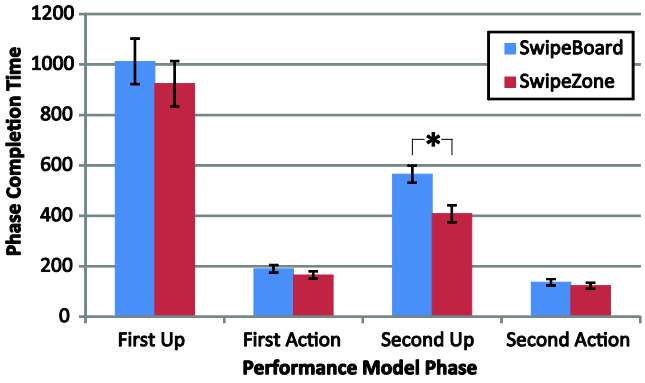

Figure 14. Times for the four phases of the performance model (based on Chen et al. [5]). SwipeZone contributed to a significantly shorter *Second Up* phase. (*$p < .01$)

Errors

We analyzed both soft errors (errors in the first level gesture) and hard errors (errors at the second level gesture). The soft error rates were 17.8% for Swipeboard and 15.8% for SwipeZone. The difference was not significant. The hard error rates were also not significantly different – 9.3% for Swipeboard and 9.1% for SwipeZone. The majority of the hard errors were due to errors made at the first level: users continued with the character entry (causing a hard error) rather than cancelling the erroneous first-level selection (causing a soft error). The hard error rate, when the first-level selection was correct, was 2.4% and 1.8% for Swipeboard and SwipeZone, respectively.

These results show that the majority of errors made with the techniques resulted from the first level. And while the SwipeZone technique reduced the execution time in *Second Up* phase, the technique itself is still error prone. We hypothesize, based on our observations, that a proportion of these errors are caused by users choosing the wrong gesture to perform (e.g. *E* instead of *W),* and not choosing the right gesture but performing that gesture improperly (e.g. swiping in the wrong zone).

LIMITATIONS

As an initial study into the issue of text entry on smart glasses, the study design has a number of limitations which should be considered and may impact for replicability of the results.

First, to simplify the task for users, we prevented incorrect characters from being displayed, so that users would not have to correct those errors. The impact of error correction on performance thus needs to be better understood.

Second, users entered short words instead of the phrases typically used in text entry studies. This again was done to simplify the task for participants. Assessing the impact of cognitive load when full phrases need to be entered should be investigated further.

Our study also consisted of one long session, instead of multiple shorter sessions. Having multiple sessions could help understand learning retention, and could also reduce fatigue on behalf of the users. While we did provide breaks throughout our study, participants may have had lower performance in the final blocks due to fatigue. Having multiple sessions could also increase the total amount of data which is captured.

In general, this study may be considered a feasibility study to show that the technique does in fact work, where more formal analysis should be conducted in the future to gather specific metrics of the techniques, such as WPM and KSPC in more externally valid scenarios. This will provide more insights into the performance of the technique in comparison to existing text-entry methods in the literature.

In addition to the limitations of our study, there is also room for performance improvements of the techniques we tested. SwipeZone users improved over time, and achieved almost 9 WPM by the end of the trials. However, this was lower than the performance in the original Swipeboard study [7], for various possible reasons. Foremost, the original Swipeboard study was run on a simulated watch-sized screen using an iPad rather than an actual wearable device. As such, usability challenges due to a small form factor were simulated but not fully tested. Second, the prior study was based on a reduced phrase set consisting of only 5 letters, to intentionally accelerate learning times. While this shed light on the novice to expert transition, it makes it difficult to directly compare our results.

The observed error rates will also need to be addressed before our tested techniques could be deployed. Our initial study shows that there is inherent user error in the gestures being performed. Our second study shows that eliminating diagonal gestures helps, but doesn't eliminate errors. As with prior research [7, 22], we did not include statistical language models or advanced error correction. Including such techniques could be one way to reduce error rates.

FUTURE WORK

For the SwipeZone technique, the tactile feedback allowed users to know where their finger was, but only once it was down. Users had to rely on their proprioception to touch down on the correct area. Technologies that provide mid-air tactile feedback could be an interesting way to allow users to know which zone their finger was above.

Fatigue is another issue to be explored further. Because of our prolonged study, we allowed users to rest their elbow during the text entry task, and to perform the study from a seated position. It would be important for future work to formally investigate fatigue issues and look at how the text entry would be impacted by different postures (standing, lying down) or activities (standing, walking, riding a bus).

Given our work is the first known working implementation of a gesture-based approach for smart eyewear text entry, there was no clear baseline comparison. However, there is a large design space of possible adaptions from the mobile text entry literature. A thorough investigation and evaluation of this design space is beyond the scope of our work but would be interesting for future studies.

It is also important to discuss the generalizability of our results. Our goal was to evaluate the feasibility of gesture-based text entry on smart eyewear. As the most readily available device, we choose to implement and evaluate our techniques using a Google Glass. We believe our techniques and high-level results generalize to other smart eyewear devices that are being manufactured with on-device touch pads (SiME Smart Glasses, Recon Jet). However, the exact dimensions of each device's touch pad may differ, and it would be interesting to understand how much impact this would have on our results and observations. Similarly, it would be interesting to investigate adaptations of the technique to other types of wearable devices that may have similarly sized touchpads, such as fitness bands and electronic clothing.

CONCLUSION

We have investigated the feasibility and human factors associated with performing gesture-based text entry on the side touch pad of smart eyewear, and both demonstrated and compared two methods of performing text entry using this input area. Our study reveals that our redesign of the Swipeboard technique offers benefits, and that text entry is possible using smart eyewear as both the input and output device. We hope this work can inform and inspire future work on gesture-based text entry for smart eyewear devices.

REFERENCES

1. Amma, C., Georgi, M., & Schultz, T. Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3D-space handwriting with inertial sensors. *ISWC '12*. 52-59.
2. Arif, A. S., Pahud, M., Hinckley, K., & Buxton, B. (2014). Experimental study of stroke shortcuts for a touchscreen keyboard with gesture-redundant keys removed. *Graphics Interface*. 43-50.
3. Bajer, B., MacKenzie, I. S., & Baljko, M. (2012). Huffman Base-4 Text Entry Glove (H4 TEG). In *International Symposium on Wearable Computers*. 41-47.
4. Blasko, G. and Feiner, S. Evaluation of an Eyes-Free Cursorless Numeric Entry System for Wearable Computers. *Wearable Computers*, (2006), 21–28.
5. Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., & Furnas, G. Pad++: A zoomable graphical sketchpad for exploring alternate interface physics. *JVLC*. 1996, 7(1), 3-32.

6. Castellucci, S. J., & MacKenzie, I. S. Graffiti vs. unistrokes : an empirical comparison. *CHI '08*. 305-308.
7. Chen, X. A., Grossman, T., & Fitzmaurice, G. Swipeboard: A Text Entry Technique for Ultra-Small Interfaces That Supports Novice to Expert Transitions. To appear *UIST '14*.
8. Clawson, J., Lyons, K., Starner, T., & Clarkson, E. (2005). The impacts of limited visual feedback on mobile text entry for the twiddler and mini-qwerty keyboards. *International Symposium on Wearable Computers*. 170-177.
9. Hong, J., Heo, S., Isokoski, P., & Lee, G. (2015). SplitBoard: A Simple Split Soft Keyboard for Wristwatch-sized Touch Screens. *CHI.* 1233-1236.
10. Isokoski, P., & Raisamo, R. (2000). Device independent text input: A rationale and an example. In *Proceedings of the working conference on Advanced visual interfaces* (pp. 76-83).
11. Kim, S., Sohn, M., Pak, J., & Lee, W. One-key keyboard: a very small QWERTY keyboard supporting text entry for wearable computing. *OzCHI '06*. 305-308.
12. Kristensson, P. O., & Zhai, S. SHARK 2: a large vocabulary shorthand writing system for pen-based computers. *UIST '04*. 43-52.
13. Leiva, L. A., Sahami, A., Catalá, A., Henze, N., & Schmidt, A. (2015). Text Entry on Tiny QWERTY Soft Keyboards. *ACM CHI*. 669-678.
14. Li, F. C. Y., Guy, R. T., Yatani, K., & Truong, K. N. The 1line keyboard: a QWERTY layout in a single line. *UIST '11*. 461-470.
15. Liu, Y., Liu, X., & Jia, Y. Hand-gesture based text input for wearable computers. *ICVS '06*. 8-14.
16. Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., & Looney, E. W. Twiddler typing: One-handed chording text entry for mobile phones. *CHI '04*. 671-678.
17. MacKenzie, S. Mobile text entry using three keys. *CHI '02*. 27-34.
18. MacKenzie, I. S., & Soukoreff, R. W. *Phrase sets for evaluating text entry techniques*. *CHI '03 EA*. 754-755.
19. MacKenzie, I. S., Soukoreff, R. W., & Helga, J. 1 thumb, 4 buttons, 20 words per minute: Design and evaluation of H4-Writer. *UIST '11*. 471-480.
20. Matias, E., MacKenzie, I. S., & Buxton, W. (1994). Half-QWERTY: Typing with one hand using your two-handed skills. *CHI Companion.* 51-52.
21. Nesbat, S. B. (2003). A system for fast, full-text entry for small electronic devices. *Proceedings of ACM Multimodal interfaces*. 4-11.
22. Oney, S., Harrison, C., Ogan, A., & Wiese, J. ZoomBoard: a diminutive QWERTY soft keyboard using iterative zooming for ultra-small devices. *CHI '13*. 2799-2802.
23. Partridge, K., Chatterjee, S., Sazawal, V., Borriello, G., & Want, R. TiltType: accelerometer-supported text entry for very small devices. *UIST '02*. 201-204.
24. Perlin, K. Quikwriting: continuous stylus-based text entry. *UIST '98*. 215-216.
25. Rico, J., & Brewster, S. Usable gestures for mobile interfaces. *CHI '10*. 887-896.
26. Rico, J., & Brewster, S. 2010. Gesture and voice prototyping for early evaluations of social acceptability in multimodal interfaces. *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction.* Article 16.
27. Rosenberg, R., & Slater, M. The chording glove: a glove-based text input device. *ToSMC*. 1999, 29(2), 186-191.
28. Wigdor, D., & Balakrishnan, R. TiltText: using tilt for text input to mobile phones. *CHI '03*. 81-90.
29. Wigdor, D., & Balakrishnan, R. A comparison of consecutive and concurrent input text entry techniques for mobile phones. *CHI '04*. 81-88.
30. Wobbrock, J.O., Fogarty, J., Liu, S.-Y.S., Kimuro, S., and Harada, S. The angle mouse: target-agnostic dynamic gain adjustment based on angular deviation. *CHI '09*, 1401–1410.
31. Wobbrock, J. O., Myers, B. A., & Kembel, J. A. EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. *UIST '06*. 61-70.
32. Zhao, S., Agrawala, & Hinckley, K. Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. CHI'06. 1088-1086.