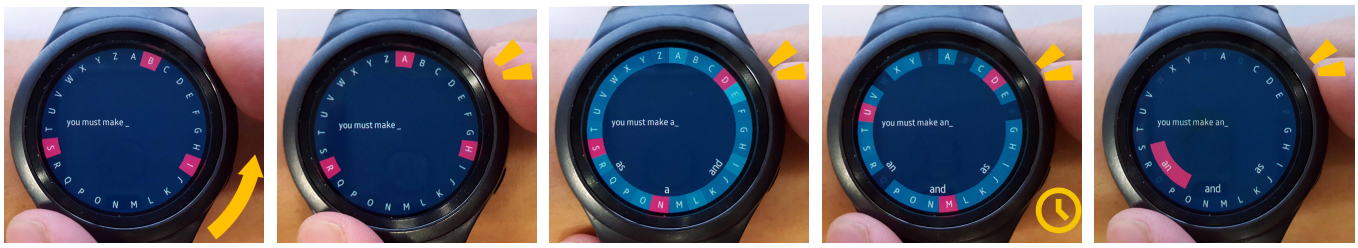
# COMPASS: Rotational Keyboard on Non-Touch Smartwatches

**Xin Yi[1], Chun Yu[1†], Weijie Xu[1], Xiaojun Bi[2], Yuanchun Shi[1]**
[1]Key Laboratory of Pervasive Computing, Ministry of Education
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
[2]Department of Computer Science, Stony Brook University, Stony Brook, NY, United States
{yix15,xu-wj13}@mails.tsinghua.edu.cn {chunyu,shiyc}@tsinghua.edu.cn xjunbi@gmail.com




(a) (b) (c) (d) (e)

Figure 1: A storyboard illustrating a user entering "an" with *COMPASS*. (a) The keyboard shows three cursors at initial locations, the user rotates the bezel counterclockwise to hit 'a' with a nearby cursor. (b) The user presses the physical button to select 'a'. (c) Upon selection, the keyboard changes the location of the cursors so as to minimize the expected rotational distance for the next key. As there is already a cursor on 'n', the user presses again to select 'n' without rotating. The color of each key indicates the probability of it being the next selection. (d) The user long-presses the physical button to enter word selection mode. Dimmed keys are not possible for the next selection. (e) The user presses the button to select "an".

## ABSTRACT
Entering text is very challenging on smartwatches, especially on non-touch smartwatches where virtual keyboards are unavailable. In this paper, we designed and implemented *COMPASS*, a non-touch bezel-based text entry technique. *COMPASS* positions multiple cursors on a circular keyboard, with the location of each cursor dynamically optimized during typing to minimize rotational distance. To enter text, a user rotates the bezel to select keys with any nearby cursors. The design of *COMPASS* was justified by an iterative design process and user studies. Our evaluation showed that participants achieved a pick-up speed around 10 WPM and reached 12.5 WPM after 90-minute practice. *COMPASS* allows users to enter text on non-touch smartwatches, and also serves as an alternative for entering text on touch smartwatches when touch is unavailable (e.g., wearing gloves).

† denotes the corresponding author.



*CHI 2017*, May 06-11, 2017, Denver, CO, USA
© 2017 ACM ISBN 978-1-4503-4655-9/17/05...$15.00
DOI: http://dx.doi.org/10.1145/3025453.3025454

## ACM Classification Keywords
H.5.2. Information Interfaces and Presentation (e.g. HCI): User Interfaces; Input devices and strategies (e.g., mouse, touchscreen)

## Author Keywords
Text entry; smartwatch; multiple cursors; circular keyboard; non-touch.

## INTRODUCTION
Smartwatches have become increasingly popular in the Post-PC computing era. They are easy-to-access, always-on and could be fashionable decorations if well designed. To meet the diverse market demand, a variety of smartwatches have been designed and manufactured, which can be classified as either *touch* or *non-touch* smartwatches. The former has a touch-sensitive screen and supports touch interaction (e.g., Apple Watch [3]), while the latter is equipped with a non-touch display in order to reduce power consumption and to be more affordable. For example, the Pebble Time watch [6] uses a non-touch e-ink color display, with an up to 7 day battery life.

To better serve users, a smartwatch is enhanced with more and more interactive features. Entering text, which is one of the most performed tasks in mobile computing, is one of the most

desired features on smartwatches. However, it has remained a big challenge, especially for non-touch smartwatches, on which the typical virtual keyboard is not supported.

One approach of entering text on non-touch smartwatches is to use speech (e.g. MOTO 360 [5]). However, it suffers from two major problems: 1) it is socially inappropriate to input with voice in some situations (e.g., at meetings or classrooms); 2) speech-based input is error-prone especially in noisy environments. Although a considerable amount of effort has been investigated in improving smartwatch text entry (e.g. [12, 16, 22, 27, 34]), very little of them is for non-touch smartwatches. In this paper, we aim to address this important but overlooked problem.

In this paper, we designed and implemented *COMPASS*, a bezel-based, multi-cursor text entry technique for non-touch smartwatches. It works for non-touch smartwatches, and also for touch smartwatches when touch input is unavailable (e.g., wearing gloves). To enter text, users rotate the bezel to hit keys on the circular keyboard with a cursor. To minimize rotational distance, the location of each cursor is dynamically optimized after each key selection according to the probability of next keys needed.

We followed an iterative design process in designing *COMPASS*. In the first user study, we compared the text entry performance and user preference with different cursor placement strategy and number of cursors. Results showed that a dynamic strategy with three cursors was optimal. We then further improved this technique according to the user behavior data, and evaluated its usability in the second user study. Users achieved a pick-up speed of 9.3 WPM, and after 90 minutes of practice, they achieved 12.5 WPM without sacrificing accuracy. Expert users even reached 15.4 WPM in the last block. We concluded that *COMPASS* is a promising text entry method for non-touch smartwatches and other rotational interfaces.

## RELATED WORK

### Interaction Techniques for Smartwatches

A number of text entry techniques have been proposed to faciliate text entry on touch smartwatches. Some techniques achieved accurate character-level input by iterative selection (e.g. ZoomBoard [27], SwipeBoard [12] and SplitBoard [19]) or using a call-out to show the finger occluded area (e.g. ZShift [22]). The text entry speed reached 9.3 WPM [27], 19.6 WPM [12], 15 WPM [19] and 9.1 WPM [22] respectively. Alternatively, some techniques provided efficient word-level input by incorporating input decoders. (e.g. VelociTap [34] and WatchWriter [16] reached 41 WPM and 22 WPM respectively). Noticeably, all these techniques adopted a QWERTY keyboard. However, the limitation of touchscreen-based techniques lies in a tradeoff between the screen occupation and input precision. Besides, touchscreens is even unavailable in certain scenarios (e.g. wearing gloves). Funk et al. [14] used a touch-sensitive wristband to support text entry on smartwatches. However, the speed reached only about 3 WPM.

To address this problem, some researchers sought to enhance the interaction capability for non-touch smartwatches. For example, Arefin et al. [8] designed a set of non-touchscreen gestures to enable interaction with the smartwatch in mid-air. Abracadabra [17] used magnetometers to enable wireless, unpowered and high fidelity finger input for smartwatches. And Serendipity [35] used the motion sensors of the smartwatch to sense finger gesture. Plaumann et al. [28] used a bezel to support list selection based on a circular alphabetical menu. Kerber et al. [20] compared bezel interaction with digital crown and touch, where the bezel showed worse performance than the latter two. Noticeably, none of these techniques has been applied to the scenario of text entry. In this regard, we seek to explore the possibility of text entry on off-the-shelf non-touch smartwatches in this paper.

### Circular Keyboard Techniques

Besides conventional QWERTY layout, there are also some keyboards that use a circular layout. Although less common, the circular layout poses some advantages in certain scenarios (e.g. rotational interfaces [29], personal area on tabletop [18], or to minimize the travelling distance [25, 32]).

Cirrin [25] used a circular keyboard to support pen input using word-level gestures. The order of the characters were optimized to minimize the stroke distance. To reduce the difficulty of selecting a letter, Cechanowicz et al. [11] modified Cirrin by expanding the characters as the stylus approached it. However, the text entry performance dropped due to a higher error rate. These two keyboards optimized the order of the characters for efficiency. However, at the same time, this would result in more cognitive load for users, which may limit the text entry speed.

Alternatively, some techniques used an alphabetic ordered list of characters to aim the pick-up speed. BubbleCircle [18] used a circular keyboard on a tabletop displays, and dynamically adjusted the key sizes according to user's input. Shoemaker et al. [32] proposed a circular keyboard for text entry on large wall displays. User moves the pointer to highlight the desired character and presses a button to select it. TUP [29] places the characters at fixed positions on the wheel, and user press a select key to select the highlighted character.

### Multi-Cursor Techniques for Target Acquisition

Using multiple cursors to assist target acquisition has been extensively studied in the scenario of Graphical User Interface (GUI). These techniques usually placed multiple cursors on the screen, which move synchronously following the mouse. A user can point to a target use the cursor that is closest to it. This posed an inherent advantage that the travel distance between targets in cursor-based interfaces can be significantly reduced. Ninja Cursors [21] uniformly distributed multiple cursors on the screen, and used a waiting queue algorithm to resolve the ambiguity. Similarly, satellite cursor [36] places a cursor in the vicinity of every target, and used an "aggregate and expand algorithm" to avoid ambiguity.

Some researchers incorporate eye-gaze to help resolve the cursor ambiguity when multiple targets were selected at the same time. Raiha et al. [31] augmented the Ninja Cursors by using eye-gaze to select the active cursor. Similarly, the Rake Cursor [10] also used a grid of cursors controlled by a mouse, and the selection of the active cursor by gaze. However, the

active cursor can only change when a mouse motion event occurs.

## COMPASS: DESIGN AND IMPLEMENTATION

We implemented *COMPASS* on a Samsung Gear S2 smartwatch (see Figure 1). The watch has a 1.2" round screen, a rotational bezel and two physical buttons. The rotational bezel can detect discrete rotation events. According to our test, the rotation precision is about 24 $levels/360^\circ$. As we focus on non-touch smartwatches, we did not leverage the touchscreen of this watch. In practice, one can also implement *COMPASS* on other hardware as long as it supports rotation and press interaction (e.g. the digital crown on Apple Watches [3] and the BMW iDrive controller [4]).

### Keyboard Layout

We adopted a circular layout that best fits the rotation interaction paradigm. To help users locate the target keys quickly, the character keys were organized in alphabetical order [29]. Furthermore, we flipped the characters on the lower half of the keyboard inside out ('H' to 'S') to help users visually recognize these characters (see Figure 1a).

An advantage of the circular layout is that different from conventional QWERTY keyboards (e.g. [16]) and T9 keyboards (e.g. the system keyboard on Samsung Gear S2 smartwatch), it allows the remaining screen area to be in a round shape. Therefore, the screen contents can be scaled to fit in the inner area without changing the look-and-feel. We can also use a transparent keyboard to further avoid shrinking the contents.

### Interaction Design

The design goal of *COMPASS* is to enable text entry on smartwatches without using the touchscreen. To achieve this, we displayed cursors along the circular keyboard. During text entry, users rotate the bezel of the smartwatch to move the cursors, and press the physical button to select characters and words [29, 32]. Compared to touchscreen-based keyboards, this design avoided the fat-finger problem, and provides tactile feedback during rotation. Considering that long rotations were slow and hard to perform, we employed a multi-cursor interaction paradigm, where multiple cursors move and select the characters simultaneously according to user's control. Figure 1 shows the storyboard illustrating a user entering words using *COMPASS*. The interaction is as follows:

1. Users rotate the bezel to hit the target key with any nearby cursors, and press the physical button on the side of the smartwatch to select the character.
2. Upon selection, *COMPASS* showed three candidate words according to the current input on the bottom of the inner area.
3. Users long press the physical button to switch to word selection mode, where there is only one cursor. We empirically set the time threshold of long press to be 250 ms according to our pilot study.
4. If there is only one candidate word left, the word would be automatically entered. Otherwise, users rotate the bezel to hit the target word with the cursor, and then press the button again to confirm the selection. *COMPASS* automatically appends a space after the selected word.
5. During typing, users can flick their wrists to delete the last entered word after selection, or the last character during entering a word.

### Flicking Gesture

During text entry, we designed wrist flick as an action to trigger deletion. Flicking is a system gesture for Android Wear [7] and Tizen OS, which is natural and easy to perform.

We recognized the flick gestures based on the gyroscope data from the Samsung Gear S2 smartwatch, which reports the rotation of the wrist at about 100Hz. Figure 2 illustrates the signal during a typical flicking gesture. According to empirical evidence, we used a 500 ms time window with two thresholds ($\pm 550^\circ/s$) on the X dimension data to recognize the gesture.

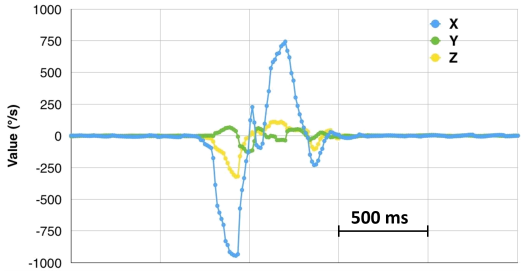



Figure 2: Illustration of the gyroscope data during a flicking gesture.

### Prediction Algorithm and Language Model

As with any modern smart keyboards, we employed a prediction algorithm in *COMPASS* to resolve the ambiguity in user's input. We employed Goodman et al.'s Bayesian model [15] to predict the target word. Given user's input $I$, it calculates the probability of a word $W$ in a predefined dictionary as:

$$P(W|I) \propto P(I|W) \times P(W) \tag{1}$$

As *COMPASS* used multiple cursors to select characters simultaneously, we have

$$I = \begin{pmatrix} I_{11} & I_{12} & \cdots & I_{1n} \\ I_{21} & I_{22} & \cdots & I_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ I_{N1} & I_{N2} & \cdots & I_{Nn} \end{pmatrix} \tag{2}$$

where $N$ is the number of cursors, and $n$ is the length of the input. As with many smart keyboard techniques, we assumes that users generate no insertion or omission errors, and we treat each input point independently (e.g. [13, 15]), therefore:

$$P(I|W) = \prod_{i=1}^{n} P(I_{\cdot i}|W_i) \tag{3}$$

where $I_{\cdot i}$ refers to the $i$th column of $I$, and $W_i$ is the $i$th character of $W$. Meanwhile, we set

$$P(I_{\cdot i}|W_i) = \begin{cases} 1 & \text{if } \exists 1 \le j \le N \quad \text{s.t.} \quad I_{ji} = W_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This assumption is based on the observation in pilot study that, users rarely makes mistake when selecting characters. In this way, *COMPASS* behaves similar with an ambiguous keyboard (e.g. T9 keyboard).

We used the top 15,000 words as well as their corresponding frequency in the *American National Corpus* [2] as our corpus. According to Nation et al. [26], this would be sufficient to cover over 95% of the common English words.

### Visual Hint

When entering text using *COMPASS*, one of the critical factors that affect the text entry performance is the ease to visually acquire the target key. In this regard, we designed visual cues to help users find their target keys during text entry.

Assuming the user has generated the input $I$ (see Equation 2), we denote $S(I)$ as the set of all words $W$ in the dictionary that $P(I|W_1W_2...W_n) \neq 0$. In other words, $S(I)$ contains all words whose prefix matches the input $I$. Now, for each character $c$ in the alphabet, the probability of it being the following character can be calculated as:

$$P(c) = \frac{\sum_{W \in S(I) \wedge W_{n+1}=c} P(W)}{\sum_{W \in S(I)} P(W)} \quad (5)$$

Based on $P(c)$, we designed two kinds of visual cues, as showed in Figure 3. First, the keys whose probability was zero would be dimmed to avoid distracting the user (e.g. 'F' and 'T'). Second, we adjusted the brightness of the remaining keys according to their probability. The higher the probability is, the lighter the key color would be (e.g. 'P' vs. 'A'). During text entry, the visual cues automatically updates upon each character selection. According to our interview, this design indeed helped users to find the target keys (especially those with high probability) more easily.


Figure 3: Visual cues to help users find the target keys. Impossible keys are dimmed, while others are highlighted according to their possibility.

### Cursor Placement Strategy

A key problem of *COMPASS* arise from the multi-cursor paradigm. On the one hand, if the relative location of the cursors keep fixed, users may be able to program their rotation behavior in advance. On the other hand, dynamically optimizing the position of the cursors could reduce the rotation distance, thus potentially increasing the text entry speed. However, this dynamic interface may introduce more cognitive load for users at the same time. Therefore, whether or not to employ a dynamic cursor placement strategy is worth exploring.

To address this problem, we designed two variations of *COMPASS*: *S-COMPASS* (*S* for *Static*) and *D-COMPASS* (*D* for *Dynamic*). The two techniques share the same design except for the cursor placement strategy.

#### S-COMPASS

The design goal of *S-COMPASS* is to keep a static cursor placement to reinforce users' motor programming ability, but at the cost of a presumably longer rotation distance. Consider that we used $N$ cursors, it is obvious that the cursors should be placed evenly along the keyboard to minimize the expected rotation distance. Given the static cursor placement strategy, *S-COMPASS* functions similar with an ambiguous keyboard, where there are $N$ characters on each "key".

#### D-COMPASS

The design goal of *D-COMPASS* is to dynamically adjust the position of the cursors according to user's input, such that the expected distance of the next rotation is minimized. During typing, the location of the cursors changed upon each character selection. Compared with *S-COMPASS*, *D-COMPASS* could potentially increase the text entry speed by reducing the rotation distance.

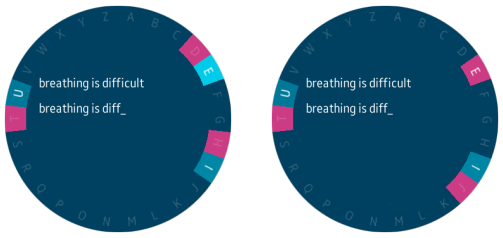
Each time the user selects a character, the algorithm searches all the $C_{26}^N$ possible cursor locations, and selects the one with the lowest *Expected Next Rotation Distance (ENRD)*, which was defined as:

$$ENRD = \sum_{c \in \chi} dis(c) \times P(c) \quad (6)$$

where $\chi$ is the set of all 26 characters, $P(c)$ was calculated according to Equation 5, and $dis(c)$ is the rotation distance that needed to hit key $c$ with the closest cursor. For example, in Figure 3, $dis(E) = 2$.

Ideally, the above optimization would minimize the expected distance of next rotation. However, as Equation 6 is a weighted distance, in practice, the optimization tended to place the cursors around keys with highest frequencies, and cause these keys to be selected at the same time (see Figure 4a). As a result, the number of remaining keys after each selection would not drop as fast as expected. To alleviate this problem, we adopted an intuitive method that prevents high-frequency keys to be selected at the same time: Instead of considering all $C_{26}^N$ possibilities, we pruned the candidates that would cause any two of the $N$ characters with the highest $P(c)$ to be selected at the same time (see Figure 4b).

This heuristic restriction helps the optimization procedure to avoid grouping high probability keys together. With a small sacrifice of the rotation distance, it may decrease the entropy of the remaining text, and thus increased the overall typing performance. According to our pilot study, optimizing rotation distance was also more perceived and appreciated by users than optimizing ambiguity.




(a) (b)

Figure 4: Example of cursor placement (a) before applying restriction, 'U', 'E' and 'I' would be selected at the same time (b) after applying restriction, 'U', 'E' and 'I' could not be selected at the same time.

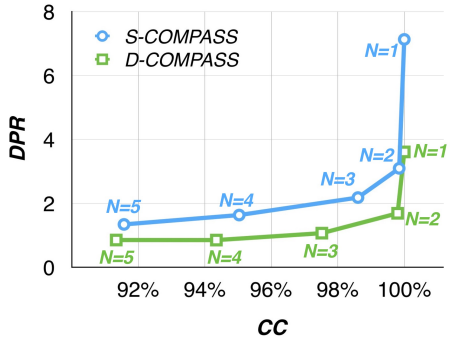



Figure 5: Simulation result. *SRD* and *CC* for each *N* and technique.

### Number of Cursors

It is intuitive that the number of cursors (*N*) is crucial for the performance of every multi-cursor techniques [21]. On the one hand, the more cursors we use, the shorter distance users would have to rotate during text entry. On the other hand, too many cursors would result in two problems: 1) as multiple characters are selected by the cursors simultaneously, this would let to input ambiguity 2) too many cursors may spam the user interface, causing visual and cognitive burden for users [30]. Therefore, we conducted a simulation to determine the range of possible *N* that is applicable. Metrics include *rotation distance*, which is indicative of the text entry speed, and *input ambiguity*, which would affect the input accuracy.

#### *Simulation Design*

We defined two metrics: 1) *Distance Per Rotation (DPR)*, which is average distance for each rotation when entering a word, normalized by key size. A greater *DPR* suggests longer distance for each rotation, and presumably longer input time. 2) *Candidate Coverage (CC)* measures the ratio of the words in the dictionary that could appear in the top three candidates given user's input. A higher *CC* implies that the technique is more accurate in input disambiguation.

We tested all words in the dictionary for both *S-COMPASS* and *D-COMPASS*. For each word, we simulated user's input by assuming that the user would always use the nearest cursor to hit the keys. We tested the number of cursors (*N*) from 1 to 5. For each level of *N*, we measured the average *DPR* of each word as well as the *CC*.

#### *Results*

Figure 5 shows the simulation results. As expected, for *S-COMPASS* and *D-COMPASS*, *DPR* and *CC* both drop monotonously with increasing *N*, suggesting a tradeoff between rotation distance and input ambiguity.

Generally, *S-COMPASS* yielded a slightly higher *CC*, but a significantly higher *DPR* than *D-COMPASS*. Interestingly, the *DPR* of *S-COMPASS* ranged from 1.34 to 3.10 for $N \geq 2$, but dynamically jumped up to 7.13 for $N = 1$, suggesting a much further rotation distance. Meanwhile, $N = 5$ only decreased *DPR* by 0.29 comparing with $N = 4$, but decreased *CC* by 3%. Therefore, we considered the applicable number of cursors for *S-COMPASS* to be from 2 to 4.

The results on *D-COMPASS* are similar. The *DPR* ranged from 0.85 to 1.69 for $N \geq 2$, and jumped up to 3.61 for $N = 1$. Meanwhile, $N = 5$ yield the same *DPR* with $N = 4$, but decreased *CC* by 3%. Therefore, the applicable number of cursors for *D-COMPASS* was also from 2 to 4.

## EXPERIMENT 1: COMPARING DIFFERENT DESIGNS

So far, we have proposed several design alternatives for *COMPASS*, each representing a tradeoff between certain design goals. In this section, we conducted a user study to compare the performance and user preference of these designs with two main goals: 1) determine the optimal number of cursors for both *S-COMPASS* and *D-COMPASS* in terms of text entry performance and user preference; 2) compare the performance and subjective feedback of *S-COMPASS* and *D-COMPASS*.

### Participants

We recruited 12 participants from the campus (8 male, 4 female), with an average age of 22.7 (SD = 2.0). None of them had used the Samsung Gear S2 smartwatch before. Each participant was compensated $12.

### Procedure

We used a within-subjects, two-factor design. One factor is *Technique* with two levels (*S-COMPASS* and *D-COMPASS*), and the other is *number of cursors* (*N*) with three levels (2, 3 and 4). Participants were seated during the experiment. They were asked to complete two sessions of text entry tasks, corresponding to *S-COMPASS* and *D-COMPASS* respectively. In each session, they completed three blocks of tasks, each corresponds to a level of *N*. The order of different *Technique* and *N* was counterbalanced. In each block, they were first allowed three minutes to familiarize themselves with the technique. They then entered 5 phrases randomly sampled from the Mackenzie and Soukoreff phrase set [24]. A two-minute break was enforced between blocks. And after the experiment, questionnaires and interviews were carried out to gather their subjective feedback.

Figure 6 shows a screenshot of the experiment platform. The task phrase was showed in the middle of the screen, with the

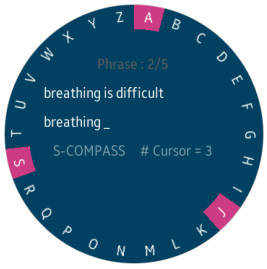



Figure 6: Experiment platform.

entered text showed below it. Participants were instructed to type "as quickly and as accurately as possible". They were free to correct the errors or leave them uncorrected. After finishing each phrase, they swipe right on the touchscreen to continue to the next phrase.

## Results

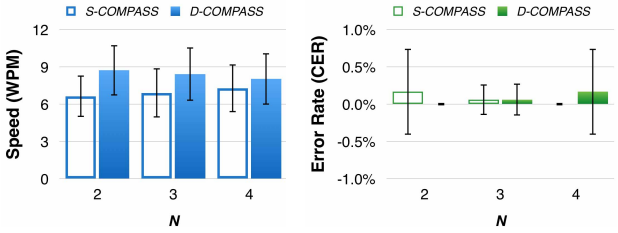
Across all participants, we gathered 2 Techniques × 3 N × 5 phrases × 12 participants = 360 phrases.

### *Speed*

We calculated the text entry speed similar with Mackenzie [1]:

$$WPM = \frac{|W|-1}{T} \times 60 \times \frac{1}{5} \quad (7)$$

where $|W|$ is the length of the final transcribed string, and $T$ is the elapsed time in seconds from the first press to the selection of the last word. Figure 7a shows the text entry speed for different techniques and $N$. *RM-ANOVA* found *D-COMPASS* to be significantly faster than *S-COMPASS* ($F_{1,11} = 100.7, p < .0001$).




(a) (b)

Figure 7: (a) Text entry speed and (b) error rate for different techniques and $N$, black bar indicated one standard deviation.

For *S-COMPASS*, WPM slightly increased with increasing $N$, with the average speed being 6.6 (SD = 1.7), 6.9 (SD = 2.0) and 7.3 (SD = 1.9) for $N = 2$, 3 and 4 respectively. This confirmed that with more cursors, the rotation distance could be shortened. On contrary, for *D-COMPASS*, WPM slightly drops with increasing $N$, with the average speed being 8.7 (SD = 2.0), 8.4 (SD = 2.2) and 8.0 (SD = 2.1) for $N = 2$, 3 and 4 respectively. We speculate that this was due to the extra time to locate the cursors after adaptation. A significant effect of $N$ was found on WPM for both *S-COMPASS* ($F_{2,22} = 3.42, p = .05$) and *D-COMPASS* ($F_{2,22} = 7.04, p < .01$).

### *Error Rate*

We calculated error rate using CER [33], Figure 7b shows the error rate for different techniques and $N$. As with previous findings, users tended to leave few errors in the final transcribed string (e.g. [23, 33]), with the average error rate for all six conditions under 0.17%. As expected, no significant effect of *Technique* was found on CER ($F_{1,11} = 0.00, p = .99$). Moreover, no significant effect of $N$ on CER was found for either *S-COMPASS* ($F_{2,22} = 0.66, p = .52$) or *D-COMPASS* ($F_{2,22} = 1.00, p = .38$).

### *Subjective Ratings*

We collected participants' subjective ratings for each condition using a 5-point Likert scale questionnaire. Dimensions included perceived speed, perceived accuracy and overall preference. Table 1 shows the average ratings. *D-COMPASS* was showed to significantly outperformed *S-COMPASS* in terms of all dimensions (perceived speed: $F_{1,11} = 12.2, p < .01$, perceived accuracy: $F_{1,11} = 11.3, p < .01$, overall preference: $F_{1,11} = 26.2, p < .001$).

| Technique | S-COMPASS | | | D-COMPASS | | |
|---|---|---|---|---|---|---|
| *N* | 2 | 3 | 4 | 2 | 3 | 4 |
| Perceived Speed | 2.4 (0.9) | 3.2 (0.8) | 3.7 (0.6) | 3.5 (1.1) | 4.3 (0.6) | 4.2 (0.6) |
| Perceived Accuracy | 3.6 (1.1) | 4.0 (0.8) | 3.8 (0.7) | 4.2 (0.8) | 4.5 (0.7) | 4.0 (0.8) |
| Overall Preference | 2.2 (1.1) | 3.2 (0.9) | 3.3 (0.8) | 3.3 (1.0) | 4.5 (0.5) | 3.8 (0.7) |

Table 1: Average ratings for different conditions, standard deviations showed in parenthesis. 5 means the most positive and 1 means the most negative.

For both *S-COMPASS* and *D-COMPASS*, Friedman test found a main effect of $N$ on perceived speed ($\chi^2(2) = 17.1, p < .001$ and $\chi^2(2) = 7.31, p < .05$) and overall preference ($\chi^2(2) = 8.34, p < .05$ and $\chi^2(2) = 10.3, p < .01$). However, no significant effect of $N$ was found on perceived accuracy ($\chi^2(2) = 1.92, p = .38$ and $\chi^2(2) = 2.61, p = .27$). Noticeably, for *S-COMPASS*, $N = 4$ yield the highest average score in perceived speed and overall preference, while for *D-COMPASS*, $N = 3$ yield the highest average score among all conditions.

### *Interaction Statistics*

We looked more into users' text entry behavior by analyzing the interaction statistics. Figure 8 shows the average *Distance per Rotation (DPR)* for each condition. Both *Technique* ($F_{1,11} = 1957, p < .0001$) and $N$ ($F_{2,22} = 1132, p < .0001$) was found to yield significantly effect on *DPR*. Consistent with the simulation result, *DPR* decreased with increasing $N$ for both *S-COMPASS* and *D-COMPASS*. And for all $N$, the *DPR* for *D-COMPASS* was less than 50% of that for *S-COMPASS*. This confirmed that *D-COMPASS* effectively reduced the rotation distance than *S-COMPASS*.

We then analyzed which interactions takes up the most amount of time during text entry. Table 2 shows the top 3 interactions with the highest time ratio. Noticeably, for *S-COMPASS*, *Rotate-Rotate* takes up 44.6% of all interactions, which corresponds to Figure 8 that users has to rotate a relatively long distance on average. On contrast, for *D-COMPASS*, the most time-consuming interaction was *Press-Press*, which happens

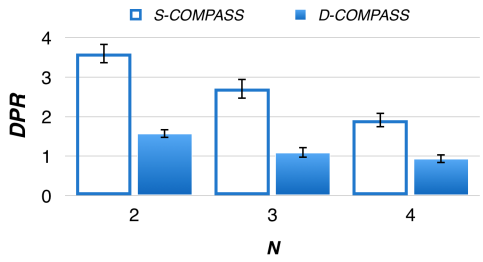



Figure 8: Average *DPR* for each condition, black bar indicates one standard deviation.

when users hit a target without rotating. This highlights that *D-COMPASS* effectively decreased the rotation frequency.

| Interaction | Average Time | Frequency | Time Ratio |
|---|---|---|---|
| Rotate - Press | 0.68 s | 21.4% | 29.0% |
| Press - Rotate | 0.84 s | 17.2% | 28.7% |
| Rotate - Rotate | 0.22 s | 44.6% | 19.0% |

(a)

| Interaction | Average Time | Frequency | Time Ratio |
|---|---|---|---|
| Press - Press | 0.92 s | 19.5% | 27.0% |
| Press - Rotate | 1.13 s | 12.4% | 21.0% |
| Rotate - Press | 0.69 s | 18.6% | 19.1% |

(b)

Table 2: Interaction statistics for (a) *S-COMPASS* and (b) *D-COMPASS*, merged from $N = 2, 3$ and 4. Shows the top 3 with the highest time ratio.

## Discussion

### *Number of Cursors*

The number of cursors seemed not to affect text entry performance significantly. For *S-COMPASS*, WPM slightly increased with $N$, while for *D-COMPASS*, WPM slightly drops with $N$. Although these differences were statistically significant, in both cases, the amplitude of change was very small ($\pm$0.6 WPM). Meanwhile, $N$ did not significantly affect the error rate or perceived accuracy.

However, subjective preference changed dramatically with $N$. For *S-COMPASS*, $N = 4$ yielded the highest score in perceived speed and overall preference, and for *D-COMPASS*, $N = 3$ yielded the highest score in all dimensions, which are all significant. Therefore, we considered 4 and 3 to be the optimal number of cursors for *S-COMPASS* and *D-COMPASS* respectively.

### *S-COMPASS vs. D-COMPASS*

Interestingly, both objective and subjective results favors *D-COMPASS* as opposed to *S-COMPASS*. The text entry speed of *D-COMPASS* was 32%, 22% and 15% higher than that of *S-COMPASS* with increasing $N$ (see Figure 7a), and there was no significant difference in error rate. Due to the dynamic cursor placement strategy, the rotation distance for *D-COMPASS* was less than 50% of that for *S-COMPASS* (see Figure 8).

In terms of subjective feedback, *D-COMPASS* significantly outperforms *S-COMPASS* in all dimensions (perceived speed, perceived accuracy and overall preference). During interview, most of the participants (11/12) preferred *D-COMPASS* to *S-COMPASS*, only 1 participant preferred *S-COMPASS*. Therefore, we considered *D-COMPASS* to be more applicable.

### *Improving COMPASS*

So far, we have showed that in terms of text entry performance and user preference, *D-COMPASS* with three cursors was the best technique among the six alternatives. However, the interaction statistics and subjective feedback suggested that there are still space for further improving the performance.

Table 2b shows that for *D-COMPASS*, *Press-Press* was the most frequent interaction, and took the most amount of time. This usually happens when there are only a few possible keys left. For example, when entered "welc" for "welcome", users has to press three times without rotating for 'o', 'm' and 'e'. And in our interview, 6/12 participants suggested that the text entry process could be accelerated in this case. To this end, we improved *D-COMPASS* by incorporating an auto-completion algorithm.

We implemented the auto-completion algorithm using a similar method as in the Android keyboard [9]. When calculating $P(I|W)$, we considered all $W$ whose length are not less than $I$, and augmented Equation 3 to the formula below:

$$P(I|W) = \prod_{i=1}^{n} P(I_i|W_i) \times \alpha^{m-n} \quad (8)$$

where $m$ is the length of $W$ ($m \geq n$). Here, $\alpha$ could be interpreted as the penalty to prevent long, high-frequency words from dominating short, low-frequency words. For example, if the frequency of "and" is higher than "an", "and" would always appear in front of "an". We tested alpha ranging from 0.0 to 1.0 in simulation, and set $\alpha = 0.7$ that yields the best compromise between aggressiveness and candidate coverage.

## EXPERIMENT 2: PERFORMANCE EVALUATION

In this section, we conducted a second user study to evaluate the performance of the improved *COMPASS* technique. Henceforth, we referred to *COMPASS* as the *D-COMPASS* with three cursors and auto-completion algorithm. Participants were asked to perform text entry tasks using the technique. We were interested in the pick-up usability of *COMPASS*, and the text entry performance that users could achieve with practice.

### Participants

We recruited another 10 participants from the campus, with an average age of 22.0 (SD = 1.5). None of them has experienced a rotational interface before (e.g. the smartwatch we used in this experiment). Each participant was compensated $30.

### Procedure

We used the same experiment platform as in the previous study (see Figure 6). Participants were seated in a chair during the experiment. They were first allowed up to three minutes to

familiarize themselves with the technique. They then completed 8 blocks of text entry tasks. In each block, they entered 10 phrases randomly chosen from the Mackenzie and Soukoreff phrase set [24]. A three-minute break was enforced between blocks. Participants were instructed to type "as fast and as natural as possible". They were free to correct the input or leave the error uncorrected. Each block took about eight minutes, and totally, the tasks took about 90 minutes. After the final block, we carried out interviews to gather their subjective feedback.

## Results

Across all participants, we gathered 8 blocks $\times$ 10 phrases $\times$ 10 participants $=$ 800 phrases.

### *Text Entry Speed*

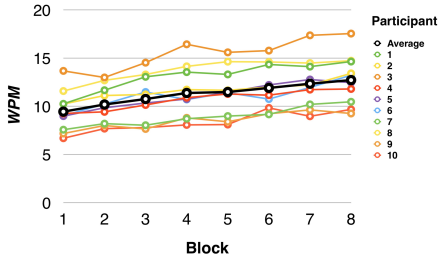



Figure 9: Text entry speed in each block, black line shows the average speed across participants.

Figure 9 shows each participant's speed and the average speed in each block. Participants' speed increased monotonously with block, suggesting a consistent learning trend. The average speed was 9.3 WPM (SD = 2.2) in block 1 and 12.5 WPM (SD = 2.5) in block 8, which was an increase of 34%. The top three participants with highest speed achieved an average of 11.9 WPM in the first block, and 15.4 WPM in the last block. A significant effect of *Block* was found on text entry speed ($F_{7,63} = 48.0, p < .0001$).

Notice that in Experiment 1, participants only reached 8.4 WPM when using *D-COMPASS* with three cursors. This improvement suggested that auto-completion may be helpful for increasing the performance of *COMPASS*. Although this speed was lower than word-level techniques using a miniature QWERTY keyboard [16]. We consider it still important as *COMPASS* used only rotation and button pressing, which is inevitably slower than using touchscreens. Moreover, WPM seemed not to stop increasing in block 8, suggesting that users may achieve higher text entry speed with more practice.

### *Error Rate*

Figure 10 shows each participant's error rate and the average error rate in each block. Consistent with the result in Experiment 1, participants left few errors in the final transcribed phrase. There were an average of 4.6 (SD = 3.3) deletions for each participant in each block. In all 800 phrases, only 4 phrases yielded a CER $> 0$. As expected, no significant effect of *Block* was found on error rate ($F_{7,63} = 0.81, p = .58$).

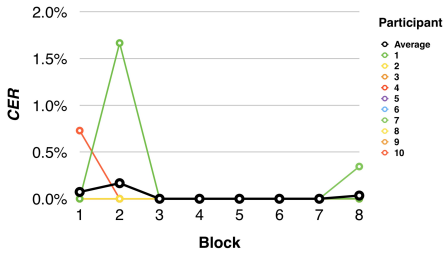



Figure 10: Error rate in each block, black line shows the average speed across participants.

### *Interaction Statistics*

To examine whether user's interaction behavior evolved with practice, we analyzed the interaction statistics. We were interested in: 1) Time taken to visually acquire the target key, which is indicative of the familiarity of the keyboard layout; 2) the usage of the auto-completion feature.

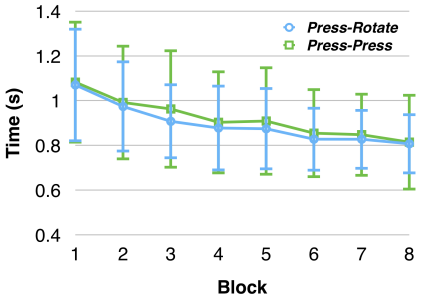



Figure 11: Average *Press-Rotate* and *Press-Press* time in each block, bar shows one standard deviation.

Ideally, the time taken to locate the target key should be measured by analyzing the eye-gaze data. Considering this impractical, we resort to use two other metrics that is measurable: the average time of *Press-Rotate* and *Press-Press*. These two values measure the time elapse from the update of the cursors to the next interaction action, which could approximate the time taken to find the target key.

Figure 11 shows the average interaction time for these two interactions in each block. Generally, the average time for *Press-Rotate* and *Press-Press* was very close, confirming the consistency of the two metrics. Both time dropped with block, suggesting that with more practice, users could spent less time searching for the target key. In the first block, the average time for *Press-Rotate* and *Press-Press* were 1.07 s (SD = 0.26) and 1.08 s (SD = 0.27) respectively. And in block 8, they dropped to 0.81 s (SD = 0.14) and 0.81 s (SD = 0.22) respectively. Significant effect of *Block* was found on both times ($F_{7,63} = 14.9, p < .0001$ and $F_{7,63} = 11.8, p < .0001$).

To quantify users' adoption of the auto-completion feature, we defined *SPC (Selection Per Character)* as a metric, which can be calculated as:

$$SPC = \frac{|N_{selection}|}{|TranscribedText|} \quad (9)$$

where $|N_{selection}|$ is the total number of *Press* and *Long Press*, and $|TranscribedText|$ is the length of the final transcribed string. A lower SPC indicates that the user used the auto-completion feature more frequently, thus spent less effort on selecting characters and words.

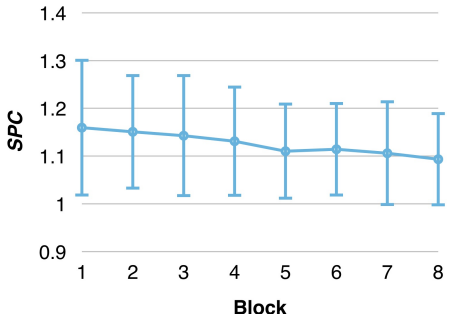



Figure 12: Average SPC in each block, bar shows one standard deviation.

Figure 12 shows the average SPC in each block. SPC dropped with block, suggesting that with more practice, participants used the auto-completion feature more frequently. In the first block, the average SPC were 1.16 (SD = 0.09). While in block 8, it dropped to 1.09 (SD = 0.05). Significant effect of *Block* was found on SPC ($F_{7,63} = 4.32, p < .001$). This result also imples that the accuracy of auto-correction was acceptable to users, and thus can increase their confidence in using it with practice.

### *Subjective Feedback*

We now summarize some subjective feedback from user interview. Though participants were new to the circular layout at first, they could get used to it easily with the help of the visual cues (highlight and dim).

*"The visual cues significantly reduced the effort to locate the target keys." (P6)*

The interaction design (rotation and press) was intuitive and fitted well with the form factor of the smartwatch.

*"Using rotation and press is very clever and intuitive on the rotary interface of this smartwatch." (P5)*

All participants found the auto-completion feature is helpful in increasing the text entry speed.

*"The auto-completion feature is amazing! Sometimes I only have to enter half the characters to get the word." (P10)*

After the experiment, some participants even suggested implementing *COMPASS* in more scenarios.

*"I think COMPASS is a great solution for text entry when wearing gloves. I'm also looking forward to using it with the digital crown on my Apple Watch." (P3)*

Participants felt the flicking gesture easy and accurate to perform, while some participants found the *Long Press* interaction a bit hard to perform.

*"Flick to delete seemed natural to me, but the long press interaction is a bit tiring." (P7)*

## LIMITATION AND FUTURE WORK

There are several limitations of this work, which we also see as opportunities for future work.

First, the current prototype of *COMPASS* was only designed for entering words that is in the dictionary, as it covers most of the daily text entry tasks. However, it is also worthwhile for any real text entry technique to support character-level input (e.g. passwords) and entering Out-Of-Vocabulary words. In practice, by treating each individual character as a "word" in the language model, they are both achievable using the current mechanism.

Second, the current cursor placement strategy (see Equation 6) did not fully modelled user's rotation ability. For example, during interview, some participants reported that they felt clockwise rotation easier to perform than counterclockwise rotation. We speculate this was due to they were right-handed users. It is interesting to further investigate user's rotation ability, and to further improve the algorithm of *COMPASS*. In addition, although evaluation results showed that the performance of the heuristic restriction in optimization was acceptable, it is valuable to further look into the issue of input disambiguation in a more principled way.

Third, the learning curve in Experiment 2 has not converge yet. Therefore, the results may not be representative of the final text entry performance that users could achieve with *COMPASS*. Therefore, we plan to conduct a more longitudinal study to evaluate the performance of *COMPASS* in the wild.

## CONCLUSION

In this paper, we proposed *COMPASS*, a bezel-based text entry technique for non-touch smartwatches. *COMPASS* placed multiple cursors along a circular keyboard. Users rotate the bezel and press the physical button to enter text, while the position of the cursors were dynamically adjusted to minimize rotational distance. We followed an iterative design process with two user studies. In the first experiment, we showed that dynamically adjusting the position of the cursors showed outperformed fixing their positions in terms of text entry speed and subjective preference, and the optimal number of cursors is three. In the second experiment, we showed that by incorporating auto-completion algorithms, users could reach a pick-up speed of 9.3 WPM, and they could improve to 12.5 WPM after 90 minutes of practice. Some participants even reached 15.4 WPM. *COMPASS* provides a potential solution for text entry on smartwatches without using the touchscreens, and could be implemented to other rotational interfaces.

## ACKNOWLEDGEMENTS

This work is supported by the National Key Research and Development Plan under Grant No. 2016YFB1001200, the Natural Science Foundation of China under Grant No. 61303076 and No. 61272230, Tsinghua University Research Funding No. 20151080408.

## REFERENCES

1. 2016. A note on calculating text entry speed. (2016). http://www.yorku.ca/mack/RN-TextEntrySpeed.html.

2. 2016. American National Corpus. (2016). http://www.americannationalcorpus.org/OANC/index.html.
3. 2016. Apple Watch. (2016). http://www.apple.com/watch/.
4. 2016. BMW iDrive. (2016). http://www.bmw.com/com/en/insights/technology/technology_guide/articles/controller.html.
5. 2016. Moto 360. (2016). https://www.motorola.com/us/products/moto-360.
6. 2016. Pebble Watch. (2016). https://www.pebble.com/.
7. 2016. Wrist Gestures. (2016). https://support.google.com/androidwear/answer/6312406.
8. Shaikh Shawon Arefin Shimon, Courtney Lutton, Zichun Xu, Sarah Morrison-Smith, Christina Boucher, and Jaime Ruiz. 2016. Exploring Non-touchscreen Gestures for Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3822–3833. DOI: http://dx.doi.org/10.1145/2858036.2858385
9. Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2014. Both Complete and Correct?: Multi-objective Optimization of Touchscreen Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2297–2306. DOI: http://dx.doi.org/10.1145/2556288.2557414
10. Renaud Blanch and Michaël Ortega. 2009. Rake Cursor: Improving Pointing Performance with Concurrent Input Channels. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1415–1418. DOI: http://dx.doi.org/10.1145/1518701.1518914
11. Jared Cechanowicz, Steven Dawson, Matt Victor, and Sriram Subramanian. 2006. Stylus Based Text Input Using Expanding CIRRIN. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '06)*. ACM, New York, NY, USA, 163–166. DOI: http://dx.doi.org/10.1145/1133265.1133299
12. Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 2014. Swipeboard: A Text Entry Technique for Ultra-small Interfaces That Supports Novice to Expert Transitions. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 615–620. DOI: http://dx.doi.org/10.1145/2642918.2647354
13. Leah Findlater and Jacob Wobbrock. 2012. Personalized Input: Improving Ten-finger Touchscreen Typing Through Automatic Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 815–824. DOI:http://dx.doi.org/10.1145/2207676.2208520
14. Markus Funk, Alireza Sahami, Niels Henze, and Albrecht Schmidt. 2014. Using a Touch-sensitive Wristband for Text Entry on Smart Watches. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems (CHI EA '14)*. ACM, New York, NY, USA, 2305–2310. DOI: http://dx.doi.org/10.1145/2559206.2581143
15. Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI '02)*. ACM, New York, NY, USA, 194–195. DOI: http://dx.doi.org/10.1145/502716.502753
16. Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3817–3821. DOI: http://dx.doi.org/10.1145/2858036.2858242
17. Chris Harrison and Scott E. Hudson. 2009. Abracadabra: Wireless, High-precision, and Unpowered Finger Input for Very Small Mobile Devices. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology (UIST '09)*. ACM, New York, NY, USA, 121–124. DOI: http://dx.doi.org/10.1145/1622176.1622199
18. Uta Hinrichs, Holly Schmidt, Tobias Isenberg, Mark S Hancock, and Sheelagh Carpendale. 2008. Bubbletype: Enabling text entry within a walk-up tabletop installation. (2008).
19. Jonggi Hong, Seongkook Heo, Poika Isokoski, and Geehyuk Lee. 2015. SplitBoard: A Simple Split Soft Keyboard for Wristwatch-sized Touch Screens. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1233–1236. DOI: http://dx.doi.org/10.1145/2702123.2702273
20. Frederic Kerber, Tobias Kiefer, and Markus Löchtefeld. 2016. Investigating Interaction Techniques for State-of-the-Art Smartwatches. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 2540–2547. DOI: http://dx.doi.org/10.1145/2851581.2892302
21. Masatomo Kobayashi and Takeo Igarashi. 2008. Ninja Cursors: Using Multiple Cursors to Assist Target Acquisition on Large Screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08)*. ACM, New York, NY, USA, 949–958. DOI:http://dx.doi.org/10.1145/1357054.1357201
22. Luis A. Leiva, Alireza Sahami, Alejandro Catala, Niels Henze, and Albrecht Schmidt. 2015. Text Entry on Tiny QWERTY Soft Keyboards. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 669–678. DOI: http://dx.doi.org/10.1145/2702123.2702388

23. I. Scott MacKenzie and R. William Soukoreff. 2002. A Character-level Error Analysis Technique for Evaluating Text Entry Methods. In *Proceedings of the Second Nordic Conference on Human-computer Interaction (NordiCHI '02)*. ACM, New York, NY, USA, 243–246. DOI: http://dx.doi.org/10.1145/572020.572056

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

25. Jennifer Mankoff and Gregory D. Abowd. 1998. Cirrin: A Word-level Unistroke Keyboard for Pen Input. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology (UIST '98)*. ACM, New York, NY, USA, 213–214. DOI: http://dx.doi.org/10.1145/288392.288611

26. Paul Nation and Robert Waring. 1997. Vocabulary size, text coverage and word lists. *Vocabulary: Description, acquisition and pedagogy* 14 (1997), 6–19.

27. Stephen Oney, Chris Harrison, Amy Ogan, and Jason Wiese. 2013. ZoomBoard: A Diminutive Qwerty Soft Keyboard Using Iterative Zooming for Ultra-small Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2799–2802. DOI: http://dx.doi.org/10.1145/2470654.2481387

28. Katrin Plaumann, Michael Müller, and Enrico Rukzio. 2016. CircularSelection: Optimizing List Selection for Smartwatches. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. ACM, New York, NY, USA, 128–135. DOI: http://dx.doi.org/10.1145/2971763.2971766

29. Morten Proschowsky, Nette Schultz, and Niels Ebbe Jacobsen. 2006. An Intuitive Text Input Method for Touch Wheels. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, New York, NY, USA, 467–470. DOI: http://dx.doi.org/10.1145/1124772.1124842

30. Philip Quinn, Andy Cockburn, and Jérôme Delamarche. 2013. Examining the costs of multiple trajectory pointing techniques. *International Journal of Human-Computer Studies* 71, 4 (2013), 492–509.

31. Kari-Jouko Räihä and Oleg Špakov. 2009. Disambiguating Ninja Cursors with Eye Gaze. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 1411–1414. DOI: http://dx.doi.org/10.1145/1518701.1518913

32. Garth Shoemaker, Leah Findlater, Jessica Q Dawson, and Kellogg S Booth. 2009. Mid-air text input techniques for very large wall displays. In *Proc. GI'09*. Canadian Information Processing Society, 231–238.

33. R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03)*. ACM, New York, NY, USA, 113–120. DOI: http://dx.doi.org/10.1145/642611.642632

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

35. Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. 2016. Serendipity: Finger Gesture Recognition Using an Off-the-Shelf Smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 3847–3851. DOI: http://dx.doi.org/10.1145/2858036.2858466

36. Chun Yu, Yuanchun Shi, Ravin Balakrishnan, Xiangliang Meng, Yue Suo, Mingming Fan, and Yongqiang Qin. 2010. The Satellite Cursor: Achieving MAGIC Pointing Without Gaze Tracking Using Multiple Cursors. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 163–172. DOI: http://dx.doi.org/10.1145/1866029.1866056