Exploring Tilt for No-Touch, Wrist-Only Interactions on Smartwatches

Anhong Guo
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
anhongg@cs.cmu.edu

Tim Paek
Microsoft Research
One Microsoft Way
Redmond, WA 98052
timpaek@microsoft.com

ABSTRACT
Because smartwatches are worn on the wrist, they do not require users to hold the device, leaving at least one hand free to engage in other activities. Unfortunately, this benefit is thwarted by the typical interaction model of smartwatches; for interactions beyond glancing at information or using speech, users must utilize their other hand to manipulate a touchscreen and/or hardware buttons. In order to enable no-touch, wrist-only smartwatch interactions so that users can, for example, hold a cup of coffee while controlling their device, we explore two tilt-based interaction techniques for menu selection and navigation: *AnglePoint*, which directly maps the position of a virtual pointer to the tilt angle of the smartwatch, and *ObjectPoint*, which objectifies the underlying virtual pointer as an object imbued with a physics model. In a user study, we found that participants were able to perform menu selection and continuous selection of menu items as well as navigation through a menu hierarchy more quickly and accurately with *ObjectPoint*, even though previous research on tilt for other mobile devices suggested that *AnglePoint* would be more effective. We provide an explanation of our results and discuss the implications for more "hands-free" smartwatch interactions.

Author Keywords
Wearable devices; tilt-based interaction; IMU sensors; mobile sensing; interaction techniques; smartwatch.

ACM Classification Keywords
H.5.2. Information interfaces and presentation (*e.g.*, HCI): User Interfaces: Input devices and strategies.

INTRODUCTION
Smartwatches are becoming increasingly popular, providing on-the-go music, notifications and health monitoring. Worn on the wrist, smartwatches do not require users to hold the device, leaving at least one hand free to engage in other activities. Unfortunately, this benefit is thwarted by the typical interaction model of smartwatches; for interactions beyond glancing at information or using speech, users must utilize their other (*i.e.*, non-outfitted) hand to manipulate a capacitive touchscreen (or bezel) and/or hardware buttons. The goal of our research is to promote more "hands-free" interactions so that smartwatch users can engage in a multitude of activities that define mobility, from carrying objects such as a briefcase or a cup of coffee, to performing tasks such as opening a door or driving a car. In short, we seek to enable no-touch, wrist-only smartwatch interactions. Furthermore, we seek to do this using only sensors that are standard on smartwatches, even though it might be advantageous to incorporate additional hardware, so that users can enjoy the benefits of "hands-free" interactions on devices that are commercially available today.

One obvious method of enabling no-touch, wrist-only interactions on smartwatches is to exploit speech recognition. Indeed, smartwatches on the market today feature virtual assistants that can be invoked using gestures and/or keyword spotting (*e.g.*, "Hey Siri" on Apple Watch [24], "OK Google" on Android Wear [25]). However, mobile environments often have too much noise for accurate recognition. Furthermore, for both social and privacy reasons, users may not wish to vocalize their interactions.

Another method of enabling no-touch, wrist-only interactions is to leverage tilt using built-in IMU sensors that are practically ubiquitous on smartwatches. In fact, a wealth of prior research has established tilt as a viable interaction technique for menu navigation on smartphones and tablets. One might assume that previous findings should generalize across all mobile devices. On the other hand, the way users interact with tilt on a smartwatch may be quite different than on a device they can grasp. In this paper, we put this assumption to the test. Our contributions are threefold: First, we describe how we applied previous research results on mobile devices to develop a tilt-based interaction technique which directly maps the position of a virtual pointer to the tilt angle of the smartwatch. We call this *AnglePoint*. Second, we introduce a variation called *ObjectPoint* which objectifies the underlying virtual pointer as an object imbued with a physics model. The purpose of *ObjectPoint* is to provide users with a UI metaphor for tilt control. Third, we evaluate the two interaction techniques in a controlled

MobileHCI '16, September 06 - 09, 2016, Florence, Italy
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4408-1/16/09...$15.00
DOI: http://dx.doi.org/10.1145/2935334.2935345

user study along a battery of tasks (*e.g.*, menu selection, continuous selection, navigation) at both quantitative and qualitative levels, and discuss the implications of our results for more "hands-free" smartwatch interactions.

TILT-BASED INTERACTION TECHNIQUES

In leveraging tilt for no-touch, wrist-only interactions on smartwatches, we first looked to the research literature. Since Rekimoto (1996) [20], which was one of the earliest publications to propose tilt for menu navigation and scrolling, researchers have explored a variety of tilt-based interaction techniques on mobile devices.

In angle-based tilt design, the position of the virtual pointer is mapped to the tilt position or angle. On small form factor devices, researchers have demonstrated the value of utilizing tilt for targeting [4,5], scrolling [15] and text entry [17,27,28]. In creating *AnglePoint*, we directly applied the same angle-based tilt design of previous research to a circular watch form factor, as will be further detailed in the next Section.

A variation on angle-based tilt design is rate-based tilt design, where the angular rate and speed of the underlying virtual pointer is mapped to the degree of tilt. A good example is the work of Weberg et al. [26] who created a rate-based tilt technique that moves a cursor on a PDA display similar to the way a piece of butter slides on a hot frying pan. In creating *ObjectPoint*, we were inspired by the objectification of tilt in rate-based tilt design as a method of facilitating learning. With *ObjectPoint*, we also explored how best to represent the virtual pointer as an object imbued with a physics model.

Before describing the implementation details of *AnglePoint* and *ObjectPoint*, it is important to mention that our design for the two interaction techniques benefitted from the systematic, ergonomic analysis of the design space for wrist-based interactions found in Tilt Techniques [19]. Furthermore, in choosing parameters for the two techniques, we sought to create tilt-based actions that would be natural and easy to learn, subtle and comfortable to perform, socially acceptable, and robust. Consistent with our goal of creating no-touch, wrist-only interactions, both techniques were designed to be self-contained within the user's watch-wearing wrist.

Prototype

We implemented *AnglePoint* and *ObjectPoint* for round smartwatches using the Android Wear operating system and interpreting data from the built-in gravity sensors at 50Hz. Using rule-based methods, we created a direct mapping from the sensor values to either the position of the virtual pointer or the acceleration of the virtual object.

AnglePoint

For *AnglePoint*, users tilt the watch and receive immediate visual and vibration feedback based on the absolute tilt level and position of the watch. Using the gravity sensor values in both the x- and y-axes, we calculated the tilt angle and direction of the watch and simply mapped the direction to a highlighted region. For example, if users tilt the watch outward in the 12 o'clock direction, *AnglePoint* will immediately highlight menu items in that direction.

Characteristic / Metaphor			Selector Size	Acceleration	Friction of Floor	Friction of Wall	Bouncing	Vibration
Physics based	Ball on Floor		Small	High	Low	Tunable	Yes	Yes
	Coin on Table		Large	Mid	High	Tunable	No	Yes
	Butter on Pan		Mid	Low	No	No	No	No

Figure 1. Initial list of metaphors for *ObjectPoint*.

ObjectPoint

For *ObjectPoint*, users tilt the watch to control the movement of a virtual object with an underlying physics model. The physics model can be adjusted depending on the size of the surface (*i.e.*, the watch size), the size of the object, acceleration when tilting the surface, friction of the surface, friction of the boundary as the object moves along it, as well as the object's elasticity (*i.e.*, whether it bounces) and momentum (*i.e.*, vibration and/or sound feedback when hitting the boundary).

Because making the virtual object correspond to a real-world object can drive user expectations (*i.e.*, the virtual object could serve as a UI metaphor), in our design phase, we explored three metaphors for *ObjectPoint*, each characterized by a unique set of physical properties. These include *Ball on Floor*, *Coin on Table*, and *Butter on Pan* (Figure 1). When moving the virtual object on the surface, the expressed characteristics we adjusted were speed of movement, position of the object, and vibration. Several of these metaphors also had the advantage of supporting object-specific actions, *e.g.*, bouncing a ball, or flipping a coin.

In informal usability testing of the various metaphors on an Android Wear device, we found that many physical properties were not suitable for wrist-only interactions. For example, with the high acceleration of *Ball on Floor*, users found it difficult to precisely control the position of the ball, affecting accurate selections. Likewise, with no vibration feedback in *Butter on Pan*, users found it hard to match their expectations of movement with the state of the interface. Finally, with the *Coin on Table* metaphor, if the coin was rendered to actual size (*e.g.*, the size of a penny), users found it hard to select menus with more than a few items.

Based on the above considerations, we decided not to go with a representation of a real-word object but instead to craft our own virtual object with physical properties that were more suitable for wrist-based control. For example, the virtual object in *ObjectPoint* varies its size depending on the menu resolution (Figure 2), as well as the "appropriate" amount of acceleration, friction, and vibration feedback for specific interaction moments. We set the friction so that *ObjectPoint* would be less sensitive to movement as

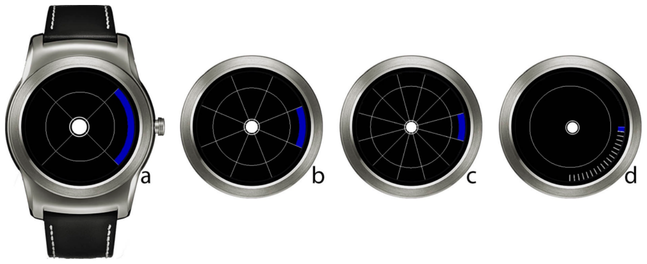

Figure 2. ***ObjectPoint*** **for menu selection with menu resolution 4, 8, 12 (a - c) and for continuous selection tasks (d).**

compared to *AnglePoint* given the shakiness of manipulating a virtual pointer on the wrist. We determined the "appropriate" values through repeated usability testing within the research team, as will be detailed later in this section. It is important to note that the amplitude and angle that users had to tilt in order to trigger actions in either *AnglePoint* and *ObjectPoint* were the same.

Start Gesture

Whether interaction involves angle- or rate-based tilt design, an important consideration in leveraging tilt at all is its reference level. We used absolute flat (all gravity on the z-axis, zero on the others) as the most natural and intuitive reference level to simulate real-world gravity. We defined *flat* as a threshold of within 5.7° from absolute flat for ease of use. Furthermore, we introduced a wrist-controlled *start gesture* to invoke smartwatch interactions based on getting the tilt of the watch face to be within this flat threshold.

Initially, we employed a one-step start gesture; namely, simply keeping the watch face flat for one second. However, when we examined sensor data for several hours of a user working inside an office, we observed too many false positives with this approach. We tried increasing the flat time to reduce false positives, but unfortunately, this made the gesture difficult to perform.

We decided to take advantage of the fact that users normally raise and tilt their wrist to look at the watch face before interacting with it to fashion a natural two-step start gesture. The start gesture is simply a concatenation of tilting the wrist inward and then making the watch face flat. To assist the user with visual feedback, we employed a familiar bubble level UI metaphor to guide the user's tilt. As shown in Figure 3a, after tilting the wrist inward over 23.6°, the screen lights up and displays a bubble level, at which point, the user simply has to get the bubble into the middle so that the watch face is flat (Figure 3b). This must be done within a specified amount of time (t_{timeout}) and the watch face must remain flat for another specified amount of time (t_{flat}).

Ideally, we want t_{flat} to be as small as possible to expedite the start of the interaction, and t_{timeout} to be as long as possible to give users enough time to flatten the watch face. To minimize false negatives, while at the same time invoking the least amount of false positives, we decided to eschew heuristics and take a data-driven approach. We first asked

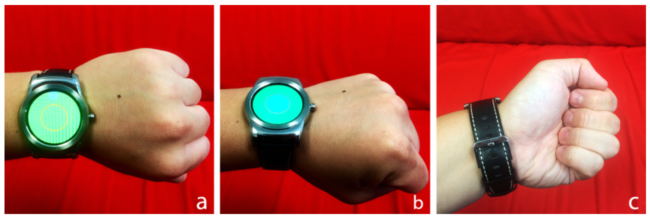

Figure 3. Start and cancel gestures. Using a bubble level metaphor, the user first (a) tilts the wrist, then (b) puts the watch to flat to start the interaction. To cancel an action, the user simply (c) puts the watch to it back.

four participants in our research group to perform the start gesture many times to see how long it would take them to flatten the watch face and stabilize. As shown in Figure 4, the majority of the trials were under 1500 milliseconds. However, to make sure we provided sufficient time to perform the gesture, we conservatively set t_{timeout} to 2500 milliseconds, capturing 93% of the trials.

Upon setting t_{timeout} to get the watch face to be flat, we then sought the shortest dwell time t_{flat} needed to produce the least false positives. As such, we collected naturalistic data from three participants in our research group. They were asked to wear a smartwatch without interacting with it for a total of 17 hours across all three participants. They engaged in a variety of activities including working in an office, walking, driving, exercising in a gym, hiking, skydiving, etc. Examining the data, we found that setting t_{flat} to 350 milliseconds produced zero false positives.

Overall, we found that our data-driven values for t_{timeout} and t_{flat} allowed users to both leisurely as well as quickly invoke the start gesture. Besides using tilt to start an interaction, users also needed a natural way to cancel actions.

Cancel Gesture

In order to provide users with a method to back out of mistakes or to cancel actions, we created a straightforward *cancel gesture*: users simply have to turn their wrist to the backside of the watch by over 36.9°, as if they were inspecting the buckle or latch of the watch's band (Figure 3c).

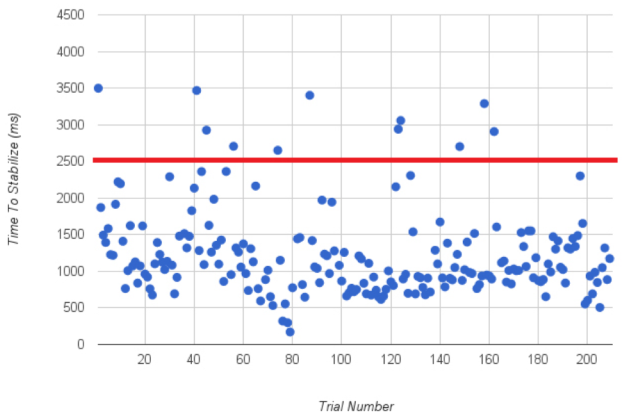

Figure 4. Scatter plot of time for all participants to put the watch to flat and stabilize after tilting the wrist.

In informal usability testing, users found this action to be natural and easy-to-learn.

Now that we have described how to start and cancel in no-touch, wrist-only smartwatch interactions, we delineate how to perform discrete and continuous menu selection as well as menu navigation using the *AnglePoint* or *ObjectPoint* interaction techniques.

Menu Selection

To make a menu selection, with *ObjectPoint*, users have to simply move the virtual object to a specific menu item. With *AnglePoint*, users have to tilt the watch in the direction of the menu item. While previous research on tilt-based interactions mostly utilized dwell-to-select as a way of providing users with more fine-grained control over selection, the disadvantage of this approach is that the dwell time imposes a constraint on how quickly interactions can proceed. For finer-grained control, we developed a novel selection technique we call *tilt-and-persist-to-select*. As the name implies, users have to first tilt or move the virtual object to an inner circle with a smaller range of motion (*i.e.*, same as the flat threshold, within 5.7°), whereupon they must continue tilting further to highlight the menu item (*i.e.*, when tilt exceeds 17.5°). To complete the menu selection, users must bring the smartwatch back to a flat position. In other words, keeping the watch tilted in some direction will not actually select the menu item. We set these parameters because they allowed users to comfortably perform interactions without losing focus on the interface.

Figure 5 demonstrates how menu selection works on a circular smartwatch with *ObjectPoint* and Figure 6 with *AnglePoint*. Our *tilt-and-persist-to-select* technique enables users to select from a list of choices by first tilting or moving the virtual object to the inner circle in the direction they intend to select (Figure 5b and Figure 6b), and then continuing to tilt further to highlight the selection (Figure 5c and Figure 6c), and finally bringing the watch back to complete the selection. Note that our *tilt-and-persist-to-select* technique could also potentially be leveraged on smartphones and tablets.

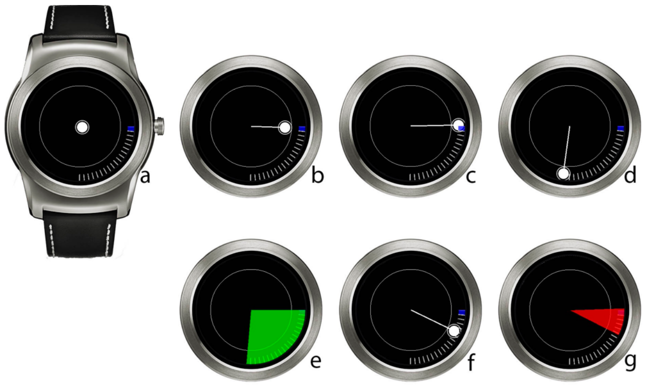

Figure 7. Continuous selection with *ObjectPoint*.

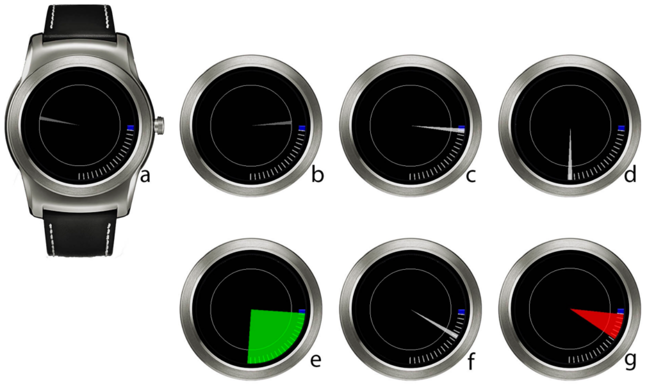

Figure 8. Continuous selection with *AnglePoint*.

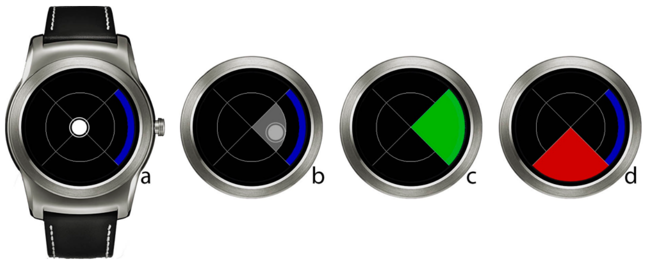

Figure 5. Menu selection with *ObjectPoint*.

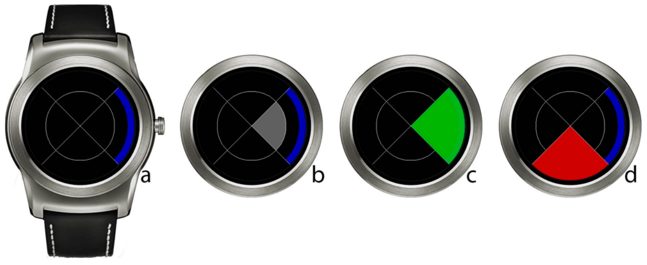

Figure 6. Menu selection with *AnglePoint*.

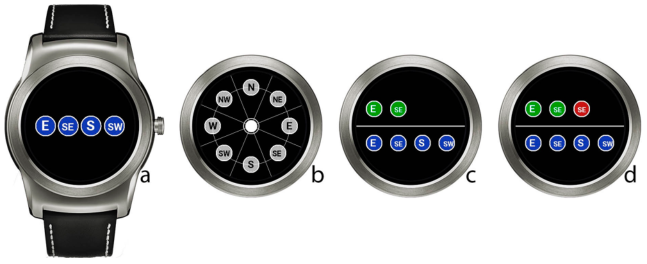

Figure 9. Navigation with *ObjectPoint*.

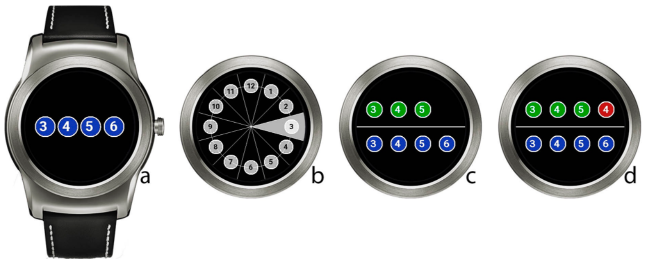

Figure 10. Navigation with *AnglePoint*.

Continuous Selection

ObjectPoint and *AnglePoint* also allow users to make a selection along a continuous range. Users first make the selection at whatever direction they intend to start the range selection (Figure 7a-c and Figure 8a-c), maintain the selection by rolling to the direction they intend to end the range (Figure 7d and Figure 8d), and finally bring the watch back from that end direction to complete the continuous selection. In the *ObjectPoint* version of continuous section, the entire action is akin to dropping a yo-yo through the inner circle and swinging it like a pendulum.

Navigation

With the menu selection techniques described above, users can navigate through a hierarchy of menus in piecemeal fashion by selecting a menu item (Figure 9b and Figure 10b), receiving visual feedback of where they are (Figure 9c and Figure 10c), making any necessary corrections with the cancel gesture (Figure 9d and Figure 10d), and continuing on to the next menu selection.

RELATED WORK

Besides previous research on tilt-based interactions for mobile devices, our interaction techniques build on several other areas of prior investigation: interaction techniques for wrist-worn devices, and bare-hand gesture interactions.

Interaction Techniques for Wrist-worn Devices

Recently, wearable devices such as smartwatches have received a great deal of attention and interest as they have become more powerful, enabling new applications through glances and micro-interactions. However, input and output on small devices like smartwatches remains limited. A number of research projects have explored increasing the input capabilities of smartwatches and increasing the amount of information shown on a smartwatch.

While speech recognition is the primary method of text input on smartwatches, as discussed previously, for both social and privacy reasons, users may prefer other options such as ZoomBoard [16] and Swipeboard [3], which use iterative zooming and swiping on the touch screen to enable text entry on ultra-small devices.

Utilizing space on wrist-worn devices to provide input has also been an active area of research. Ashbrook *et al.* [1] explored round wristwatch interactions using the bezel for displaying interfaces and touch interactions. EdgeTouch [14] demonstrated a set of grasp gestures by enabling touch sensing on the edge of the device. Xiao *et al.* [29] used the watch face as a mechanical interface to enhance interactions on smartwatches. NanoTouch [2] used touch on the backside of the small screen devices to avoid finger occlusions. Consisting of multiple touch-sensitive segments, Facet [13] enabled pose detection, as well as touch interaction to span across multiple segments.

Using space around the watch for input has also been explored. Skin buttons [11] projected icons onto the skin area around the watch. Abracadabra [7] used a finger-worn magnet and magnetometer on the device and enabled finger tacking and gesturing above the watch. Gesture Watch [10] used infrared proximity sensors to sense hand gesture made over the display.

While related research offers promising input directions for smartwatches, they do not allow users to interact with a smartwatch using only the watch wearing hand. Furthermore, many techniques require instrumenting the watch with additional hardware, unlike *ObjectPoint* and *AnglePoint*, which use the standard built-in inertial sensors.

Bare-hand Gesture Interactions

More related to our approach, people have proposed systems to provide bare-hand gesture interaction in many ways. Saponas et al. [22] demonstrated real-time finger gesture recognition for interactions using EMG signals. Skinput [8] sensed vibration on the skin to detect taps on various locations of the hand and forearm. GestureWrist [21] allowed users to interact with wearable computers using gesture-based commands on a wristband-type input device that recognizes hand gestures and forearm movements. PinchWatch [12] proposed a one-handed device with a wrist-worn display for one-handed micro-interactions by capturing gestures with a chest-worn camera.

On smartwatches specifically, Xu *et al.* [30] investigated using smartwatch-measured motion to identify users' hand and finger gestures for in-air finger-writing. Porzi *et al.* [18] recognized smartwatch-based gestures to activate functions on smartphones for visually impaired people. Google recently released updates to Android Wear to enable scrolling through notifications with a flick of the wrist [6].

One limitation of these gesture-based systems is that they are not easily scalable, and when increasing the number of supported gestures, it not only requires the users' cognitive load to remember them, but also increases the difficulty for the recognition system to work robustly. When performing gestures to control a device worn on the gesture detecting hand, users often lose sight of the interface itself. Our tilt-based techniques maintain the visual feedback of the interface by detecting small movement of the wrist; and by using control models that users are familiar with. *ObjectPoint* maps the movement with real-time interface changes, releasing the cognitive load of remembering gestures.

USER STUDY

Because the effectiveness of *AnglePoint* and *ObjectPoint*, which we denote as the independent variable *Control Model*, may depend on the number of menu items, which we denote as the independent variable *Menu Resolution*, we conducted a 2 (*Control Model*) × 3 (*Menu Resolution*) within-subjects factorial design experiment, where participants used both *AnglePoint* and *ObjectPoint* in counter-balanced order. *Menu Resolution*, however, was always presented in ascending (smallest to largest) order to allow for learning. As our dependent variables, we examined how well *Control Model* at various *Menu Resolution* facilitated menu selection, continuous selection and navigation measures. In particular, for menu selection, we measured *task completion time*, *error rate*, and the number of *failed attempts* before a success. For continuous selection, we measured *task completion time*, *error rate*, and *deviation of start and end angles*. Finally, for menu navigation, we measured *task completion time*, and number of *deletes* per task.

Hypotheses
From previous research, we held the following hypotheses:

- *ObjectPoint* will be easier to learn and understand than *AnglePoint* because people will have a better understanding of how to control their tilt with a virtual object.
- For menu selection and navigation, users will be able to perform both tasks quickly and accurately in either *ObjectPoint* or *AnglePoint* when the screen is divided into four regions. However, with higher menu resolutions where the screen is divided into eight or twelve parts, *AnglePoint* will experience greater difficulties than *ObjectPoint*.
- For continuous selection, users will have better control using *ObjectPoint* than using *AnglePoint*, again because of their understanding of the physics of the virtual object.

Participants and Apparatus
We recruited 20 participants (five females) from an IT company with an age range of 24 to 44 (M = 28.4). Their occupations included student, software engineer, IT support, and service operations. Although eight participants reported wearing watches on a daily basis, none wore smartwatches. Only one participant wore a fitness tracker.

For the experiment, we used an LG Watch Urbane running Android Wear 5.1.1. We asked participants to wear the device on whichever wrist they would normally wear a watch. 19 out of 20 participants wore the watch on their left wrist.

Procedure
After informed consent and a brief introduction, participants first learned and practiced start and cancel gestures in 15 trials. Throughout the experiment, participants utilized the start gesture to begin each session and the cancel gesture to back out of mistakes.

For the start gesture, we collected 20×46=920 testing instances of start gestures from all participants. Among all instances, only 1.63% (SD = 3.30%) of the time participants needed to perform the start gesture more than once. When asked to rate how easy it was to perform the start and cancel gestures along a Likert scale of 1 to 7, participants rated both the start gesture (M = 6.25, SD = 1.02) and cancel gesture (M = 6.30, SD = 0.80) to be very easy to perform.

For each *Control Model*, participants performed three tasks: menu selection, continuous selection, and navigation. The order of tasks was fixed since subsequent tasks depended on knowing how to perform the previous task. They were instructed to complete each task "as quickly and as accurately as possible" without sacrificing accuracy for speed and vice versa.

For *menu selection*, participants were asked to select items from circular menus with 4, 8 and 12 *Menu Resolution* items. The target menu item was highlighted in blue as shown in Figure 5a and Figure 6a. When participant correctly selected the target, visual feedback was shown in green (Figure 5c and Figure 6c); otherwise, in red (Figure 5d and Figure 6d). Participants were required to perform each menu selection task until they succeeded. During practice, participants performed menu selection of each menu item in a clockwise fashion for each of the three menu resolutions 4, 8 and 12. For testing, they completed 24 tasks in one session for each menu resolution, where the position of the menu item was randomized.

For *continuous selection*, participants were asked to select ranges of 90 degrees at clock positions along the circular boundary of the smartwatch. As shown in Figure 7a and Figure 8a, the start angle was marked with blue and the target range was delineated with white ticks at 5 degree intervals. When a participant's selected start and end angles deviated less than 45 degrees from the target angles, the selection was accepted, and visual feedback was given in green (Figure 7c-e and Figure 8c-e); otherwise, it was shown in red (Figure 7cfg and Figure 8cfg). If the selection was rejected, participants were required to perform the selection again until they succeeded. During practice, participants selected ranges of 90 degrees starting at 3 / 6 / 9 / 12 o'clock positions in both clockwise and counterclockwise directions. For testing, they completed 24 tasks of selecting 90-degree ranges, half clockwise and half counterclockwise, starting at each clock position in randomized order.

For *navigation*, participants were asked to navigate through a sequence of menu items at various resolutions; in particular, [N/E/S/W] directions for resolution scale of 4, [N/NE/E/SE/S/SW/W/NW] directions for resolution scale of 8, and [1 to 12] clock hours for resolution scale of 12. A sequence of 4 menu items was first displayed for 3 seconds (Figure 9a and Figure 10a). After participants made a selection (Figure 9b and Figure 10b), their current progress on the sequence was displayed for 1 second. When their selection matched the current digit, that digit was highlighted in green (Figure 9c and Figure 10c), otherwise in red (Figure 9d and Figure 10d). For incorrect selections, participants were required to perform the cancel gesture to delete their current menu item selection. All menu items had to be correctly entered in sequence in order to complete the task. During practice, participants completed one task for each of the three menu resolutions of 4, 8 and 12. For testing, they completed four tasks for each of the menu resolutions. We included navigation as a third task in order to simulate a full end-to-end scenario instead of only focusing on single tasks. As such, users had to navigate an interface by first performing an action, then based on feedback, plan and perform the next action.

After performing all three tasks using *ObjectPoint* or *AnglePoint*, participants were asked to complete a questionnaire where they rated the method's learnability, understandability, comfort, usefulness, social acceptability, perceived speed, accuracy, and satisfaction along a Likert scale. At the end of the study, participants were also asked to rank the two techniques along the same dimensions.

In total, the study took approximately 60 minutes and participants were compensated with corporate meal cards.

RESULTS

For menu selection and navigation tasks, we performed a factorial repeated-measures ANOVA with *Control Model* and *Menu Resolution* as independent variables. For continuous selection tasks, we used only *Control Model* as the independent variable. For our statistical analyses, we used the Greenhouse-Geisser correction for correcting violations of sphericity and post-hoc tests using a paired t-test with the Bonnferroni correction.

Menu Selection

As performance measures, we measured *task completion time*, *error rate*, and the number of *failed attempts* before a success. Task completion time was defined as the total time in milliseconds for the participants to select the target, starting from the moment instructions were displayed. Error rate was computed as the percentage of menu selection tasks that the participants failed at their first attempt. The number of failed attempts before success was also tracked to assess the difficulty of selecting the correct target.

Average Task Completion Time

We found a significant main effect of *Control Model* on average task completion time, *F*(1, 479)=182.51, *p*<.0001. Post-hoc analyses revealed that for all menu resolutions, the *ObjectPoint* model took significantly less time than the *AnglePoint*. Not surprisingly, we also found a significant main effect of *Menu Resolution* on average task completion time, *F*(1.56, 747.24)=387.42 with 12 menu items taking longer 8 menu items, and 8 taking longer than 4.

Interestingly, we found a significant interaction effect between *Control Model* and *Menu Resolution*, *F*(1.56, 747.24)=31.58, *p*<.0001. As shown in Figure 11, as the menu items increased, the disparity in average task completion time for *ObjectPoint* vs. *AnglePoint* also increased.

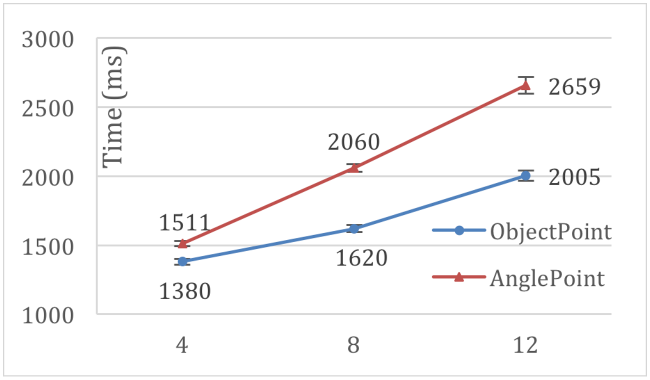

Figure 11. Average task completion time for each *Control Model* and *Menu Resolution* for menu selection tasks. Error bars represent the standard error of the mean. (N=480)

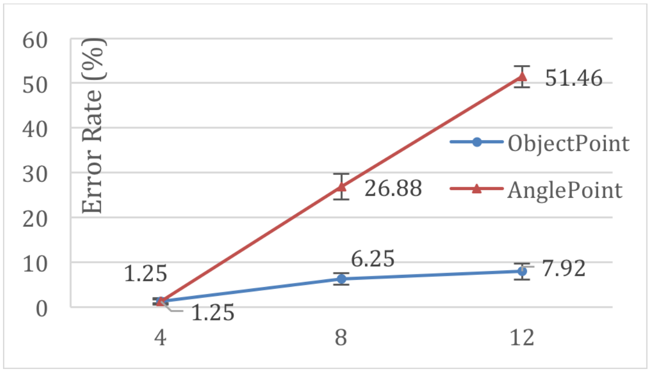

Figure 12. Average error rate for each *Control Model* and *Menu Resolution* for menu selection tasks. Error bars represent the standard error of the mean. (N=20)

Average Error Rate

We found a significant main effect of *Control Model* on average error rate, *F*(1, 19)=387.12, *p*<.0001. Post-hoc analyses revealed that for menu resolutions 8 and 12, error rate in *ObjectPoint* was significantly lower than in *AnglePoint*. Not surprisingly, we again found a significant main effect of *Menu Resolution* on average error rate, *F*(2, 38)=136.90, *p*<.0001.

Similar to the results for task completion time, we found a significant interaction effect between *Control Model* and *Menu Resolution*, *F*(1.26, 23.94)=66.93, *p*<.0001. As shown in Figure 12, we again observed that as menu resolution increased, *AnglePoint* increased the error rate significantly more than *ObjectPoint*.

Average Number of Failed Attempts Before Success

We found a significant main effect of *Control Model* on average number of failed attempts before success, *F*(1, 479)=249.79, *p*<.0001. Post-hoc analysis revealed that for

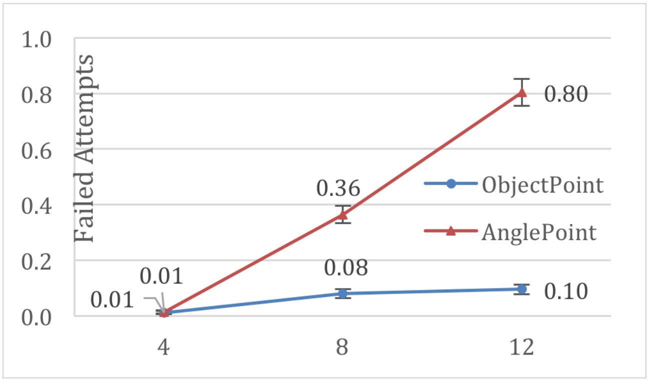

Figure 13. Average number of failed attempts before success for each *Control Model* and *Menu Resolution* for menu selection tasks. Error bars represent the standard error of the mean. (N=480)

menu resolutions 8 and 12, the number of failed attempts in *ObjectPoint* was significantly lower than in *AnglePoint*. Not surprisingly, we again found a significant main effect of *Menu Resolution* on average number of failed attempts, *F*(1.46, 728.08)=145.67, *p*<.0001.

Again, similar to task completion time and error rate, we found a significant interaction effect between *Control Model* and *Menu Resolution*, *F*(1.26, 699.34)=100.12, *p*<.0001. As shown in Figure 13, we again observed that as menu items increased, *AnglePoint* exhibited significantly more failed attempts than *ObjectPoint*.

Continuous Selection

For continuous selection, we measured *task completion time*, *error rate*, and *deviation of start and end angles* as our dependent variables. Deviations of start and end angles were measured by the degrees to which the participant's selection deviated from the target start and end angles.

Although we did not find any significant main effect of *Control Model* on task completion time, we did find that the average error rate in *ObjectPoint* (*M* = 9.58%, *SD* = 8.56%) was significantly lower than in *AnglePoint* (*M* = 28.33%, *SD* = 12.21%), *F*(1, 19)=38.67, *p*<.0001. In particular, looking more closely at where participants deviated from the target, we found that the average deviation of the end angle in *ObjectPoint* (*M* = 8.65, *SD* = 8.35) was significantly lower than *AnglePoint* (*M* = 11.34, *SD* = 10.29), *F*(1, 479)=22.50, *p*<.0001. Interestingly, we did not find a significant difference for the start angle. We discuss possible explanations in the Discussion section.

Navigation

For menu navigation, we measured *task completion time*, and number of *deletes* per task as our dependent variables.

Average Task Completion Time

We found a significant main effect of *Control Model* on average task completion time, *F*(1, 79)=135.31, *p*<.0001. Post-hoc analyses revealed that for menu resolutions 8 and 12, task completion time in *ObjectPoint* was significantly lower than in *AnglePoint*. Again, not surprisingly, we also found a significant main effect of *Menu Resolution* on average task completion time, *F*(1.56, 123.24)=168.93.

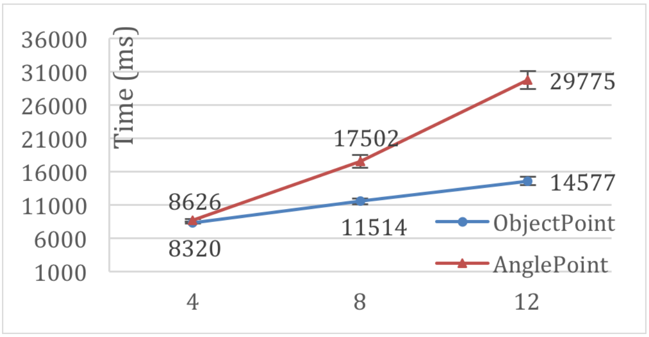

Figure 14. Average task completion time for each *Control Model* and *Menu Resolution* for navigation tasks. Error bars represent the standard error of the mean. (N=80)

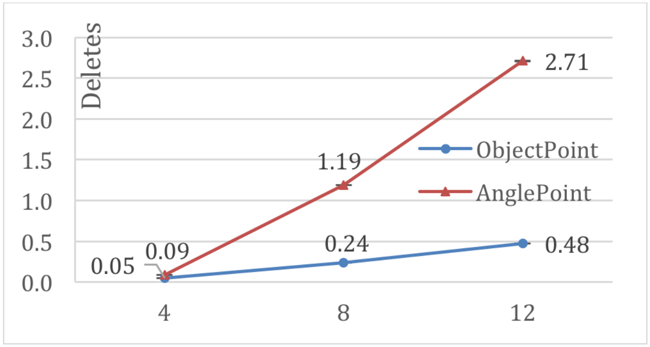

Figure 15. Average number of deletes for each *Control Model* and *Menu Resolution* for navigation tasks. Error bars represent the standard error of the mean. (N=80)

Similar to previous tasks, we found a significant interaction effect between *Control Model* and *Menu Resolution*, *F*(1.68, 132.72)=62.80, *p*<.0001. As shown in Figure 14, we again observed that as the menu items increased, the disparity in average task completion time for *ObjectPoint* vs. *AnglePoint* also increased. It is important to note that task completion time included 1 second feedback delays between actions (Figure 9cd and Figure 10cd). In real usage, users would skip the delay once they became familiar with the interface. Therefore, for navigation with four menu items in all four levels, the action time is about 5 seconds.

Average Number of Deletes

We found a significant main effect of *Control Model* on average number of failed attempts before success, *F*(1, 79)=84.49, *p*<.0001. Post-hoc analysis revealed that for menu resolutions 8 and 12, number of deletes in *ObjectPoint* was significantly lower than in *AnglePoint*. Not surprisingly, we again found a significant main effect of *Menu Resolution* on average number of failed attempts, *F*(1.56, 123.24)=59.64, *p*<.0001.

Similar to all previous tasks, we found a significant interaction effect between *Control Model* and *Menu Resolution*, *F*(1.74, 137.46)=38.81, *p*<.0001. As shown in Figure 15, as menu items increased, *AnglePoint* exhibited significantly more number of deletes than *ObjectPoint*.

Subjective Ratings

Participants' subjective ratings of the two control models are shown in Figure 16. Overall, participants rated controlling the watch using *ObjectPoint* higher, and preferred it over *AnglePoint* across all dimensions with the greatest differences in Perceived Accuracy (where the Questionnaire statement was "*For one-handed interaction, I can accurately control the watch using this method"*) and Satisfaction (where the statement was "*I would use this method to control the watch with one hand"*).

User Feedback

In open comments, participants wrote statements along the lines of: "*The [ObjectPoint] method was not only faster and*

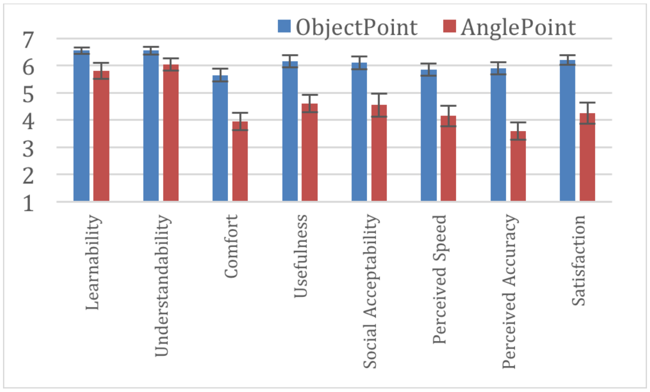

Figure 16. Likert scale ratings for *ObjectPoint* and *AnglePoint* (1 to 7, 7 as strongly agree). Error bars represent the standard error of the mean. (N=80)

more accurate, but also fun to play with. (P2)" For *AnglePoint*, we received, "*Without the ball as reference, it's more difficult to operate* (P9)" as well as "*It's surprising how this small change makes so much difference. The confidence I got from having feedback made me go faster, feeling that at any moment I can make a correction based on the feedback* (P2)". Several participants also mentioned how *ObjectPoint* could be applied for rehabilitation training "*The ObjectPoint feedback is a great training mechanism* (P10)" as well as accessibility: "*Very good idea, helpful and meaningful invention, especially for disabled group* (P15)"

DISCUSSION
We discuss the implications of our results for no-touch, wrist-only smartwatch interaction, and other factors for applying *ObjectPoint* and *AnglePoint* in a larger context.

Learnability
In our results, we observed that both *ObjectPoint* and *AnglePoint* were easy to learn and understand, even with the minimal training we provided in the studies. However, we confirmed our hypothesis that *ObjectPoint* was easier to learn and understand than *AnglePoint* based on the Likert scale ratings shown in Figure 16. From user feedback, we believe *ObjectPoint* provided users with a better understanding of how their tilt motions would be translated into interface actions due to the underlying physics model. As articulated in the user comments presented above, this gave them "confidence" to move more quickly and accurately.

Menu Selection and Navigation
Looking at the performance measures for menu selection and navigation, users were able to perform the two tasks quickly and accurately in both *ObjectPoint* and *AnglePoint* when the screen was divided into just four regions. However, as the screen was divided into eight or twelve parts (Figure to Figure 15), we observed interaction effects for all measures, including average task completion time, average error rate, average number of failed attempts before success, and average number of deletes. With higher menu resolution, *ObjectPoint* performed significant better than *AnglePoint*. We believe this interaction effect was influenced by the way in which we provided finer-grain control via *tilt-and-persist* and the inner circle; recall that user had to first tilt or move the object within the inner circle in the direction they intended to select (Figure 5b and Figure 6b), and then further tilted to make the selection. In *ObjectPoint* (Figure 5b), users can see exactly where they are currently, and make changes and corrections to the tilt based on the position of the virtual object. However, in *AnglePoint* (Figure 6b), when a menu item is highlighted, users do not know whether or not they are close to the tilt boundaries of that menu item, and when they tilt further to make the selection, they run into the risk of selecting the adjacent menu item. The risk of this happening is higher when the menu resolution is higher, since the potential distance of tilt between the current position and the boundaries are smaller, making it harder to keep the tilt within the item boundaries during *tilt-and-persist*.

Continuous Selection
For continuous selection, users demonstrated better tilt control using *ObjectPoint*, with significantly lower error rate than *AnglePoint*. However, we found a significant difference for the deviation of end angle, but not for the start angle. Since we divided the screen into 5 degree intervals, the initial selection at the start angle is similar to menu selection with a resolution of 360 / 5 = 72. At such a high resolution, the potential benefit of feedback and correction in *ObjectPoint* would not make a difference for selection, resulting in the non-significant difference in deviation of start angle. On the other hand, after the users roll to the direction they intend to end the range (Figure 7d and Figure 8d) and bring the watch back from that direction to complete their selection, for *ObjectPoint*, because of the friction between the virtual object and the outer boundary, it maintains the direction better. However, for *AnglePoint*, the end angle will always be the last absolute direction of tilt before returning back to a flat state. This could explain the significant difference in deviation of the end angle.

Contrast with Previous Research
In studies by Oakley and O'Modhrain [15], and Teather and MacKenzie [23], users were more efficient and accurate in controlling items on a menu with angle-based mapping of tilt than rate-based mapping on a PDA or tablet. Our results contradicted previous work in that *ObjectPoint*, a rate-based technique, was superior to *AnglePoint* across all metrics – to some extent, because *AnglePoint* was based on previous studies, it served as a baseline.

Furthermore, Oakley and O'Modhrain [15] observed that the primary benefit of angle-based tilt control is the lack of reliance on any hidden virtual model. We found this surprising given that the advantage of *ObjectPoint* seemed to be that the underlying physics model of the virtual object drove user expectations and helped them to make corrections. This contrast may be due to fundamental differences in the tasks; previous research investigated tilt levels along

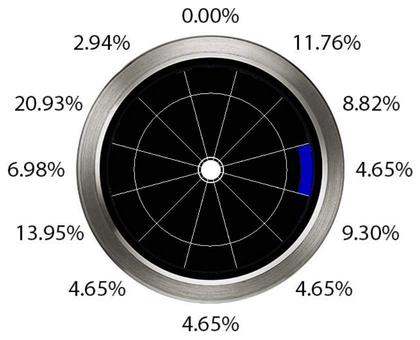

Figure 17. Average error rate of individual menu items for menu selection with menu resolution 12, using *ObjectPoint*.

one direction, while our techniques focused more on tilt along a circular menu. Further comparisons with previous research results are difficult due to the variations in experimental paradigm and conditions.

Menu Layout

Previous work has discussed discretizing functions for raw tilt angles to improve angular tilt control using linear, quadratic and sigmoid functions [19]. Our study used uniform sizes for menu items in each resolution to facilitate easier evaluations of the two *Control Mode* techniques. However, we noticed differences in difficulty for different menu item locations in our user study (Figure 17). This could be improved by applying different sizes of menus in different locations, based on ergonomic wrist capabilities [19]. We plan to investigate this further in future work.

Extending to Other Form Factors

Even though a bevy of round watches are being released to the commercial market (*e.g.*, LG Watch Urbane, Moto 360, Huawei Watch, LG G Watch R), their user interface designs are still mostly derived and modified from square watches, such as simply moving the UI elements inside the circular boundary. Our techniques take full advantage of the round form factor with interactions specifically designed for circular menus (*e.g.*, continuous selection along the circular physical boundary). That said, we believe *ObjectPoint* and *AnglePoint* can be extended to other form factors. For example, for square smartwatches, the inner circle and outer circle can be adapted to inner square and outer square for selection. Furthermore, the edges of the physical screen can still be used to select a range. Even for wristbands (*e.g.*, Microsoft Band), *ObjectPoint* can be applied with one-dimensional movement and actions, similar to the metaphor of sliding a piece of butter on a hot pan [26].

Example Application

Taking advantage of *ObjectPoint* interactions and interface design, we developed a demo music application as shown in Figure 18. The screen is divided into four parts: users can switch to the previous and next songs using menu selection techniques for the left and right regions, users can toggle pause and play by highlighting the top region, and finally,

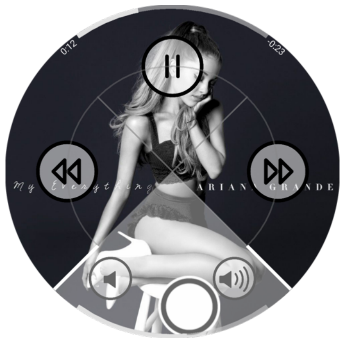

Figure 18. A music application taking advantage of *ObjectPoint* interactions.

they can control volume by using continuous selection in the bottom region.

FUTURE WORK

We see many avenues for future work. *ObjectPoint* and *AnglePoint* use a start gesture to invoke users to start the interactions when the watch is flat. However, as mentioned by one of our participants, it might be hard to keep the watch face flat in all scenarios, *e.g.*, when riding on a bumpy bus, or lying on the bed. If users are in situations such as riding on a bumpy bus, changing the physical properties of the object could be employed to compensate for errors, *e.g.*, decreasing acceleration of the virtual object and increasing simulated friction of the surface. On the other hand, if users are lying on the bed, where they might want to define a more neutral position, other gestures could potentially be used to redefine the neutral level.

Another avenue for future work is enabling eyes-free, no-touch, wrist-only interactions with *ObjectPoint*. For this, we would need to provide appropriate audio and haptic feedback that matched the physics model. With our current implementation of *ObjectPoint*, the tactile feedback could not be localized, and we didn't integrate audio feedback. Therefore, when taking away the visual feedback, which is the only thing left, there's no difference between the two models, and users have nothing to rely on to assist the interaction. As future work, we would like to investigate how localized tactile feedback (*e.g.*, Skin Drag Displays [9]), and sound associated with the movement of an object could support better wrist-only eyes-free smartwatch interactions.

CONCLUSION

In this paper, we presented two tilt-based interaction techniques for no-touch, wrist-only interactions on smartwatches: *AnglePoint*, which directly maps the position of a virtual pointer to the tilt angle of the smartwatch, and *ObjectPoint*, which objectifies the underlying virtual pointer as an object imbued with a physics model. We evaluated the two interaction techniques in a user study, and found that *ObjectPoint* was superior to *AnglePoint* across all metrics. The techniques presented in this paper can be utilized to promote more "hands-free" smartwatch interactions.

REFERENCES

1. Daniel Ashbrook, Kent Lyons, and Thad Starner. 2008. An investigation into round touchscreen wristwatch interaction. In *Proceedings of the 10th international conference on Human computer interaction with mobile devices and services* (MobileHCI '08). ACM, New York, NY, USA, 311-314. http://doi.acm.org/10.1145/1409240.1409276
2. Patrick Baudisch and Gerry Chu. 2009. Back-of-device interaction allows creating very small touch devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 1923-1932. http://doi.acm.org/10.1145/1518701.1518995
3. 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. http://doi.acm.org/10.1145/2642918.2647354
4. Andrew Crossan, and Roderick Murray-Smith. "Variability in wrist-tilt accelerometer based gesture interfaces." *Mobile Human-Computer Interaction-MobileHCI 2004*. Springer Berlin Heidelberg, 2004. 144-155.
5. 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). ACM, New York, NY, USA, 435-438. DOI=http://dx.doi.org/10.1145/1409240.1409307
6. Google. 2015. Wrist gestures with Android Wear. Video. (27 April 2015.). Retrieved September 18, 2015 from https://www.youtube.com/watch?v=_R0qbB4hVbU
7. 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. http://doi.acm.org/10.1145/1622176.1622199
8. Chris Harrison, Desney Tan, and Dan Morris. 2010. Skinput: appropriating the body as an input surface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '10). ACM, New York, NY, USA, 453-462. http://doi.acm.org/10.1145/1753326.1753394
9. Alexandra Ion, Edward Jay Wang, and Patrick Baudisch. 2015. Skin Drag Displays: Dragging a Physical Tactor across the User's Skin Produces a Stronger Tactile Stimulus than Vibrotactile. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15). ACM, New York, NY, USA, 2501-2504. http://doi.acm.org/10.1145/2702123.2702459
10. Jungsoo Kim, Jiasheng He, Kent Lyons, and Thad Starner. 2007. The Gesture Watch: A Wireless Contact-free Gesture based Wrist Interface. In *Wearable Computers, 2007 11th IEEE International Symposium on*. 15–22. http://dx.doi.org/10.1109/ISWC.2007.4373770
11. Gierad Laput, Robert Xiao, Xiang 'Anthony' Chen, Scott E. Hudson, and Chris Harrison. 2014. Skin buttons: cheap, small, low-powered and clickable fixed-icon laser projectors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology* (UIST '14). ACM, New York, NY, USA, 389-394. http://doi.acm.org/10.1145/2642918.2647356
12. 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 workshop on Ensembles of on-body devices* (MobileHCI '10) ACM, New York, NY, USA.
13. Kent Lyons, David Nguyen, Daniel Ashbrook, and Sean White. 2012. Facet: a multi-segment wrist worn system. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (UIST '12). ACM, New York, NY, USA, 123-130. http://doi.acm.org/10.1145/2380116.2380134
14. Ian Oakley and Doyoung Lee. 2014. Interaction on the edge: offset sensing for small devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 169-178. http://doi.acm.org/10.1145/2556288.2557138
15. Ian Oakley and Sile O'Modhrain. 2005. Tilt to scroll: evaluating a motion based vibrotactile mobile interface. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*. 40–49. http://dx.doi.org/10.1109/WHC.2005.138
16. 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. http://doi.acm.org/10.1145/2470654.2481387
17. Kurt Partridge, Saurav Chatterjee, Vibha Sazawal, Gaetano Borriello, and Roy Want. 2002. TiltType: accelerometer-supported text entry for very small devices. In *Proceedings of the 15th annual ACM symposium on User interface software and technology* (UIST '02). ACM, New York, NY, USA, 201-204. http://doi.acm.org/10.1145/571985.572013

18. Lorenzo Porzi, Stefano Messelodi, Carla Mara Modena, and Elisa Ricci. 2013. A smart watch-based gesture recognition system for assisting people with visual impairments. In *Proceedings of the 3rd ACM international workshop on Interactive multimedia on mobile & portable devices* (IMMPD '13). ACM, New York, NY, USA, 19-24. http://doi.acm.org/10.1145/2505483.2505487

19. Mahfuz Rahman, Sean Gustafson, Pourang Irani, and Sriram Subramanian. 2009. Tilt techniques: investigating the dexterity of wrist-based input. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 1943-1952. http://doi.acm.org/10.1145/1518701.1518997

20. Jun Rekimoto. 1996. Tilting operations for small screen interfaces. In *Proceedings of the 9th annual ACM symposium on User interface software and technology* (UIST '96). ACM, New York, NY, USA, 167-168. http://doi.acm.org/10.1145/237091.237115

21. Jun Rekimoto. 2001. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In *Wearable Computers, 2001 fifth IEEE International Symposium on*. 21–27. http://dx.doi.org/10.1109/ISWC.2007.4373770

22. 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). ACM, New York, NY, USA, 167-176. http://doi.acm.org/10.1145/1622176.1622208

23. Robert J. Teather and I. Scott MacKenzie. 2014. Position vs. velocity control for tilt-based interaction. In *Proceedings of Graphics Interface 2014* (GI '14). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 51-58. http://doi.acm.org/10.1145/2619648.2619658

24. Apple Watch. 2015. Retrieved September 23, 2015 from http://www.apple.com/watch/watch-reimagined/

25. Android Wear. 2015. Retrieved September 23, 2015 from https://www.android.com/wear/

26. 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). ACM, New York, NY, USA, 435-436. http://doi.acm.org/10.1145/634067.634320

27. Daniel Wigdor and Ravin Balakrishnan. 2003. *TiltText*: using tilt for text input to mobile phones. In *Proceedings of the 16th annual ACM symposium on User interface software and technology* (UIST '03). ACM, New York, NY, USA, 81-90. http://doi.acm.org/10.1145/964696.964705

28. John Williamson, and Roderick Murray-Smith. "Hex: Dynamics and probabilistic text entry." *Switching and Learning in Feedback Systems*. Springer Berlin Heidelberg, 2005. 333-342.

29. Robert Xiao, Gierad Laput, and Chris Harrison. 2014. Expanding the input expressivity of smartwatches with mechanical pan, twist, tilt and click. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '14). ACM, New York, NY, USA, 193-196. http://doi.acm.org/10.1145/2556288.2557017

30. Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition using Smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* (HotMobile '15). ACM, New York, NY, USA, 9-14. http://doi.acm.org/10.1145/2699343.269935