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. The Swipeboard technique, recently proposed for ultra-small touch screens such as smart watches, may be particularly suitable for smart eyewear, as it supports eyes-free input. We investigate the limitations and feasibility of implementing Swipeboard on Google Glass, using its 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 15.2% higher than Swipeboard, with a statistically significant improvement in the last half of the study blocks. 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 their feasibility and the associated human factor issues.

INTRODUCTION
Smart eyewear (e.g., Google Glass) enables 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. While text entry may not be a primary task on smart eyewear, there are scenarios where a user may wish to enter text, e.g., quickly replying to a text message, or updating one's status on social media.

Prior research has explored solutions to enable entering text while wearing a head-mounted display. One solution is to use a supplementary device, such as the Twiddler [13]. 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 [12]; yet they remain a suboptimal solution as such techniques are often computationally expensive and could potentially be error-prone due to the uncertainty of the environment. Voice input may be a useful option in certain situations, but when surrounded by peers its usage might incur privacy concerns and become socially unacceptable [21].

Some recent research on wearable text entry has taken a gesture-based approach using a device's touch input. For example, Swipeboard is a watch-based technique that specifies each character using only two touches: the first touch selects a subgroup of keys and the second touch specifies a key within the subgroup [5]. Such gesture-based methods offer a subtle, effective and low-cost text entry solution. More importantly, the technique is target-agnostic, making it a promising candidate for use on, for instance, the side touch pad of Google Glass. However, this technique has yet to be implemented, deployed or tested on smart eyewear.

In this paper we investigate the limitations and feasibility of applying a gesture-based text entry technique (Swipeboard) to a smart eyewear form factor (Google Glass). In our initial study we evaluate the recognition accuracy of the 8 directional swipes required for Swipeboard, and analyze how well these swipes can be distinguished from one another, when performed with the Google Glass touchpad. The results show that the unique configuration of the Google Glass touch pad makes it difficult to distinguish between the 8 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).

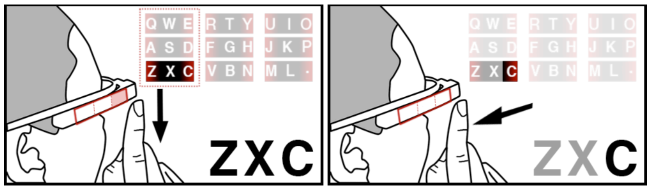

Figure 1. SwipeZone divides the width of the Google Glass side touch pad into three zones, swiping down on the front zone selects the lower-left group 'ZXC', then swiping to the back selects 'C' – the rightmost key in the group.

Our second study tests both Swipeboard and SwipeZone on a Google Glass unit through a series of text entry tasks. 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. Further, we compare the results to prior studies to show that SwipeZone's entry speeds are similar to those achieved on smart watch devices, yet with a higher error rate. 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 and the associated human factor issues.

RELATED WORK

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

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 [13]. The chording glove embeds the buttons into the glove where users can specify characters directly using hand gestures [20]. Liu et al. propose a vision-based mechanism that uses a head-mounted camera to recognize mid-air handwriting [12].

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 [8].

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. 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.

Text Entry Techniques for Small Form Factors

Past work has explored various text entry techniques for devices with very small form factors. Some researchers propose the use of motion sensors, e.g., using tilting to specify characters for text entry [19, 26]. MultiWidget uses a dialing gesture along the watch's edges to specify a numeric value [2]. Zoomboard brings the zoomable user interface [3] idea to the design of a QWERTY keyboard on very small touch screens [18].

Another approach is using alternate key mapping, such as chording, key selections or gestural shortcuts. For example, Wigdor and Balakrishnan adds three chording keys to speed up typing on a numeric phone keypad [27]. Mackenzie demonstrates the use of three keys to enable selection-based text entry [14]. The 1line keyboard incorporates touch into key selection, reducing the keyboard into one line of keys [11]. Gestural shortcuts can also be effective with small form factors. EdgeWrite uses stylus-based gestures guided by the physical edges of a device, thus making it easier and faster to perform [27]. 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 [5].

Importantly, gesture-based input like Swipeboard is 'target-agnostic' [28], making it promising for smart eyewear, as absolute touch coordinates are never required. However, aside from a proof-of-concept video of the 1line keybaord[1], we are unaware of any implementations or evaluations of such techniques for smart eyewear.

Gesture-based Text Entry Techniques

To take a closer look at gesture-based text entry, we summarize two key steps for designing such techniques.

1) *Encoding the characters into a gestural vocabulary*. Chen et al. summarized two ways of encoding characters: a continuous approach maps a word or a character to a continuous stroke (e.g., Graffiti [4], EdgeWrite [27], Shark[2] [9]), and a discrete approach maps each character to a number of symbolic tokens (e.g., H4-Writer uses a base-4 encoding [15] and Quikwriting uses base-9 [20]).

2) *Designing techniques to perform the gestures*. The next step is concerned with designing interaction techniques for the users to perform the gesture and specify the encoded character. Continuous encoding typically uses touch or stylus input as it fits well with its continuous nature. Discrete encoding, however, often has design options. A naïve approach is simply letting users type in the code from their rote memory, such as using Morse code. Most successful discrete solutions, however, often employ certain menu selection techniques that allow users to recognize and choose the codes rather than recalling them. For example, Quikwriting [20], H4-Writer [15] and Swipeboard [5] all use techniques akin to a marking menu [10].

However, depending on the form factor of the device and its screen real estate, marking menus might not be the best choice (for instance, considering a short rectangular touch area). There are other techniques that repurpose the original marking menu design to overcome its limits. For example, Zhao et al. introduced the ideas of zone menu [30], which divides a large marking menu into smaller ones located in separate zones, thus reducing the complexity of specifying precise orientation when selecting an item. Their polygon menu [30] has a similar idea: both a stroke's orientation and its position are used to specify an item. To accelerate multi-

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

level selection, Zhao and Balakrishnan studied the technique of using small discrete strokes and found it to be faster and more effective than a continuous one [31]. This work in menu selection can potentially serve as guidelines for our work on gestural text entry on smart eyewear.

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 the Google Glass.

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).

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

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. used previous performance model data to estimate that in optimal conditions an expert could perform at 464ms per character, or 25.87 WPM.

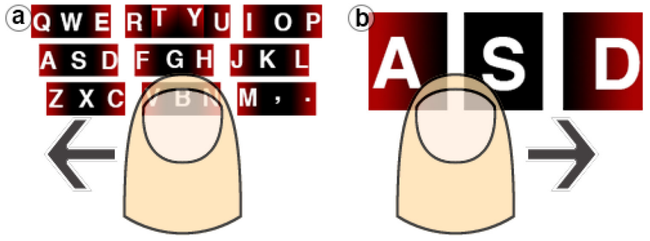

Figure 2. The original Swipeboard technique. a) The first swipe specifies one of the nine regions subdivided from a QWERTY keyboard. b) The second swipe specifies the character, in this example, 'D'. Figure from Chen et al. [5].

Adaptation to Smart Eyewear

Adapting any text entry technique to smart eyewear raises some questions. First, it is important to discuss the motivation for requiring text entry at all. While we do not expect users to compose long emails or perform intense document editing using smart eyewear, we feel it is likely there will be cases where users will want access to text entry. Examples include: composing a short tweet or text message, looking up information on the web, or searching for an app or contact name. While speech input may be possible, there are situations where speech is not socially acceptable [21], or does not perform well [7].

Thus, our goal is to explore alternate approaches to support text entry on smart eyewear, focusing on utilizing its built-in touch-sensitive area. The Swipeboard technique, originally deployed on a watch-sized touch screen, is a promising candidate for an exploratory implementation.

Implementation

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 initial implementation exactly reproduced the Swipeboard technique. The X and Y coordinates of the touch events were obtained through the Android *MotionEvent* object that the Google Glass touch pad produces. There were some interesting observations from this initial implementation.

First, it was unclear how to map the X direction from the input to display. Because the left and right swipes are actually in the forward and backward directions, a 90° rotation is required. Depending on the direction of this rotation, the X direction could be mapped in two ways.

Second, there seemed to be an advantage of using the Google Glass side touch pad, because the finger never occludes the display of the keyboard, unlike the previous small display implementation. As such, the user has an opportunity to see the character map at all times.

Third, 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.

Most 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 final two 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 successfully 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 3” by 0.41”, with a resolution of 1366x187. It is unclear how well Swipeboard’s atomic gestures can be performed and distinguished on a touch pad with this unique form factor. In this study, we investigate the feasibility of deploying Swipeboard’s gestures on the touch pad of Google Glass.

Apparatus

The study was performed using a Google Glass unit. The study software running on the Google Glass was written using the ADT plugin for the Eclipse IDE. 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 the Google Glass system.

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.

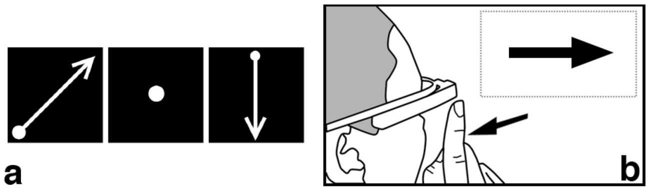

Figure 3. a) Exemplar visual stimulus (NE, TAP, S); b) illustration of the mapping of an EAST stroke.

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

The main focus of our analysis is on the vector (*X, Y*) indicating the directionality of each atomic gesture. We also provide an analysis on execution times.

Analyzing and Visualizing Swiping on the Glass

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.

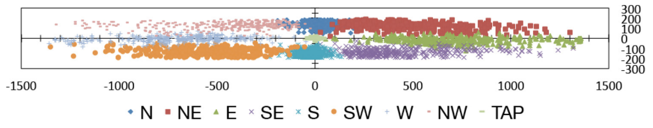

Figure 4. Scatterplot of each gesture’s end points in Study 1.

We summarize several key findings from this data. First, using the vector angles to determine a stroke’s direction may not be effective. Second, even if angles can be used, the correct angles cannot be defined by evenly dividing a circumference into eight 45° sectors. Due to the short and wide form factor of the touch pad, the average angles used for the diagonal strokes are highly skewed towards the *X* axis. The average absolute deviation from the *X* axis for the diagonal strokes is 18.8°.

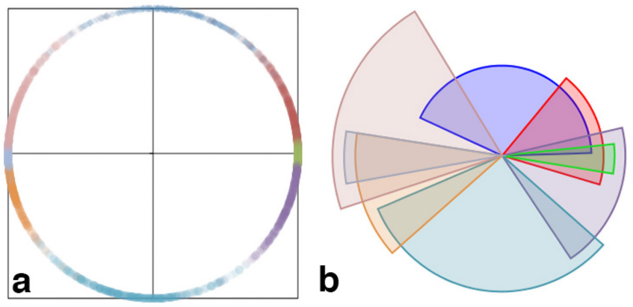

Figure 5. Directionalities of the atomic gestures from Study 1: scatterplot of the normalized vectors computed from each gesture (a); possible ranges of each swipe, computed from their mean ± 3× standard deviation (b).

Building Models to Recognize Swiping Directions

To better understand the feasibility of utilizing this gesture set, we build customized algorithms to show how well the swipes can be recognized.

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 *tapx* and *tapy*.

The next step distinguishes different swipes. We experimented with two potential methods, based on either the vector angle, or the vector coordinates.

Using the angle is a typical approach for determining a swipe direction, where each swipe has an upper and lower bound angle. For example, for the original Swipeboard gestures, the swipes are divided by 45°. For example, if the angle is between -22.5° and 22.5° it is recognized as *EAST*. Due to the observed skewing, we parameterize the angles for swiping on the Google Glass touch pad, as θ_1 and θ_2, to define all 8 direction zones (Figure 6a).

Alternatively, the swiping directions can be defined by Cartesian coordinates (corresponding to the end points of the swipes). As shown in Figure 6b, we can define 8 quadrants by a $\boldsymbol{\delta_x}$ and $\boldsymbol{\delta_y}$ parameter. For example, a stroke with $X > \boldsymbol{\delta_x}$ and $Y > \boldsymbol{\delta_y}$ would be classified as *NE*.

To find the optimal parameter values we perform a naïve stepwise iterative search across all possible combinations of the parameter pairs. We iterate with steps of 1px for $\boldsymbol{\delta}$ and 0.1° for θ. We calculated the optimal parameters and resulting accuracy on a per-user basis, and also across the entire data set of all users. Defining accuracy on a per-user basis shows what accuracies would be possible if there was a required calibration, whereas the accuracies across the entire data set show a more realistic scenario where a fixed heuristic is applied for all users.

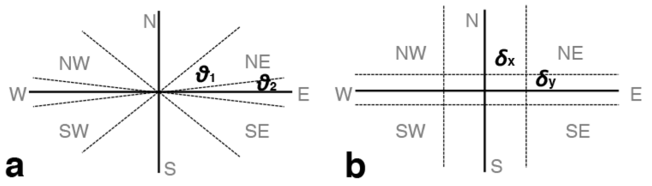

Figure 6. The gestures can be determined based on 1) angles, θ_1 and θ_2, which divide the circumference into eight sectors; or 2) coordinates, δ_x and δ_y, which groups the end points of each swipe into eight swiping directions.

Accuracy of Recognition

The algorithm first calculated the optimal values for recognizing a *TAP* (*tapx* = 53, *tapy* = 42). The accuracy of this step was 99.97%, indicating its high generalizability across users. We fixed these parameters without further per-user tuning.

The optimized values of $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ for each participant are shown in Table 1, together with the resulting accuracies. The average of all per user accuracies is 95.9%. When optimizing across the entire data set, the values of $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ are 48.6 and 4.8, with an accuracy of 93.5%. This method would result in a 6.5% error rate.

The optimal values of $\boldsymbol{\delta_x}$ and $\boldsymbol{\delta_y}$ are also shown in Table 1, together with the resulting accuracies. The average of all per user accuracies is 96.7%. When optimizing across the entire data set, the values of $\boldsymbol{\delta_x}$ and $\boldsymbol{\delta_y}$ are 136 and 59, with an accuracy of 94.0%. This method would result in a 6% error rate.

Subject	θ_1	θ_2	Accuracy	δ_x	δ_y	Accuracy
1	44	2.6	97.2	175	39	98.6
2	43.8	7.8	98.6	112	62	97.9
3	49.2	5.4	85.1	268	60	91.3
4	50.4	5.6	96.9	134	57	95.8
5	33.2	3.6	92.7	189	58	92.7
6	44.8	11.2	99.0	92	65	99.0
7	51.6	6.4	96.2	82	62	98.3
8	49.2	3.8	97.6	141	56	97.6
9	20.2	4.2	99.3	190	67	98.3
10	47.6	6	96.2	83	83	97.6
Average			**95.9**			**96.7**
All	48.6	4.8	**93.5**	136	59	**94.0**

Table 1. Parameter values and resulting accuracies for θ_1 and θ_2. Parameter values and resulting accuracies for δ_x and δ_y.

It can be seen in Table 1 that Subject 3 has lower accuracies compared to the other users. Upon examining the data, there is a slight rotation in Subject 3's data points. This could be due to how the user wore the glasses, and how they oriented their head during the study.

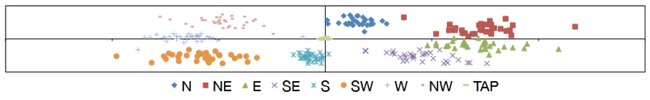

Figure 7. Data points for Subject 3 are slightly rotated.

Execution Times

A repeated measures ANOVA showed that the direction had a significant effect on the execution time of the gestures ($F_{7,\ 63} = 6.753$, $p < .0001$). Figure 8a shows that the diagonals seem 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 8b). Post-hoc pairwise comparison using Bonferroni correction showed that the difference between all pairs was significant ($p < .05$).

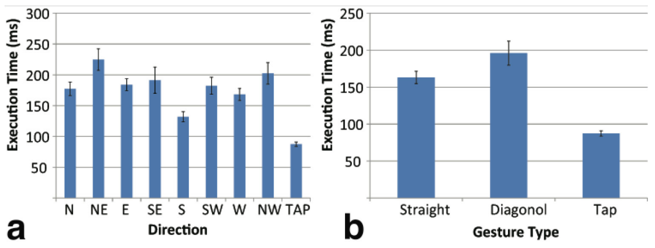

Figure 8. Execution time by gesture shows diagonal swipes took significantly longer. Error bars are standard error.

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, called SwipeZone, requires only the tap and four Straight swiping gestures, thus eliminating the need to swipe diagonally, lowering the potential recognition errors and reducing users' execution time.

Our first modification in designing SwipeZone is to slightly change to the layout of the QWERTY keyboard used in the Swipeboard technique. 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 9). This provides a more consistent grouping, as each group now contains three characters, eliminating the need for diagonal swipes in the second level. It also provides a more visually aligned layout.

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 [30]. In particular, we leverage the relatively wide dimensions of the Google Glass touch pad, dividing it horizontally into three equally sized zones (front, middle and back, marked in Figure 10).

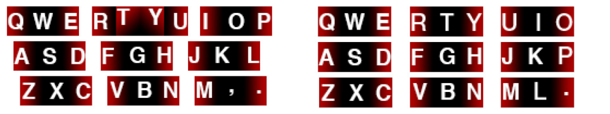

Figure 9. Swipeboard's layout is consistent with QWERTY but has a 4-character group. Our modified layout shifts the locations of 'P' and 'L' so each group has three characters.

With SwipeZone, the diagonal gestures are replaced by vertical swipes in the corresponding zones, as shown in Figure 11a: 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 1).

For tactile reference, we include a strip of tape on the middle zone (Figure 10). The tape's rough surface is easily distinguishable from the other two zones' smooth surfaces and does not impact the touch sensing capabilities.

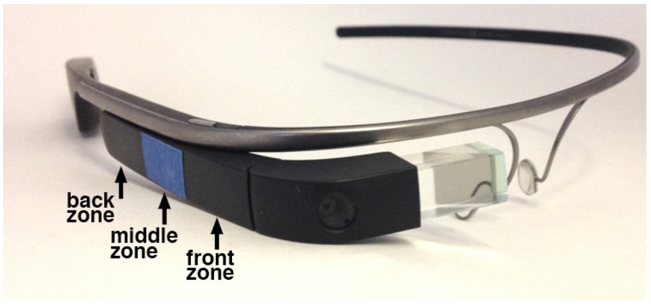

Figure 10. A strip of blue tape helps distinguish the three zones through tactile feedback.

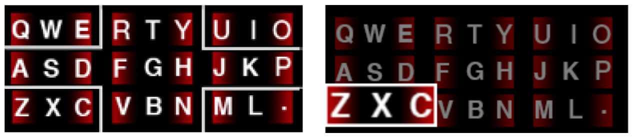

Figure 11. Visual lines help distinguish the characters which require a swipe in one of the side zones. Level 2 shows the selected region in-place, so the entire keyboard is still visible.

We draw borders around the four regions that require zone-specific swipes, to help remind users that a zone-based gesture is required (Figure 11a). 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 11b).

STUDY 2 – EVALUATING SWIPEZONE

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.

Apparatus

The apparatus was the same as the first study. A pair of Google Glass was used, and the software was written using the ADT plugin for the Eclipse IDE. 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 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 from an external recruiting list 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). We chose a between-subject design for the techniques so that we could provide adequate training time for each participant, and reduce cross-technique learning effects.

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 [15]. 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.

Participants sat in a chair facing a black background that 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's armrest to prevent fatigue.

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. The participant used the assigned technique to transcribe the word (Figure 12). 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.

A message was displayed at the start of each block indicating how many blocks remained. Users were told they could take breaks between blocks to rest their eyes or arms.

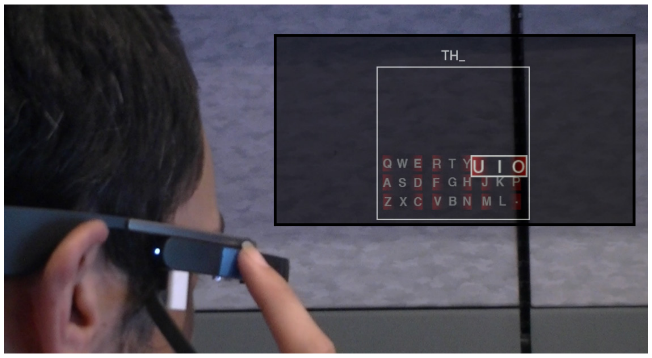

Figure 12. Users performed the text entry task on a Google Glass unit. Correctly typed letters are displayed in white.For Swipeboard, we used the updated layout with each group containing only 3 characters (for consistency with SwipeZone). The Cartesian tessellation (Figure 6b) was used to classify the gestures as it had higher recognition accuracy than the angle-based approach.

Based on observations from the first study, we allowed users to choose the input display mapping for the horizontal direction. We set this mapping automatically using a short calibration. The user was shown a series of 20 alternating *East* and *West* arrows, and was asked to swipe in the direction they thought each arrow represented. We then automatically set the mapping based on this individual preference.

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 [5] 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 quite reach significance at the $p < .05$ level for the keyboard type ($F_{1,14} = 3.350$, $p = 0.089$). However, Figure 13 does show an apparent trend in the data. 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 the fact that the

Technique was a between-subject variable. However, based on the statistical test, we cannot make definite claims about the performance differences of the two techniques.

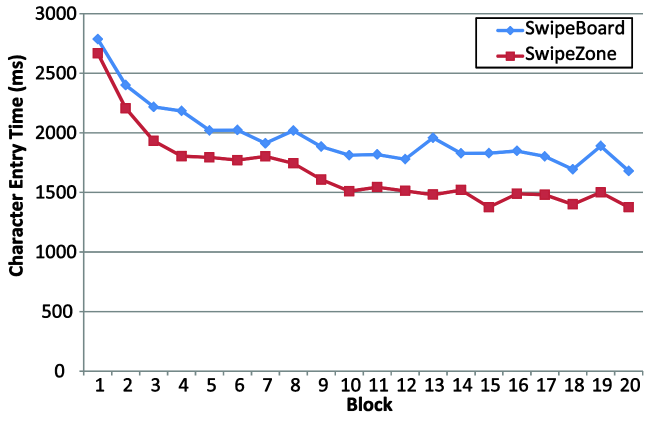

Figure 13. Error-free character completion time by block.

As illustrated in Figure 13, 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.

We also looked at the character entry time for each of the five characters in the tested words. As shown in Figure 14. There was a significant effect ($F_{4,56} = 14.041$, $p < .0001$). A post hoc pairwise comparison with Bonferroni adjustment shows that the first character is significantly slower than the others ($p < .05$ in all cases). This was likely caused by the keyboard not being visible until the trial started. Thus, there was a longer visual search for the first character. For remaining characters, the user could look ahead while entering the previous character. This, in part, validates our design decision to show the characters in place, instead of replacing the entire keyboard with a zoomed-in region.

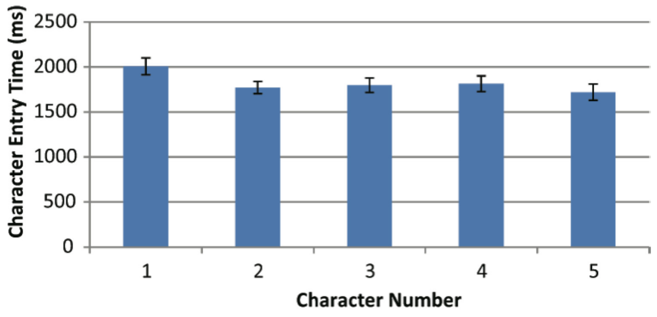

Figure 14. Completion times for the 5 characters in each word. The first character is significantly slower than the others.

The effect of the actual character being entered was also significant ($F_{25,375}$=16.9, $p < .0001$). Figure 15 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 also at the individual character level.

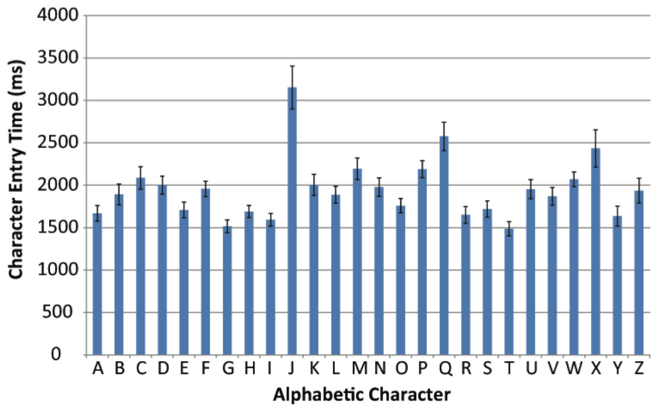

Figure 15. Completion times for each character. Uncommon characters (e.g., J, X, Q) took significantly longer, suggesting learning also happened at the individual character level.

Words Per Minute (WPM)

Figure 16 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.

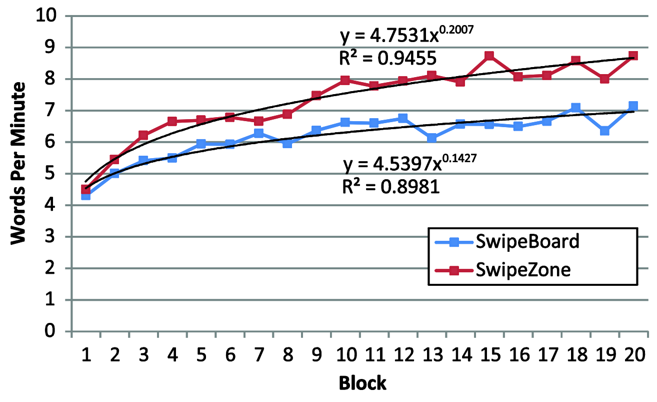

Figure 16. WPM rates and the associated power curves.

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 17).

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 17 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 successful. Users had a higher tendency to do this in the Swipeboard condition because of the difficulty of distinguishing between the 8 directional gestures. In contrast, with SwipeZone, users performed no diagonal swipes and had tactile feedback during their stroke to inform them if it would be successful. Thus, the SwipeZone users could potentially require no 'wait time' for any visual or audio feedback and could immediately proceed to the next level of the gesture.

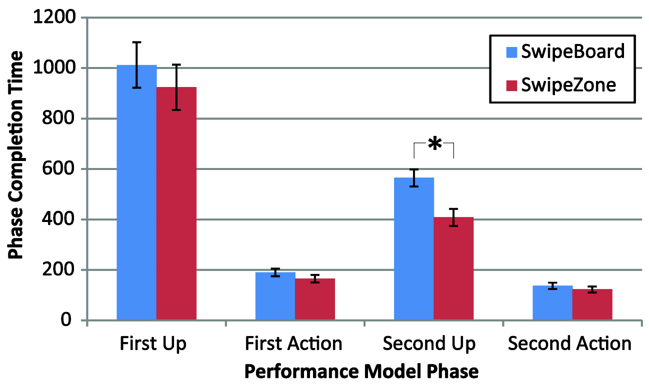

Figure 17. Times for the four phases of the performance model (based on Chen et al. [5]). SwipeZone contributed to a significantly shorter *Second Up* phase. (*p < .01)

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 (Figure 18).

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. Although it is difficult to analyze, 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),* as a pose to choosing the right gesture but performing that gesture improperly (e.g. swiping in the wrong zone).

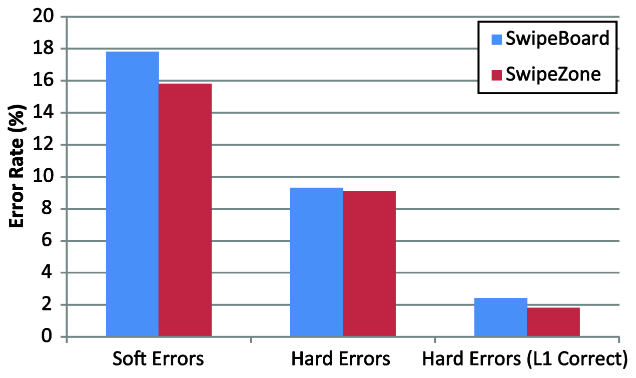

Figure 18. Error rates for each technique. Soft errors were made in the first step (L1), hard error in the second step (L2). We also show hard errors rates when L1 was correct.

DISCUSSION AND FUTURE WORK

While the results over our second study are encouraging, our data also shows that there is still room for performance improvements. 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 [5], 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.

One possible enhancement would be to increase the fidelity of the tactile feedback on the touchpad. For example, a gradient could be used instead of a uniform texture, so users could feel exactly where their finger is within the middle zone. For the Swipeboard technique, thin lines on the touchpad surface could potentially be used as 'tracks' which the finger could follow, similar to EdgeWrite [29].

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, before touching down on the touchpad.

Our studies also revealed an interesting issue related to the mental rotation required to perform gestures on the side touchpad. There is a certain level of ambiguity as to how the horizontal direction is mapped from the display to the input area. In our second study, nine participants used our default mapping (Figure 3b), while 7 preferred to have the mapping reversed. A calibration, as we performed in our second study, would be recommended for future gesture-based techniques. Our first study also showed that a user's gestures could be impacted by the current tilt of their head. This suggests that the device's accelerometer could potentially be useful in normalizing the stroke directions. We leave this topic for future research.

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).

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. Blasko, G. and Feiner, S. Evaluation of an Eyes-Free Cursorless Numeric Entry System for Wearable Computers. *Wearable Computers*, (2006), 21–28.
3. 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.
4. Castellucci, S. J., & MacKenzie, I. S. Graffiti vs. unistrokes : an empirical comparison. *CHI '08*. 305-308.
5. 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*.
6. Isokoski, P. Text input methods for eye trackers using off-screen targets. *ETRA '00*. 15-21.
7. Gong, Y. Speech recognition in noisy environments. (1996). *Speech communication*, 16(3), 261-291.
8. 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.
9. Kristensson, P. O., & Zhai, S. SHARK 2: a large vocabulary shorthand writing system for pen-based computers. *UIST '04*. 43-52.
10. Kurtenbach, G., & Buxton, W. User learning and performance with marking menus. *CHI '94*. 258-264.
11. 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.
12. Liu, Y., Liu, X., & Jia, Y. Hand-gesture based text input for wearable computers. *ICVS '06*. 8-14.
13. 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.
14. MacKenzie, S. Mobile text entry using three keys. *CHI '02*. 27-34.
15. MacKenzie, I. S., & Soukoreff, R. W. *Phrase sets for evaluating text entry techniques*. *CHI '03 EA*. 754-755.
16. 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.
17. Ni, T., & Baudisch, P. Disappearing mobile devices. *UIST '09*. 101-110.
18. 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.
19. Partridge, K., Chatterjee, S., Sazawal, V., Borriello, G., & Want, R. TiltType: accelerometer-supported text entry for very small devices. *UIST '02*. 201-204.
20. Perlin, K. Quikwriting: continuous stylus-based text entry. *UIST '98*. 215-216.
21. Rico, J., & Brewster, S. Usable gestures for mobile interfaces. *CHI '10*. 887-896.
22. Rosenberg, R., & Slater, M. The chording glove: a glove-based text input device. *ToSMC*. 1999, 29(2), 186-191.
23. San Agustin, J., Skovsgaard, H., Hansen, J. P., & Hansen, D. W. Low-cost gaze interaction: ready to deliver the promises. *CHI '09 EA*. 4453-4458.
24. Thomas, B., Tyerman, S., and Grimmer, K. Evaluation of text input mechanisms for wearable computers. *Virtual Reality* 3, no. 3 (1998). 187-199.
25. Ward, D. J., Blackwell, A. F., & MacKay, D. J. Dasher - a data entry interface using continuous gestures and language models. *UIST '00*. 129-137.
26. Wigdor, D., & Balakrishnan, R. TiltText: using tilt for text input to mobile phones. *CHI '03*. 81-90.
27. Wigdor, D., & Balakrishnan, R. A comparison of consecutive and concurrent input text entry techniques for mobile phones. *CHI '04*. 81-88.
28. 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.
29. 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.
30. Zhao, S., Agrawala, M., & Hinckley, K. Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. *CHI '06*. 1077-1086.
31. Zhao, S., & Balakrishnan, R. Simple vs. compound mark hierarchical marking menus. *UIST '04*. 33-42.