Wrist Gestures for Single Handed Text Entry

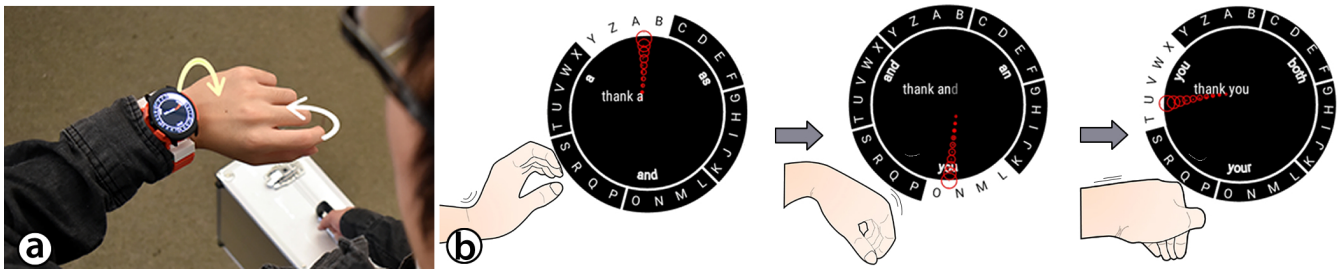

Figure 1. (a) One-handed text entry on a smartwatch by whirling the wrist; (b) to enter "you", a user selects [YZAB] → [ONML] → [TUVWX] by striking the wrist N → S → W. The entered text and suggested auto-complete are shown on the screen.

ABSTRACT

We present WrisText - a one-handed text entry technique for smartwatches using the joystick-like motion of the wrist. A user enters text by whirling the wrist of the watch hand, towards six directions which each represent a key in a circular keyboard, and where the letters are distributed in an alphabetical order. The design of WrisText was an iterative process, where we first conducted a study to investigate optimal key size, and found that keys needed to be 55º or wider to achieve over 90% striking accuracy. We then computed an optimal keyboard layout, considering a joint optimization problem of striking accuracy, striking comfort, word disambiguation. We evaluated the performance of WrisText through a five-day study with 10 participants in two text entry scenarios: hand-up and hand-down. On average, participants achieved a text entry speed of 9.9 WPM across all sessions, and were able to type as fast as 15.2 WPM by the end of the last day.

Author Keywords

Smartwatch; text entry; one-handed input;

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

INTRODUCTION

Text entry is a common and important task in daily mobile life [7], comprising of approximately 40% of mobile activity [10]. However, entering text on a smartwatch is challenging because of the small form factor and its wearable context. One of the most commonly observed problems is the need to use one or both hands for a task (e.g. driving or walking while holding an umbrella or shopping bags). This is cumbersome in the context of smartwatches, as a user is required to interrupt their ongoing task to enter text, which reduces the purposefulness of smartwatches, as they are predominantly valuable for accessing information while on-the-go.

To mitigate this problem, one solution is speech input, which is socially inappropriate in some situations (e.g., at meetings or classrooms) [61], and may also expose the users' privacy. Another solution is to enable one-handed interaction for smartwatches using the same-side hand (SSH) [31]. However, prior work has primarily been targeted at general interactions, such as assigning discrete commands to micro-interactions [36, 58], finger postures [16, 46, 63], continuous gestural input [19, 51]. One-handed text entry has been largely overlooked.

In this paper, we present WrisText, a one-handed text entry technique for smartwatches using the wrist's joystick-like motion [19] (Figure 1). With it, a user whirls the wrist of the same-side hand to strike directional marks to select keys on a circular keyboard on a smartwatch. To explore the design space of this new text entry technique, we took an iterative design approach, where we optimized the keyboard layout based on a number of factors, including keyboard learnability, striking accuracy, word disambiguation, and striking comfort. We first conducted a

target acquisition task to determine the proper size of the arc-shaped keyboard keys (e.g., 55.4˚). Based on the result, we designed a keyboard layout with six keys, containing groups of four to five English letters following an alphabetical order (Figure 1). Next, we performed a step-wise search for optimal layout variations, and identify one that balances striking accuracy, striking comfort, and word disambiguation (Figure 1). Finally, we conducted a 5-day study with ten participants to evaluate the speed and accuracy of WrisText in common smartwatch usage scenarios, such as holding the smartwatch in front of the chest and placing the hand downwards, alongside the body. Our results revealed that participants could achieve an average of 15.2 (s.e. = 0.5) WPM in the fifth day with 0.1% uncorrected errors. Extending the study by three more days with two randomly picked participants improved the speed further to 24.9 WPM.

Our contributions for this work include: (1) a one-handed text entry technique on smartwatches using the wrist's joystick motion; (2) an optimized keyboard layout design for WrisText; and (3) a demonstration of the effectiveness of WrisText through a 5-day user study.

RELATED WORK

In this section, we present existing literature in enabling one-handed interaction on smartwatches and text entry methods on mobile and wearable devices.

One-handed Interaction on Smartwatches

Research on one-handed input for smartwatches has mainly focused discrete and continuous gestural input.

One-handed discrete gestural input. This research area has mainly focused on detecting pinch (e.g. thumb touching the other fingers) [2, 6, 16, 24, 36, 47, 63] and different hand postures (e.g. fist or thumb-up) [11, 16, 18, 46, 63]. GestureWrist [46] exemplifies early work in this category. The technique uses an array of capacitive sensors to detect the changes in forearm shape to inform different hand postures. Fukui, et al. [18] and Ortega-Avila et al. [42] achieved a similar result using infrared photo reflectors. More recently, WristFlex [16] and Tomo [63] improved sensing capability using force resistors or electrical impedance tomography sensors. Skinput [24] detects pinch gestures using an array of contact microphones (e.g. piezo sensors) worn on the upper arm to detect sound waves generated by fingers tapping each other. Amento et al. [6] showed that a single piezo sensor placed in a wristband can help detect similar gestures to those detected by the commercial product, Aria [2]. Other approaches, including using EMG sensors [32, 40, 47] and cameras [12, 36], require the sensor to be worn on the upper arm (or other body parts), thus being less practical for smartwatch users.

One-handed continuous gestural input. Crossan, et al. [15] and Strohmeier, et al. [50] use wrist pronation for 1D gestural input. Rahman et al. [45] studied the number of discrete levels in each of the wrist tilt axes, the result of which can be applied to control a 2D cursor by tilting the watch in the x- and y-axes. This technique has been developed for hand-held devices [14, 25, 45, 56] and recently on a smartwatch [22]. Float [51] uses an improved tilt sensing algorithm, allowing precise control of cursor movement. Techniques such as using the watch as a peephole display [30, 59] may be used to control the cursor, but moving the screen may lead to a loss of visual contact with the screen as it moves away from a user's view. This makes such an approach unusable for text entry tasks. Waving the hand mid-air shares the same issue. With existing technologies, hand movement can be tracked using acoustic [39, 55, 62], IMU [5], WIFI [52], RFID [54] with relatively good accuracy. Letter identification accuracy can be further improved using techniques, like EdgeWrite [57]. The issue, however, is that aside from losing visual contact with the screen, moving the watch with large hand movement also may impact task completion time [30].

On the other hand, whirling the wrist [19] maintains a relatively stable screen during a gesture, ensuring constant visual contact with the display. The technique allows one-handed input to be carried out in wider smartwatch usage contexts (e.g., hanging the hand alongside the body), whereas many other techniques (e.g., tilt) may fail due to restricted hand movement.

Keyboard Layout Optimization

QWERTY [26] is the de facto standard of keyboard layout for both physical and virtual keyboards. Though it works well on a physical keyboard, it is suboptimal as a virtual keyboard layout for finger or stylus input. As such researchers have proposed various efficient alternatives [8, 35, 37, 38]. In the mobile context, a sizable amount of research has been conducted to optimize the keyboard layout for gestural typing and touch typing. As for the gestural typing, Smith, et al. [49] used Pareto front optimization to optimize keyboard layout based on three metrics, including gesture clarity, gesture speed, and similarity to QWERTY. Moreover, Bi and Zhai [9] introduced three types of QWERTY constraints in layout optimization, and investigated layout learnability and text entry performance. For the touch typing, Oulasvirta et al. [43] proposed the KALQ layout, designed to improve two thumb typing on a split keyboard.

Text Entry Methods on Mobile and Wearable Devices

The primary challenge of text entry on a smartwatch is to overcome the "fat finger problem" [53], where keys on a Qwerty keyboard of a smartwatch are too small to be selected with efficiency and precision. A large body of research has been carried out to improve text entry experience on smartwatches. Most are two-handed techniques and use finger touch as the input modality [13, 21, 23, 27, 41, 48, 61]. A common technique used in the existing literature is expanding the size of the Qwerty keyboard. For example, Zoomboard [41] requires the user to first zoom into a region containing the desired key to

expend the size of the keyboard. This makes selection easier. Splitboard [27] took a similar approach by showing a keyboard larger than the screen of the smartwatch. The issue is that half of the keyboard gets cut off by the screen edge, thus requiring the user to scroll if the desired key falls off-the-screen. DualKey [23] has keys that can be twice as big, as the technique associates each key with two letters. Tapping the key using different fingers selects different letters. TouchOne [3] is a commercially available product with a circular keyboard designed to be used by both hands. DriftBoard [48] takes a different approach by allowing a user to pan a moveable keyboard to position the desired key under a fixed cursor point.

Aside from these techniques, other approaches avoid pointing and selection on a keyboard using touch. For example, Swipeboard [13] addresses the problem using a hierarchical marking menu. With it, the user first swipes a directional mark to select a region where the desired letter resides. Once the region is selected, the user selects the desired letter by striking another directional mark. WatchWriter [21] applies the shape writing technique [33] on a smartwatch. Unlike other techniques, with which the user enters letters one by one, WatchWriter allows the user to enter an entire word by drawing a touchscreen gesture. Last but not least, Compass [61] was designed for non-touch smartwatches. The technique allows the user to enter keys on a circular layout by pointing at them using a rotary bezel of the watch screen.

Within the existing research, the most relevant work to ours is that of Katsuragawa, et al. [29] , who proposed entering text entry on a large wall display by detecting hand movement with a smartwatch's built-in IMU sensors. However, the technique was not designed for smartwatch use as it relies on an external display.

DESIGN CONSIDERATIONS

We consider the following factors for designing an efficient one-handed text entry method for smartwatches.

Screen Stability

Smartwatches already suffer from limited screen real-estate. Keeping the watch screen stable becomes even more crucial for entering text to allow a user to constantly monitor the screen content (e.g., entered text) to adjust input behavior or take necessary actions (e.g., backspacing, selecting suggestions). While it is impossible to completely eliminate screen movement when a gesture is drawn, our goal is to ensure this new text entry method can minimize screen oscillations. It is also important to have the screen in a reasonable viewing range to provide the same degree of fidelity as touch interactions. Therefore, techniques that may impact screen stability or viewing range (e.g., peephole display [30, 59] or mid-air hand gestures [39, 55, 62] are not in our considerations. Discrete input operations [16, 18, 46, 63] do not have this problem but are limited to a small number of commands insufficient for text entry.

Eyes-free Input.

While viewing range of the screen is our key design consideration, a text entry method can benefit from eyes-free input to facilitate common smartwatch use situations, such as walking with the watch hand hanging along the body. This introduces a number of challenges for input as the degree-of-freedom of the wrist is then limited by arm anatomy. For example, when the arm is hanging vertically, it may be hard to input by tilting the smartwatch screen in the x and y directions with the device's built-in IMU sensors [29], akin to when the hand is held horizontally in front of the chest. Output can be another challenge in this situation as the user loses visual contact with the screen. Feedback on the entered text can be provided via audio using a wireless headphone or vision through a near-eye display (e.g., Google Glass).

Learnability

Learning needs to be reduced for novice users to quickly transition to expert users. Advanced techniques that may lead to more efficient input will also need to be considered allowing users to develop their skills over time to achieve maximum performance in speed and accuracy. However, trade-offs may exist between efficiency and learnability. For example, letter gesture input methods (e.g., Zoomboard [41]) require less learning efforts than word gesture input methods (e.g., WatchWriter [21]) but can be slower in entering text. In our current exploration, we focus on letter gesture input for the sake of learnability.

WRISTEXT

Considering these factors, we designed our one-handed text entry techniques based on wrist whirling gestures [19]. The technique is effective for drawing common touchscreen gestures using the same-side hand in varying mobile contexts, such as standing or walking. Whirling the wrist also allows the smartwatch screen to be maintained in a relatively stable position during the gesture.

Our technique works on a round watch face by allowing a user to enter English letters by striking a sequence of directional marks on a circular keyboard (similar to a marking menu [34]) (Figure 1). The keyboard contains keys, each associated with a group of letters. Based on the user's input, the system searches (in a dictionary) all the words corresponding to the sequence of the selected keys, and provides a list of candidate words ordered by frequency of use (like T9). The user then pinches their thumb and index finger (detected using a piezo, similar to the one used in WristWhirl [19]) to switch to the selection mode, in which the first word is highlighted. If it is the desired word, the word will be committed automatically (*auto-commit*) upon the user typing the next word (e.g., striking the first letter of the next word), after which, a space will be inserted automatically. Otherwise, the user must locate the word in the list before typing the next one. Navigating the list can be carried out using pinch. The word can also be committed manually by rubbing the thumb and index finger. *Auto-complete* and *auto-correct* were

implemented following the algorithm described in [20, 61] (see details later). With auto-complete, the user can pick the desired word from the candidate list (in the selection mode) without having to input all the letters. Finally, the user can rub their fingers (thumb and index finger) twice to delete the last letter. The keyboard can be integrated into an existing app or activated using the pinch gesture.

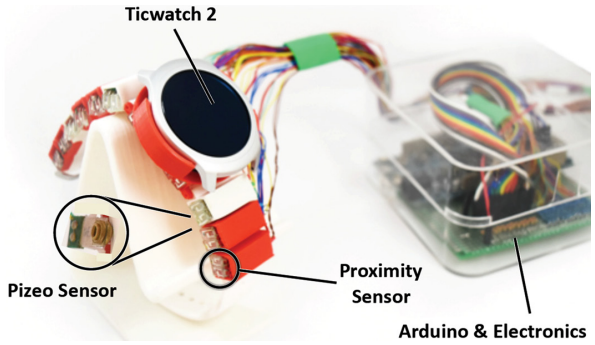

Figure 2. WrisText hardware.

WrisText Hardware

Our hardware is similar to the one presented in WristWhirl [19]. The device contains a Ticwatch 2 and a plastic watch strap augmented with 12 infrared proximity sensors (LITON LTE-301 & 302) placed approximately 0.4 cm apart from each other (Figure 2). The proximity sensor operates at 940nm with a maximum sensing distance of approximately 12cm. We connected the sensors to an Arduino DUE microcontroller, which is connected to a laptop reading sensor data at a speed of 9600 Hz. Data is then sent and visualized on the Ticwatch 2 through Bluetooth. Pinch and rub is detected using piezo vibration sensors placed inside the wrist strap [6]. Prior to using the device, it needs to be calibrated by rotating the wrist in a circular motion several times. Once the device is calibrated, the user can whirl their wrist to control the movement of a cursor to draw gestures. The wrist's joystick-like motion is tracked through the infrared proximity sensors. Tracking and cursor control was implemented following the algorithm presented in WristWhirl [19].

USER STUDY 1: KEYBOARD KEY SIZE

Keys need to be large enough to facilitate pointing; however, large keys may cause ambiguity issues when associated with multiple letters. A one-on-one mapping between keys and letters (e.g., target size of 360˚/26 = 13.8˚) is ideally unambiguous; yet our initial test with three participants suggested that such pointing resolution was nearly impossible to achieve using the wrist. The goal of this study was thus to investigate the performance of target acquisition using wrist whirling, which led to learning the optimal target size to inform our keyboard design. Further, we also investigated if a target's location affects the users' performance as well as physical comfort.

Participants

Fifteen right-handed participants (3 female) between the ages of 22 and 30 participated in the study.

Apparatus

The study was conducted using our prototype device described in the WrisText section without the Ticwatch. Participants were asked to wear the device on the wrist of their left hand during the study. As a common practice, the experimental interface was shown on a laptop screen instead of the smartwatch. The 14-inch screen had a resolution of 1920 × 1080, and was placed at a comfortable distance from participants. A computer mouse was used by participants to indicate the start of a trial.

Study Design

The experiment employed an 8 × 5 within subject factorial design. The independent variables were: (i) *Target Location*, which is defined by the eight typical gesturing directions: East, North-East, North, North-West, West, South-West, South, and South-East; and (ii) *Target Size*, which is defined as multiples of 13.8˚: 27.6˚, 41.4˚, 55.2˚, 69˚, 82.8˚. In each trial, participants performed tasks in one of the Target Location × Target Size combinations. Each condition repeated 5 times. The Target Location × Target Size combinations were randomized among trials. The experiment can be summarized as 8 Target Locations × 5 Target Sizes × 5 Repetitions × 15 Participants = 3000 trials.

Figure 3. User Study 1 setup.

Task and Procedure

Prior to the study, the device was calibrated for each participant, followed by a training session. The task required participants to select an arc-shaped target by striking the wrist towards its direction. The target with one of the five sizes was placed in one of the eight regions, within which the target was placed randomly as long its center fell inside the region. Participants were asked to perform the task in a sitting position. A trial started after participants clicked the computer mouse using their right hand. Once started, participants were instructed to perform the pointing task using their left hand as fast and as accurately as possible (Figure 3). Selection was made by crossing [4]. In other words, a target was selected once the cursor passed a threshold distance from the center of the circle. Based on our test, we set the threshold to 54.5% of the radius. A trial failed if the cursor crossed the threshold from outside the target. A trial ended regardless if the target was selected successfully or not. Upon the end of a trial, a new target of a random size appeared in a random location. We repeated this process until participants completed all the trials, at which point participants were asked to rate the level of comfort for each of the eight directions.

Results

Dependent measures included the number of errors and average task completion time. An error was recorded when

participants failed to select the target. Task completion time was recorded as the time elapsed from when the mouse is clicked to the end of a trial.

Data was analyzed using a two-way repeated measures ANOVA and Bonferroni corrections for pair-wise comparisons. For violations to sphericity, we used a Greenhouse-Geisser adjustment for degrees of freedom.

Task Completion Time

There was no significant effect of *Target Location* ($F_{3.34, 46.76} = 0.81$, $p = 0.5$) and *Target Size* ($F_{4, 56} = 1.50$, $p = 0.22$). There was also no significant interaction effect on *Target Location* × *Target Size* ($F_{28, 392} = 1.148$, $p = 0.28$).

Overall, participants spent on average 402ms (s.e. = 28ms) per strike. Interestingly, task completion time was not affected by either size or location. We attributed this result primarily to the crossing mechanism, which prevented participants from making fine adjustment once the cursor passed the threshold distance.

Accuracy

The overall average accuracy was 85.4%. ANOVA yielded a significant effect of *Target Size* ($F_{1.61, 22.47} = 78.52$, $p < 0.001$). We found no significant effect of *Target Location* ($F_{3.75, 52.43} = 2.30$, $p = 0.07$) but there was a distinct trend toward significance. We also found no significant interaction effect for *Target Location* × *Target Sizes* ($F_{28, 392} = 1.16$, $p = 0.27$). Post-hoc pair-wise comparisons revealed significant differences between all pairs of *Target Size* (all $p < 0.05$). As expected, accuracy increased with the size of the target increasing (Figure 4 right). The targets around 55.2˚ appeared to be well balanced between striking accuracy (≥ 90%) and key size.

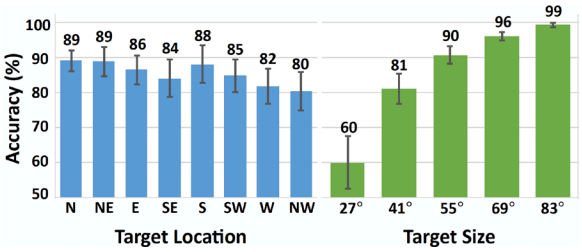

Figure 4. Left: striking accuracy shown by target location; Right: striking accuracy shown by target size. Error bars show ±2 SE in all figures.

Comfort Ratings

In addition to quantitative measurements, we also surveyed subjective ratings of the physical exertion of the eight striking directions on a continuous numeric scale: 0 (extremely easy); 1 (easy); 2 (somewhat easy); 3 (moderate); 4 (somewhat hard); 5 (hard); 6 (extremely hard)). Overall, the directional strikes were rated somewhat easy to perform (avg. = 2.6; s.e. = 0.3). One-way ANOVA yielded a significant effect of *Target Location* ($F_{3.42, 47.88} = 4.12$, $p < 0.05$). The virtual directions were rated easy to perform with the average rating for the North and South locations being 1.2 and 0.5 respectively. With respect to the 55.2˚ target, this finding is consistent with the quantitative result, which revealed a high level of accuracy in these two locations (e.g., 89% and 88%). Note that such consistency is not always the case (Figure 5). For example, the backslash directions (e.g., North-West and South-East) were rated among the hardest to strike (e.g., 3.9 and 4) but not the least in accuracy. Horizontal directions were rated somewhat easy to perform (East: 2.87 and West: 2.53) but their accuracies were on the lower end (although statistically no difference in all locations).

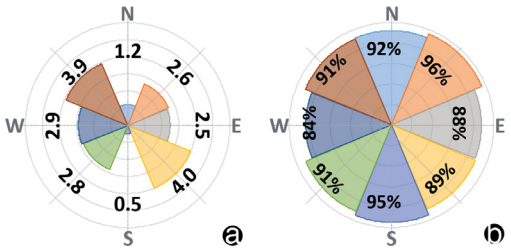

Figure 5. Comfort vs accuracy. (a) comfort ratings and (b) accuracy of the 55˚ target shown by target location.

Our observations suggest that this is mainly attributed to the inconsistency between people's perceived direction that a certain wrist strike may produce and the actual direction the wrist strike produces. Striking horizontally, for example, is harder than perceived. Participants often overshot the target and landed somewhere near the South-West and South-East locations, depending on strike direction. Training can help solve this problem. Backslash was also hard to perform, so participants required practice. Once learned, gesture performance was fairly good. This is consistent with the findings from WristWhirl [19].

To summarize, the results confirmed that aside from size, striking comfort is another important factor to consider for the design of WrisText. The level of overall striking comfort depends on the location of the target keys. There is also a trend for the location to influence target acquisition accuracy. We considered both in our design.

KEYBOARD KEY CONFIGURATION DESIGN

Study 1's results suggest that a target needs to be at least 55.2˚ for participants to maintain a reasonable pointing accuracy (e.g., ≥ 90%). Keys, however, are preferred to be small to avoid input ambiguation. With this constraint, we considered the following three types of configurations.

Figure 6. From left to right, Configuration 1,2, and 3.

Configuration 1: This configuration contains keys of the same size (e.g., 60˚ each). Figure 6a shows an example of this configuration with a minimum variance in the number of letters between keys.

Configuration 2: This configuration constrains the size of the letters (e.g., 13.8˚ each), thus keys are different in size. Figure 6b shows an example, containing 4 four-letter keys (55.2˚) and 2 five-letter keys (69˚).

Configuration 3: A variation of Configuration 2 (Figure 6c), containing 5 four-letter keys and 1 six-letter key (82.8˚).

To understand the performance difference between the options, we estimated the overall striking accuracy for each of them using the result from Study 1. For example, the overall accuracy of Configuration 2 can be estimated by averaging the accuracies of four 55.2˚ targets and two 69˚ targets, e.g., (90% × 4 + 96% × 2) / 6 = 92%. Similarly, the estimated accuracy for Configuration 3 is (90% × 5 + 99% × 1) / 6 = 91.5%. We did not have the accuracy data for Configuration 1, as the 60° target was not tested in the study. However, this missing data can be estimated by curve-fitting using a polynomial regression (Figure 7). The result revealed that the estimated accuracy of the 60˚ key is around 92%. Since the difference between the three options is small, we decided to use Configuration 1 for aesthetic reasons (as the keyboard is symmetrically divided).

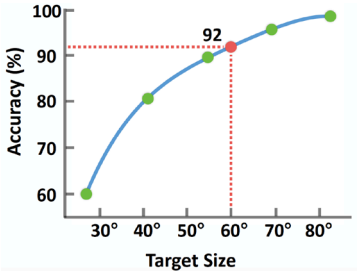

Figure 7. Accuracy regression curve.

The next step was to assign letters to the layout. We began with the first five-letter key, which can be placed in 26 different locations (each starting with one of the 26 letters). Once the first is set, the second five-letter key can only be placed at 5 locations, which also sets the locations of the rest of the keys (all four-letter keys). Thus, in total there are 26 × 5 = 130 ways of distributing the letters by bucketing them into keys. Note that half of these 130 combinations are redundant versions, generated by swapping the first and second five-letter keys yielding a total of 130 / 2 = 65 unique key configurations.

GENERAL OPTIMIZATION APPROACH

This section presents an optimization approach developed based on Smith et al.'s work [49]. The goal is to find a keyboard layout that can improve overall striking accuracy, comfort, input clarity, and is relatively easy to learn.

Optimization Metrics

We frame the problem of designing a circular keyboard layout as a multi-objective optimization problem, where the four objectives are to improve: (1) striking comfort, (2) striking accuracy, (3) word disambiguation, and (4) layout learnability.

Striking Comfort. Striking comfort considers placing frequently used letters near the locations most comfortable for users to strike.

Striking Accuracy. Frequently used letters should also be placed near locations where effective accuracy can be achieved. As suggested by Study 1, a balance needs to be stricken between comfort and accuracy.

Word Disambiguation. Each keyboard key will be associated with a number of letters. Thus, a series of key strikes may map to a set of different words. To minimize ambiguity, a brute-force search can be employed to try all the possible mappings between keys and letters, and identify the layouts that introduce minimum word ambiguation (details are discussed later).

Layout Learnability. Placing the letters in an alphabetical order can help users learn and locate the target keys quickly. Organizing the letters to the maximize comfort, accuracy, and word disambiguation may lead to an unfamiliar layout. Alternatively, a hybrid approach is swapping the keys while retaining the order of the letters inside the keys.

Optimization Procedure

To maximize the objectives based on these metrics, a step-wise search can be employed by iterating all the possible layout variations based on these metrics and their weights. Below we explain our optimization procedure.

Calculating Disambiguation Scores

For each of the 65 key configurations, a program can simulate key strikes for each word in a chosen corpus. Due to the ambiguity of words, a series of key strikes can be mapped to a list of (at least one) candidate words. This list is then ordered based on word frequency provided by the corpus. The rate of the target word that appears on the top of the list is used as the disambiguation score for each key configuration. Higher scores are preferred.

Calculating Accuracy and Comfort Scores

For each variation of the keyboard key configuration, the comfort and accuracy scores are assigned to each of the six keys. The comfort score of a key depicts how comfortable users feel when striking in the direction of that key. The accuracy score depicts how accurate users can select the key using the wrist. Both scores can be determined through an empirical study. Furthermore, each key will be assigned a weight based on the frequency of the encapsulated letters. For example, a key associated with more frequently used letters received higher weight those associated with less frequently used letters. Weight was calculated using the average letter frequency of that key since average was an intuitive way to represent the overall frequency of the letters. For example, the weight for [A B C D] is (8.7% + 2% + 4.3% + 3.8%) / 4 = 4.7%. Finally, the accuracy score for a keyboard layout was simply the sum of the weighted accuracy scores of the six keys. Comfort scores were calculated in the same manner.

Metric Normalization

Scores need to be normalized to be appropriately weighted. For example, they can be normalized in a linear fashion, so that the worst possible score is mapped to a 0 and the best possible score is mapped to a 1.

Weight Iteration

Users may weigh the three metrics differently, depending on usage scenarios. This may impact the layout of the keyboard. Therefore, the last step is to iterate different weight combinations (e.g., using a step size of 0.01) and identify an optimized layout for each combination by maximizing the following objective function.

$$F(Com, Acc, DisAmb) = \alpha \times Com + \beta \times Acc + \gamma \times DisAmb$$

where α, β, and γ is the weight of comfort (*Com*), accuracy (*Acc*) and disambiguation (*DisAmb*), and $\alpha + \beta + \gamma = 1$.

WRISTEXT KEYBOARD LAYOUT

We used the aforementioned approach to optimize our keyboard layout, where we approximated the accuracy and comfort scores using the results from Study 1. For each key, the comfort score was approximated by using the average subjective rating from the location closest to the center of the key. The accuracy score was approximated using a similar technique of selecting the accuracy data from the location closest to the center of the key. For the sake of learnability, the letters were organized in an alphabetical order, in a clockwise direction, similar to [44, 61]. Assuming the keyboard can be rotated in the clockwise direction with a resolution of 1˚, there are 23400 (65 × 360) variations of keyboard layouts. For the corpus, we used the top 15,000 words from the American National Corpus [1], which covers over 95% of common English words [61].

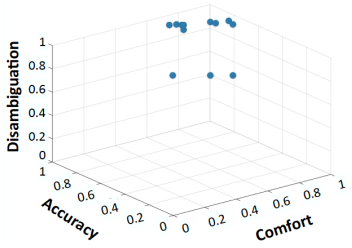

Figure 8. Optimized keyboard layouts shown by accuracy, comfort, and disambiguation.

For each weight combination, we iterated all 23400 keyboard layouts, and identified the one with the highest score from function *F* as the optimal layout. Among all the optimal layouts, many were to be identical, indicating that only a relatively small set of the layouts performed well on the three metrics. We plot these optimal layouts in Figure 8, in which the x-axis, y-axis, z-axis represent comfort, accuracy and disambiguation respectively. Since the comfort and accuracy scores were estimated using the discrete values from the 8 directions, each point in the figure represents multiple layouts with the same x, y, and z values. Each layout represents a compromise between the three metrics: a high value in one metric does not necessarily lead to similarly high values in the other metrics. The choice of layout for a user depends on the importance of each metric in different usage scenarios. A new user who is less accurate may prefer a layout biased towards accuracy, while a user who can strike accurately may prefer a layout biased toward clarity. Our result guides users to choose one to satisfy their needs.

In our case, we wished to choose a layout that balances all metrics, so we used one that is near the line where x = y = z. It represents a layout that perform equally well over the three metrics. Fifteen layouts satisfied this requirement. As expected, all share the same key configuration with only slight differences in the orientation of the dial, e.g., ±1˚ between two adjacent variations. We chose the one residing in the center of the variations (Figure 1). In comparison to the worst performance in each individual metric, this layout has 27%, 10.85% and 5.76% improvements on striking comfort, striking accuracy and word disambiguation respectively. With this layout, 85.9% of the words in the corpus are the first one in the candidate list and 95.3% were among the top three.

AUTO-COMPLETE & AUTO-CORRECT

Like many modern smart keyboards [21, 61], we developed a feature to support automatic completion and correction. Our technique built on [61] for auto-complete and [20] for auto-correct. We employed a Bayesian model to predict a target word *W* based on user's input *S* (e.g., a series of key strikes). The system then calculates the probability of *W* in a dictionary using:

$$P(W|S) \propto P(S|W) \times P(W)$$

As with many smart keyboard techniques [28], we assume that users generate no insertion or omission errors and we treat each key strike independently [17, 20]. Thus, we have:

$$P(S|W) = \prod_{i=1}^{n} P(S_i|W_i) \times \alpha^{m-n}$$

where S_i refers to the *i*th letter of the word entered by the user, and W_i is the *i*th letter of *W*. When calculating *P(S|W)* for auto-complete, we consider all *W* in the dictionary whose length is between *S* and *S+8,* with 8 being determined based on our test. Here, α is the penalty to prevent long, high-frequency words from dominating short, low-frequency words, and *m* is the length of *W,* where $m \geq n$. Thus, the larger the difference in length between *W* and *S*, the more a penalty is given to *W*. We tested α from 0.3 to 0.7 in our simulation, and found α = 0.62 yields the best compromise between the aggressiveness of auto-completion and coverage of candidates.

Auto-correct deals with cases where the user fails to select a desired key. Our estimated striking accuracy for a 60˚ key is 92%. We thus used 0.92% for $P(S_i|W_i)$ if S_i and W_i were from the same key. In the case when S_i and W_i belonged to keys that were adjacent to each other or separated by a third key, we chose 0.035 and 0.005 respectively. These numbers worked well in our present work. Future

development can infer the numbers using an experiment similar to Study 1.

USER STUDY 2: PERFORMANCE EVALUATION

We conducted a five-day user study to investigate the performance of WrisText. The goal of the study was to measure how well people could perform text entry using our technique in common smartwatch usage conditions. We were also interested in knowing how user performance can be improved through practice over time.

Participants

Ten right-handed participants (4 female) between the ages of 20 and 25 participated in the study.

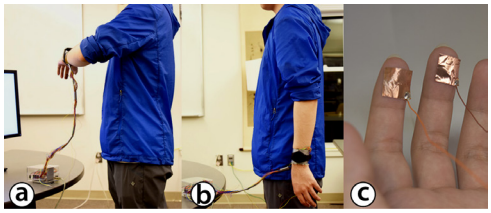

Figure 9. Study setup. (a) hand-up condition; (b) hand-down condition; and (c) capacitive sensors.

Apparatus and Task Conditions

The device was similar to the one described in the WrisText section earlier, except that we used finger-worn capacitive sensors to detect pinch gestures to eliminate errors caused by pinch detection (Figure 9). The rub gesture was replaced by pinching the thumb and middle finger. Participants were asked to wear the device on the wrist of the non-dominant left hand to perform a series of text entry tasks in two different hand postures, hand-up and hand-down (Figure 9). They were also required to perform the tasks while standing to simulate smartwatch usage situations. In the hand-up condition, participants held the watch in front of their chest. This allowed them to see the keyboard and the gesture trace they were drawing on the watch screen. The text entered by participants, along with the top three candidates generated by our software, was also shown on the screen of the smartwatch (Figure 1). In the hand-down condition, participants were required to have the watch hand hanging naturally alongside the body. In this condition, no visual feedback of the keyboard was given. Text entry was carried out eyes-free. The top three candidates and the text entered by the user were shown on a 27-inch monitor, placed at a comfortable distance from the participant. This was to simulate the situation where a near-eye display (e.g. Google Glass) is available to the user. In both conditions, test phrases were shown on the monitor.

Procedure and Design

The study consisted of a series of sessions, with one session occurring per day. In each session, participants typed 18 phrases with each hand posture, the same across all participants, randomly picked from [60]. No phrases were repeated across different sessions. Thus, we used 144 different phrases (18 phrases × 8 sessions) in the study. Prior to the experiment, participants were asked to practice for as long as they wanted. Participants were encouraged to take breaks during the session. Each experimental session lasted around 40 minutes, depending on participant speed. Hand postures were counter-balanced among participants. We collected subjective feedback from participants in the last session. In total, we collected 10 participants × 5 sessions × 18 phrases × 2 postures = 1800 phrases.

Results

We analyzed the data using a two-way repeated measures ANOVA and Bonferroni corrections for pair-wise comparisons. For violations to sphericity, we used a Greenhouse-Geisser adjustment for degrees of freedom.

Text-Entry Speed

ANOVA yielded a significant effect of *Session* ($F_{1.96, 17.65} = 311.20$, $p < 0.001$) and a marginal significance of *Hand Posture* ($F_{1, 9} = 5.21$, $p = 0.048$). There was also significant interaction effect on *Session* × *Hand Posture* ($F_{4, 36} = 7.55$, $p < 0.001$), suggesting more improvement in the hand-down condition than in the hand-up condition. Post-hoc pair-wise comparisons revealed significant differences between all pairs of *Session* (all $p < 0.05$).

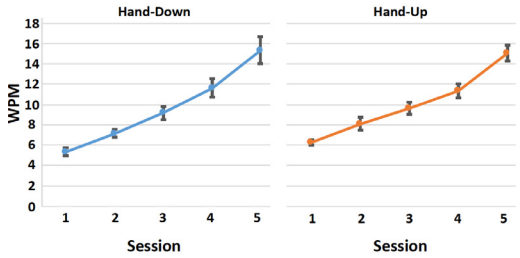

Figure 10. Mean text entry speed over 5 days.

Overall, the average text entry speed across all tested conditions was 9.9 WPM (s.e. = 1.1). In particular, participants achieved 9.7 WPM (s.e. = 1.0) and 10.1 WPM (s.e. = 0.9) in the hand-down and hand-up conditions respectively. Figure 10 shows the mean WPM by session/day. The steep curve over five days shows a substantial performance improvement. The average speed for the first session was 5.8 WPM (s.e. = 0.1). It bumped up to 15.2 WPM (s.e. = 0.5) in the last session, with an increase of 162%. As expected, it was easier for participants to start the first day with hand-up (6.3 WPM; s.e. = 0.1) than with hand-down (5.4 WPM; s.e. = 0.2) as they relied on visual cues to get familiar with the keyboard layout. However, as their skills improved over time, the speed in the hand-down condition (11.6 WPM; s.e. = 0.4) caught up in Session 4 (hand-up speed: 11.3 WPM; s.e. = 0.3). The result of this study is promising. Since the trend was still increasing, we picked two participants based on availability with gender balanced, and asked them to do three more sessions.

Figure 11 shows the data from these two participants for all eight sessions. As shown in the figure, the speed continued to increase after session 5, and finally achieved an average of 24.9 WPM (s.e. = 0.1) in session 8. The highest speed

(e.g., 27.2 WPM) was observed from P1 in the last session in the hand-down condition. After eight sessions, the average speed reached 26.9 WPM (s.e. = 0.3) in the hand-down condition and 22.8 WPM (s.e. = 0.4) in the hand-up condition. Interestingly, typing with hand-down outperformed hand-up. This is consistent with the trend from the main sessions. Our observations suggest that this is primarily because participants tended to slow down to ensure that they could strike precisely when they saw the cursor on the watch screen. This is consistent with the findings from WristWhirl [19]. However, formal investigation is needed to confirm the trend and findings.

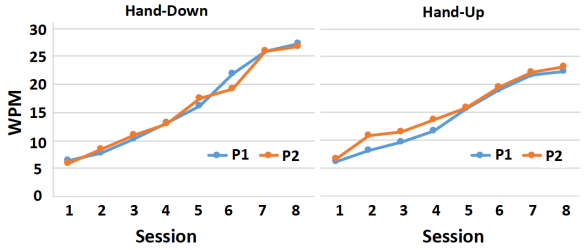

Figure 11. Text entry speed of the two extra participants over 8 days.

Error Rate

Error rate is reported based on uncorrected error rate (UER) and total error rate (TER). Uncorrected errors were the errors found in the final input phrases whereas total errors included both corrected and uncorrected errors.

For UER, ANOVA yielded a marginal significance of *Session* ($F_{2.33,\ 20.93} = 3.35$, $p = 0.048$). There was no significant effect of *Hand Posture* ($F_{1,\ 9} = 0.89$, $p = 0.37$) and no significant interaction effect on *Session* × *Hand Posture* ($F_{1.51,\ 13.55} = 1.71$, $p = 0.219$). For TER, ANOVA yielded a significant effect of *Session* ($F_{1.74,\ 15.63} = 96.19$, $p < 0.001$) and *Hand Posture* ($F_{1,\ 9} = 18.09$, $p < 0.005$). There was also a significant interaction effect on *Session* × *Hand Posture* ($F_{2.27,\ 20.46} = 8.10$, $p < 0.005$), suggesting that TER dropped quicker in the hand-down condition than in the hand-up condition. Post-hoc pair-wise comparisons showed significant differences between all pairs of *Session* (all $p < 0.05$).

Overall, the average TER and UER across all study conditions was 5.92% (s.e. = 1.43%) and 0.16% (s.e. = 0.03%) respectively. In particular, the average TER and UER was 4.98% (s.e. = 1.13%) and 0.13% (s.e. = 0.03%) in the hand-up condition, and 6.87% (s.e. = 1.75%) and 0.19% (s.e. = 0.05%) in the hand-down condition. Figure 12 shows UER and TER by *Session* in two different hand conditions. TER was higher in the hand-down condition than in the hand-up condition for the first three sessions, but reached a similar level in the last two sessions with the improvement of participants' skill. The average TER in the hand-up condition in Session 1 was 10.47%. It improved significantly after practice. As such, we expect that the pointing accuracy by striking with the wrist will also increase with more practice.

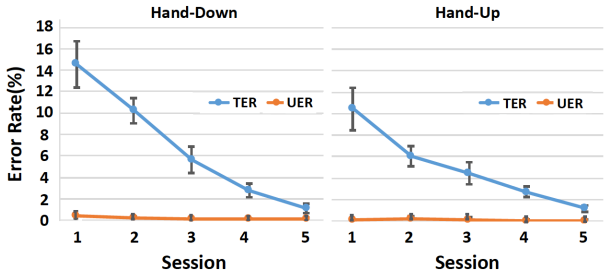

Figure 12. Mean UER and TER over 5 days.

Auto-Complete Rate

Auto-complete rating of a word was found by dividing the number of automatically filled letters by the length of that word. Thus, average auto-complete rate was the average of the auto-complete rate of all the tested words.

Overall, auto-complete accounted for 21.55% (s.e. = 1.55%) of the input words across all tested conditions with 20.43% (s.e. = 0.33%) occurring in the hand-up condition and 22.68% (s.e. = 0.99%) in the hand-down condition. For example, text entry speed without auto-complete on Day 5 was 15.2 × (100% - 21.55%) = 11.9 WPM. ANOVA yielded a significant effect of *Session* ($F_{4,\ 36} = 3.735$, $p < 0.05$) but no significant effect on *Hand Posture* ($F_{1,\ 9} = 3.97$, $p = 0.078$) and *Session* × *Hand Posture* ($F_{1.67,\ 14.99} = 2.61$, $p = 0.113$).

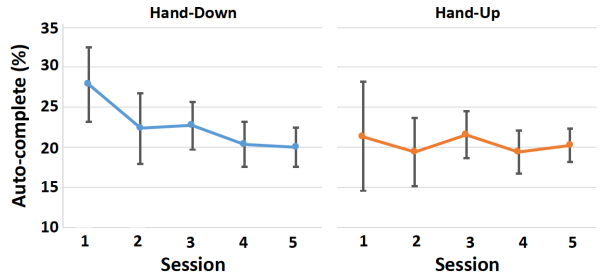

Figure 13. Mean auto-complete rate over 5 days.

Examining the auto-complete rates separately for the hand-down and hand-up conditions, we found that the significance in *Session* only existed in the hand-down condition ($F_{1.55,\ 13.95} = 60.8$, $p < 0.05$). In particular, we found a significant drop between Session 1 and Session 2 ($p < 0.05$) and no significant difference between the remaining sessions (all $p > 0.05$) (Figure 13). This suggests that participants relied on auto-completion more at the beginning when they were new to the technique, especially when the keyboard layout was not visually available. Many of the instances showed an inefficient use of auto-complete. For example, participants tended to use auto-complete even though the desired word was not on the top of the candidate list as this allowed them to strike less, thus having less of a chance to make pointing errors, but with the cost of visual search time. It did not take them long to develop sufficient confidence for their striking skills.

Subjective Feedback

Overall, participants welcomed the idea of WrisText for one-handed text entry on a smartwatch. Most were amazed

at the speed they had achieved even though some felt slightly awkward about whirling the wrist at the beginning. A participant commented *"I cannot believe I can enter text in such a fast speed, even faster than I could on my smartphone!" (P1).* Everyone agreed that auto-correct and auto-complete were useful for saving time. Interestingly, once participants were familiar with our technique and keyboard layout, they preferred to enter text with their hand-down due to the lack of fatigue. For example, a participant reported *"After I got familiar with the keyboard layout and striking directions, it was much easier for me to enter text when my hand down." (P2)*.

DISCUSSION, LIMITATIONS, AND FUTURE WORK

We present insights we learned from this work, discuss the limitations, and propose future research.

Text entry speed and error rate. The average speed of WrisText across the first five sessions is 9.9 WPM. Participants were able to achieve 15.2 WPM in the last session. This is comparable to some of the two-handed techniques. For example, the users of Zoomboard [41], Swipeboard [13], Splitboard [27] and WatchWriter [21] were able to achieve 9.8 WPM, 9.1 WPM, 15.3 WPM and 24 WPM after 15 minutes of practice. Longer practice with Zoomboard [41] and Swipeboard [13] led to 17.1 WPM and 19.6 WPM respectively. As for TER, WrisText achieved 5.92% across the five sessions. This is lower than Zoomboard (19.6%) [41] and Swipeboard (17.5%) [13] . It is also promising to observe the increase in speed after Session 5. This suggests that expert performance could be even higher, which warrants a longer-term study.

Sensor and device form factor. WrisText is based on the wrist's joystick motion so it shares the same limitations as WristWhirl [19]. For example, WrisText needs extra sensors to detect the wrist's striking motion but we believe the sensors may be found in future smartwatches. For future research, we are interested in extending this work by exploring one-handed text entry technique using the existing sensors in the smartwatches. Additionally, WrisText was designed for smartwatches with a round screen. We foresee that the concept can be extended to rectangular screens, which warrants future investigation.

Improving keyboard layout. This work makes the first attempt to design a one-handed text entry technique for smartwatches. Our current keyboard layout was optimized based on the data we collected and the algorithm we used (e.g., how the accuracy and comfort scores were assigned). It is a reasonably designed keyboard layout to demonstrate the feasibility of our technique. However, just like QWERTY, we see it as a long-term research effort to keep improving the design of our keyboard layout. For example, the 5-day study indicates pointing accuracy could be significantly improved with practice. It is thus reasonable to assume that the size of the keys can be reduced to further reduce input ambiguity. Additionally, pointing accuracy can be further improved using static decoding. We plan to collect more striking data, to understand and model users' pointing behavior. With this model, we will be able to create key configurations that can better balance striking accuracy and comfort. In this work, the keys' comfort and accuracy scores were estimated based on the location of the eight samples to the center of the key. This rough estimation can be improved by collecting data with a finer sampling resolution. Furthermore, the accuracy and comfort data were means but treated as single numbers in the layout optimization. As such, the variance in the data was overlooked. Future research will explore better optimization methods. Finally, our keyboard layout was optimized based on the data collected in the hand-up condition. It worked well in the hand-down condition but an interesting future research direction could be studying what the user preferred posture is in common smartwatch us conditions, upon which, the layout can be optimized.

Fatigue and RSI. WrisText is ideal for entering short messages over a brief period of time when the other hand is occupied. It is not meant to replace the existing two-handed text entry methods. We expect that prolonged use of wrist motion may cause RSI but users can switch back to a traditional method (e.g., touchscreen) if there is any discomfort.

User study. Aside from the two tested hand postures, many smartwatch usage scenarios warrant careful investigation (e.g., walking with the non-watch hand carrying objects). Future research will test WrisText in more usage scenarios. We will also deploy our device in the field and evaluate the effectiveness of WrisText in real-world environments.

CONCLUSION

In this paper, we proposed, designed, and studied a one-handed text entry technique on smartwatches. The technique allows users to enter text using the same hand wearing the smartwatch, by whirling the wrist in six directions to select letters in a circular keyboard. We designed the layout of the keyboard in an iterative approach, where we first studied the optimal size of the keyboard keys, and found that keys needed to be 55º or wider to achieve over 90% striking accuracy. We then optimized the keyboard layout by considering factors, including keyboard learnability, striking accuracy, striking comfort, and word disambiguation. This led to a final design which was evaluated in a 5-day study with 10 participants. The result indicates that participants could achieve an average text entry speed of 9.9 WPM across all the sessions, and were able to type as fast as 15.2 WPM in the last day. We believe smartwatches will become the major platform for mobile text entry, and our technique may serve as important groundwork for future work on new text entry techniques for wearable devices.

ACKNOWLEDGEMENTS

[Redacted in Preprint]

REFERENCES

1. American National Corpus. 2016. Retrieved April 20, 2017 from http://www.anc.org/
2. Aria Wearable. 2014. Retrieved July 15, 2017 from http://www.ariawearables.com/
3. TouchOne Keyboard. 2015. Retrieved July 15, 2017 from http://www.touchone.net/
4. Johnny Accot and Shumin Zhai. 2002. More than dotting the i's - foundations for crossing-based interfaces. In *Proceedings of the 20th Annual ACM Conference on Human Factors in Computing Systems* (CHI'02), 73-80. DOI=http://dx.doi.org/10.1145/503376.503390
5. Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves and Frank DeRuyter. 2011. Using mobile phones to write in air. In *Proceedings of the 9th international conference on Mobile Systems, Applications, and Services* (MobiSys'11), 15-28. DOI=http://dx.doi.org/10.1145/1999995.1999998
6. Brian Amento, Will Hill and Loren Terveen. 2002. The sound of one hand: a wrist-mounted bio-acoustic fingertip gesture interface. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems* (CHI EA'02), 724-725. DOI=http://dx.doi.org/10.1145/506443.506566
7. Xiaojun Bi, Tomer Moscovich, Gonzalo Ramos, Ravin Balakrishnan and Ken Hinckley. 2008. An exploration of pen rolling for pen-based interaction. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (UIST'08), 191-200. DOI=https://doi.org/10.1145/1449715.1449745
8. Xiaojun Bi, Barton A. Smith and Shumin Zhai. 2010. Quasi-qwerty soft keyboard optimization. In *Proceedings of the 28th Annual ACM Conference on Human Factors in Computing Systems* (CHI'10), 283-286. DOI=https://doi.org/10.1145/1753326.1753367
9. Xiaojun Bi and Shumin Zhai. 2016. IJQwerty: What Difference Does One Key Change Make? Gesture Typing Keyboard Optimization Bounded by One Key Position Change from Qwerty. In *Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems* (CHI'16), 49-58. DOI=https://doi.org/10.1145/2858036.2858421
10. Barry Brown, Moira McGregor and Donald McMillan. 2014. 100 days of iPhone use: understanding the details of mobile device use. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices and services* (MobileHCI'14), 223-232. DOI=http://dx.doi.org/10.1145/2628363.2628377
11. Vikram Cannanure, Xiang 'Anthony' Chen and Jennifer Mankoff. 2016. Twist'n'Knock: A One-handed Gesture for Smart Watches. In *Proceedings of the 42nd Graphics Interface Conference* (GI'16), 189-193. DOI=https://doi.org/10.20380/GI2016.24
12. Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang and Bing-Yu Chen. 2015. CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology* (UIST'15), 549-556. DOI=https://doi.org/10.1145/2807442.2807450
13. 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), 615-620. DOI=https://doi.org/10.1145/2642918.2647354
14. Andrew Crossan and Roderick Murray-Smith. 2004. Variability in Wrist-Tilt Accelerometer Based Gesture Interfaces. In *International Conference on Mobile Human-Computer Interaction* (MobileHCI'04), 144-155. DOI= https://doi.org/10.1007/978-3-540-28637-0_13
15. Andrew Crossan, John Williamson, Stephen Brewster and Rod Murray-Smith. 2008. Wrist rotation for interaction in mobile contexts. In *Proceedings of the 10th International conference on Human computer interaction with mobile devices and services* (MobileHCI'08), 435-438. DOI=http://dx.doi.org/10.1145/1409240.1409307
16. Artem Dementyev and Joseph A. Paradiso. 2014. WristFlex: low-power gesture input with wrist-worn pressure sensors. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (UIST'14), 161-166. DOI=https://doi.org/10.1145/2642918.2647396
17. Leah Findlater and Jacob Wobbrock. 2012. Personalized input: improving ten-finger touchscreen typing through automatic adaptation. In *Proceedings of the 30th Annual ACM Conference on Human Factors in Computing Systems* (CHI'12), 815-824. DOI=http://dx.doi.org/10.1145/2207676.2208520
18. Rui Fukui, Masahiko Watanabe, Tomoaki Gyota, Masamichi Shimosaka and Tomomasa Sato. 2011. Hand shape classification with a wrist contour sensor: development of a prototype device. In *Proceedings of the 13th international conference on Ubiquitous computing* (Ubicomp'11), 311-314. DOI=https://doi.org/10.1145/2030112.2030154
19. Jun Gong, Xing-Dong Yang and Pourang Irani. 2016. WristWhirl: One-handed Continuous Smartwatch Input using Wrist Gestures. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (UIST'16), 861-872. DOI=https://doi.org/10.1145/2984511.2984563

20. 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), 194-195. DOI=http://dx.doi.org/10.1145/502716.502753
21. 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 34th Annual ACM Conference on Human Factors in Computing Systems* (CHI'16), ACM, 3817-3821. DOI=https://doi.org/10.1145/2858036.2858242
22. Anhong Guo and Tim Paek. 2016. Exploring tilt for no-touch, wrist-only interactions on smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services* (MobileHCI'16), 17-28. DOI=https://doi.org/10.1145/2935334.2935345
23. Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. In *Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems* (CHI'16), 59-70. DOI=https://doi.org/10.1145/2858036.2858052
24. Chris Harrison, Desney S. Tan and Dan Morris. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the 28th Annual ACM Conference on Human Factors in Computing Systems* (CHI'10), 453-462. DOI=https://doi.org/10.1145/1753326.1753394
25. Ken Hinckley, Jeff Pierce, Mike Sinclair and Eric Horvitz. 2000. Sensing techniques for mobile interaction. In *Proceedings of the 13th annual ACM symposium on User interface software and technology* (UIST '00), 91-100. DOI=http://dx.doi.org/10.1145/354401.354417
26. Yamada Hisao. 1980. A Historical Study of Typewriters and Typing Methods: from the Position of Planning Japanese Parallels. *Journal of Information Processing*, 175-202.
27. 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), 1233-1236. DOI=https://doi.org/10.1145/2702123.2702273
28. Akiyo Kano and Janet C Read. 2009. Text input error categorisation: solving character level insertion ambiguities using Zero Time analysis. In *Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology* (BSC-HCI'09), 293-302.
29. Keiko Katsuragawa, James R. Wallace and Edward Lank. 2016. Gestural Text Input Using a Smartwatch. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (AVI'16), 220-223. DOI=https://doi.org/10.1145/2909132.2909273
30. Frederic Kerber, Antonio Krüger and Markus Löchtefeld. 2014. Investigating the effectiveness of peephole interaction for smartwatches in a map navigation task. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices and services* (MobileHCI'14), 291-294. DOI=http://dx.doi.org/10.1145/2628363.2628393
31. Frederic Kerber, Markus Lochtefeld, Antonio Kruger, Jess McIntosh, Charlie McNeill and Mike Fraser. 2016. Understanding Same-Side Interactions with Wrist-Worn Devices. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction* (NordiCHI'16), 1-10. DOI=https://doi.org/10.1145/2971485.2971519
32. Frederic Kerber, Pascal Lessel and Antonio Kruger. 2015. Same-side Hand Interactions with Arm-placed Devices Using EMG. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems* (CHI EA'15), 1367-1372. DOI=https://doi.org/10.1145/2702613.2732895
33. Per-Ola Kristensson and Shumin Zhai. 2004. SHARK 2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (UIST'04), 43-52. DOI=http://dx.doi.org/10.1145/1029632.1029640
34. Gordon P. Kurtenbach, Abigail J. Sellen and William A. S. Buxton. 1993. An empirical evaluation of some articulatory and cognitive aspects of marking menus. *Journal of Human-Computer Interaction*, 1-23. DOI=http://dx.doi.org/10.1207/s15327051hci0801_1
35. James R. Lewis, Peter J. Kennedy and Mary J. LaLomia. 1999. Development of a Digram-Based Typing Key Layout for Single-Finger/Stylus Input. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 415-419. DOI=http://dx.doi.org/10.1177/154193129904300505
36. Christian Loclair, Sean Gustafson and Patrick Baudisch. 2010. PinchWatch: a wearable device for one-handed microinteractions. In *Proceedings of the 12th international conference on Human-computer interaction with mobile devices and services* (MobileHCI'10).
37. I. Scott MacKenzie and Shawn X. Zhang. 1999. The design and evaluation of a high-performance soft keyboard. In *Proceedings of the 17th Annual ACM conference on Human Factors in Computing Systems* (CHI'99), 25-31. DOI=http://dx.doi.org/10.1145/302979.302983
38. 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), 213-214. DOI=http://dx.doi.org/10.1145/288392.288611

39. Wenguang Mao, Jian He and Lili Qiu. 2016. CAT: high-precision acoustic motion tracking. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (MobiCom'16), 69-81. DOI=https://doi.org/10.1145/2973750.2985617
40. Sachi Mizobuchi, Shinya Terasaki, Turo Keski-Jaskari, Jari Nousiainen, Matti Ryynanen and Miika Silfverberg. 2005. Making an impression: force-controlled pen input for handheld devices. In *CHI '05 Extended Abstracts on Human Factors in Computing Systems (CHI EA '05)*, 1661-1664. DOI=http://dx.doi.org/10.1145/1056808.1056991
41. 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 31th Annual ACM Conference on Human Factors in Computing Systems* (CHI'13), 2799-2802. DOI=https://doi.org/10.1145/2470654.2481387
42. Santiago Ortega-Avila, Bogdana Rakova, Sajid Sadi and Pranav Mistry. 2015. Non-invasive optical detection of hand gestures. In *Proceedings of the 6th Augmented Human International Conference* (AH'15), 179-180. DOI=http://dx.doi.org/10.1145/2735711.2735801
43. Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen and Per Ola Kristensson. 2013. Improving two-thumb text entry on touchscreen devices. In *Proceedings of the 31th Annual ACM Conference on Human Factors in Computing Systems* (CHI'13), 2765-2774. DOI=https://doi.org/10.1145/2470654.2481383
44. Morten Proschowsky, Nette Schultz and Niels Ebbe Jacobsen. 2006. An intuitive text input method for touch wheels. In *Proceedings of the 24th Annual ACM Conference on Human Factors in Computing Systems* (CHI'06), 467-470. DOI=http://dx.doi.org/10.1145/1124772.1124842
45. Mahfuz Rahman, Sean Gustafson, Pourang Irani and Sriram Subramanian. 2009. Tilt techniques: investigating the dexterity of wrist-based input In *Proceedings of the 27th Annual ACM Conference on Human Factors in Computing Systems* (CHI'09), 1943-1952. DOI=https://doi.org/10.1145/1518701.1518997
46. Jun Rekimoto. 2001. GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices. In *Proceedings of the 5th IEEE International Symposium on Wearable Computers* (ISWC'01), 21.
47. T. Scott Saponas, Desney S. Tan, Dan Morris, Ravin Balakrishnan, Jim Turner and James A. Landay. 2009. Enabling always-available input with muscle-computer interfaces. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (UIST '09), 167-176. DOI=https://doi.org/10.1145/1622176.1622208
48. Tomoki Shibata, Daniel Afergan, Danielle Kong, Beste F. Yuksel, I. Scott MacKenzie and Robert J.K. Jacob. 2016. DriftBoard: A Panning-Based Text Entry Technique for Ultra-Small Touchscreens. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (UIST'16), 575-582. DOI=https://doi.org/10.1145/2984511.2984591
49. Brian A. Smith, Xiaojun Bi and Shumin Zhai. 2015. Optimizing Touchscreen Keyboards for Gesture Typing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI'15), 3365-3374. DOI=https://doi.org/10.1145/2702123.2702357
50. Paul Strohmeier, Roel Vertegaal and Audrey Girouard. 2012. With a flick of the wrist: stretch sensors as lightweight input for mobile devices. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction* (TEI'12), 307-308. DOI=https://doi.org/10.1145/2148131.2148195
51. Ke Sun, Yuntao Wang, Chun Yu, Yukang Yan, Hongyi Wen and Yuanchun Shi. 2017. Float: One-Handed and Touch-Free Target Selection on Smartwatches. In *Proceedings of the 35th Annual ACM Conference on Human Factors in Computing Systems* (CHI'17), 692-704. DOI=https://doi.org/10.1145/3025453.3026027
52. Li Sun, Souvik Sen, Dimitrios Koutsonikolas and Kyu-Han Kim. 2015. WiDraw: Enabling Hands-free Drawing in the Air on Commodity WiFi Devices. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (MobiCom'15), 77-89. DOI=http://dx.doi.org/10.1145/2789168.2790129
53. Daniel Vogel and Patrick Baudisch. 2007. Shift: a technique for operating pen-based interfaces using touch. In *Proceedings of the 25th Annual ACM Conference on Human Factors in Computing Systems* (CHI '07), 657-666. DOI=https://doi.org/10.1145/1240624.1240727
54. Jue Wang, Deepak Vasisht and Dina Katabi. 2014. RF-IDraw: virtual touch screen in the air using RF signals. In *Proceedings of the 2014 ACM conference on SIGCOMM* (SIGCOMM '14), 235-246. DOI=https://doi.org/10.1145/2619239.2626330
55. Wei Wang, Alex X. Liu and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking* (MobiCom'16), 82-94. DOI=https://doi.org/10.1145/2973750.2973764

56. Lars Weberg, Torbjörn Brange and Åsa Wendelbo Hansson. 2001. A piece of butter on the PDA display. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems* (CHI EA'01), 435-436. DOI=http://dx.doi.org/10.1145/634067.634320
57. Jacob O. Wobbrock, Brad A. Myers and John A. Kembel. 2003. EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (UIST'03), 61-70. DOI=https://doi.org/10.1145/964696.964703
58. Katrin Wolf, Anja Naumann, Michael Rohs and Jorg Muller. 2011. Taxonomy of microinteractions: defining microgestures based on ergonomic and scenario-dependent requirements. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I* (INTERACT'11), 559-575.
59. Ka-Ping Yee. 2003. Peephole displays: pen interaction on spatially aware handheld computers. In *Proceedings of the 21th Annual ACM Conference Conference on Human Factors in Computing Systems* (CHI'03), 1-8. DOI=http://dx.doi.org/10.1145/642611.642613
60. Xin Yi, Chun Yu, Weinan Shi, Xiaojun Bi and Yuanchun Shi. 2017. Word Clarity as a Metric in Sampling Keyboard Test Sets. In *Proceedings of the 35th Annual ACM Conference on Human Factors in Computing Systems* (CHI'17), 4216-4228. DOI=https://doi.org/10.1145/3025453.3025701
61. Xin Yi, Chun Yu, Weijie Xu, Xiaojun Bi and Yuanchun Shi. 2017. COMPASS: Rotational Keyboard on Non-Touch Smartwatches. In *Proceedings of the 35th Annual ACM Conference on Human Factors in Computing Systems* (CHI'17), 705-715. DOI=https://doi.org/10.1145/3025453.3025454
62. Sangki Yun, Yi-Chao Chen and Lili Qiu. 2015. Turning a Mobile Device into a Mouse in the Air. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (MobiSys'15), 15-29. DOI=http://dx.doi.org/10.1145/2742647.2742662
63. Yang Zhang and Chris Harrison. 2015. Tomo: Wearable, Low-Cost Electrical Impedance Tomography for Hand Gesture Recognition. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology* (UIST'15), 167-173. DOI=http://dx.doi.org/10.1145/642611.642613